IBM RISC System/6000
Enterprise Server S70 Advanced c/s

using

Oracle8 Enterprise Edition 8.0.5

IBM TXSeries 4.2 for AIX

TPC Benchmark™ C

Full Disclosure Report

IBM System Performance and Evaluation Center

Submitted For Review
June 30, 1999

Special Notices

The following terms used in this publication are trademarks of **International Business Machines** Corporation in the United States and/or other countries:

RISC System/6000
AIX
IBM

The following terms used in this publication are trademarks of other companies as follows:

TPC Benchmark	Trademark of the Transaction Processing Performance Council
ORACLE, SQL*DBA, SQL*Loader	Trademark of Oracle, Inc.
Oracle8,SQL*Net and SQL*Plus	Trademark of Oracle, Inc.

First Edition June 30, 1999

Request for additional copies of this document should be sent to the following address:

TPC Benchmark Administrator
IBM Commercial Performance
Mail Stop 9571
11400 Burnet Road
Austin, TX 78758
FAX Number (512) 838-1852

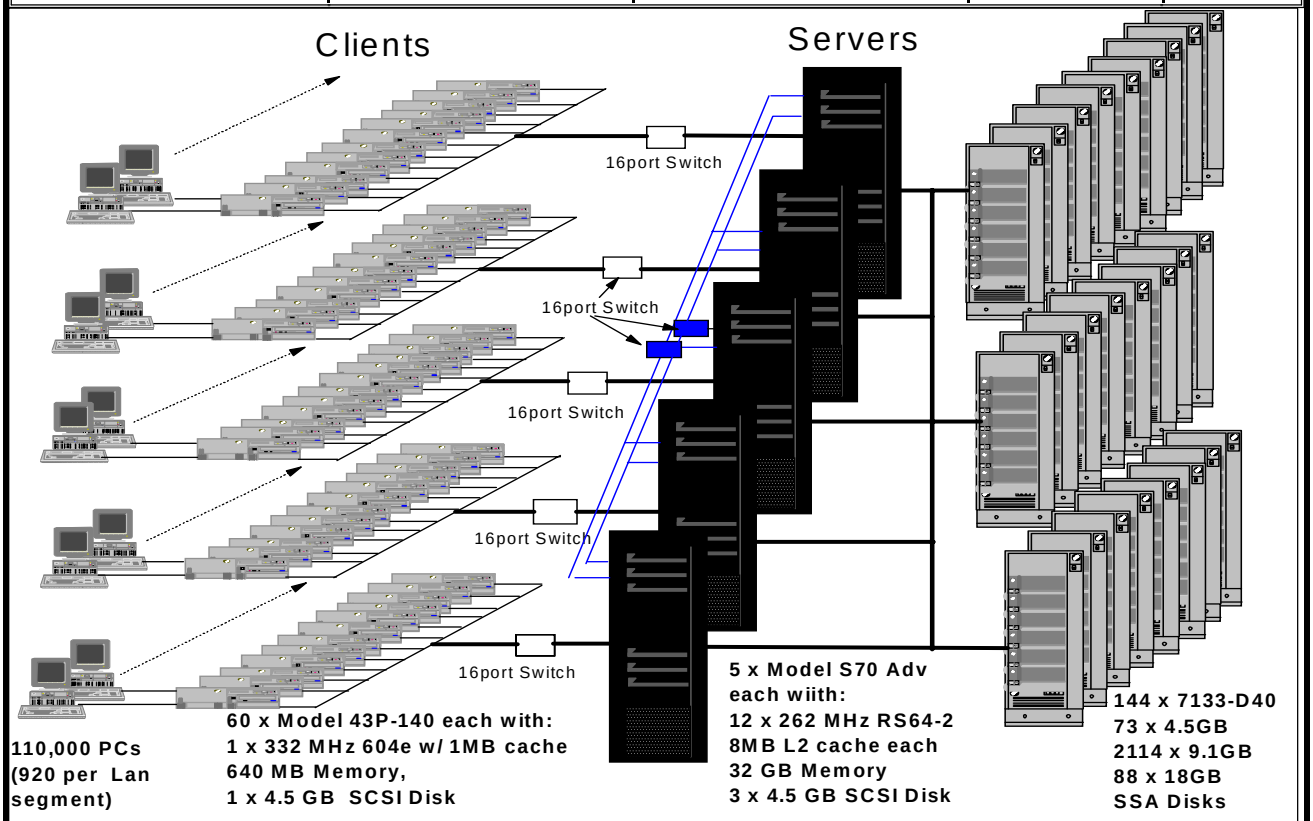
IBM RS/6000 Enterprise Server S70 Adv Cluster

TPC-C Rev. 3.4

Report Date: June 30, 1999

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$13,521,883	110,434.1 tpmC	$122.44/tpmC	NOW

Processors	Database Manager	Operating System	Other Software	No. Users
60 x IBM RS64-II	Oracle Version 8.0.5	AIX 4.3.2	TXSeries 4.2 for AIX	110,000



System Components	Clients		Server	
	Quantity	Description	Quantity	Description
Processor	60	332 MHz PowerPC 604e w/ 1MB L2 cache each	5 x 12	262 MHz IBM RS64-II w/ 8MB L2 cache each
Memory	60 x 640 MB		5 x 32 GB	
Disk Controllers	60	SCSI-2 Adapters	5 x 3 5 x 26	SCSI-2 Adapters SSA Adapters
Disk Drives	61	4.5 GB SCSI	2,114 73 88 5 x 3	9.1 GB SSA Disks 4.5 GB SSA Disks 18 GB SSA Disks 4.5 GB SCSI Disks
Total Storage	4.5 GB each client		19,764.69 GB	
Terminals	60	System Console	1	System Console
Term. Connect	15,180	CentreCOM 8-Port Ethernet Hubs + 10% spares		

IBM RS/6000 Enterprise Server S7A c/s

TPC-C Rev. 3.4

Report Date: June 30,1999

Description	Part Number	Source	Unit Price	Qty	Extended Price	5 yr. Maint Price
Server Hardware						
RS/6000 Server Model S7A	7017-S7A	1	15,036	5	75,180	244,800
3 SCSI Adapters, Rack,						
CD-ROM, SCSI Hot Swap 6-pack						
4.5 GB 1" Ultra SCSI Hot Swap	2913	1	1,400	15	21,000	0
Async/Terminal Cable	2934	1	45	5	225	0
10/100 Mbs Ethernet PCI Adapter	2968	1	275	20	5,500	0
4 x 2048 MB R1 Memory	4179	1	98,304	20	1,966,080	0
4way RS64 II 262 Mhz Proc, 8MB L2	5312	1	75,000	10	750,000	806,400
4way RS64 II 262 Mhz Proc, 8MB L2	5313	1	75,000	5	375,000	201,600
Advanced SerialRAID Adapter	6225	1	3,000	130	390,000	0
I/O Drawer	6320	1	23,766	20	475,320	0
Primary I/O Drawer Group	6321	1	600	5	3,000	0
Support Processor Group	6322	1	2,700	15	40,500	0
Secondary I/O Drawer Group	6323	1	734	15	11,010	0
I/O Drawer Rack-Rack, Drawer Cabl	3126	1	3,175	5	15,875	0
I/O Rack	7000	1	3,500	10	35,000	0
System Rack Model R00	7015-R00	1	3,110	25	77,750	37,200
Additional Power Distribution Unit	6171	1	1,000	25	25,000	0
SSA Disk Subsystem, Black Cover	7133-D40	1	13,000	144	1,872,000	1,382,400
50/60 Hz AC, 300 VDC Power Supply	8022	1	2,000	144	288,000	0
4.5 GB Disk Drive Modules	8204	1	1,900	73	138,700	0
9.1 GB Disk Drive Modules	8209	1	2,760	2,114	5,834,640	0
18 GB Disk Drive Modules	8218	1	4,500	88	396,000	0
SSA Cables	8801	1	95	576	54,720	0
			Subtotal		12,850,500	2,672,400
Server Software						
AIX 4.3 for S7A	5765-C34	1	300	5	1,500	0
AIX C compiler	5765-C64	1	1,578	1	1,578	0
Performance Toolbox	5765-654	1	1,300	1	1,300	0
HACMP Conc Resource Mgr	5765-A86	1	20,000	5	100,000	0
Oracle Version 8.0.5 OPS		2	1,850,480	1	1,850,480	1,850,480
			Subtotal		1,954,858	1,850,480
Client Hardware						
RS/6000 Model 43P-140, 332 MHz	7043-140	1	7,295	60	437,700	201,600
4.5gb Disk, Intg SCSI-2 FW, Intg Enet, 1MB L2 cache						
4.5 GB SCSI Internal Disk	2900	1	1,300	1	1,300	0
128MB DIMM Memory	4102	1	448	60	26,880	0
128MB DIMM Memory Expansion	4115	1	896	240	215,040	0
Async Terminal/Printer Cable	2934	1	45	60	2,700	0
10/100 Mbps Ethernet Adapter, PCI	2986	1	275	180	49,500	0
IBM ASCII Terminal, Keyboard	3153-BG3	1	577	65	37,505	0
			Subtotal		770,625	201,600
Client Software						
AIX 4.3.2 Unlimited Users	5756-C34	1	4,750	60	285,000	0
AIX C compiler	5765-C64	1	1,578	1	1,578	0
IBM TXSeries 4.2 for AIX	5697-D17	1	3,353	60	201,156	70,500
			Subtotal		487,734	70,500
User Connectivity						
16port Ethernet Switch w/40% disc	Nbase	4	1,560	9	14,040	2,808
Ethernet Hub (8port) + 10% Spares	DEH2924	3	36.00	15180	546,480	0
			Subtotal		560,520	2,808
			Discounts		(7,006,041)	(894,101)
				Total	9,618,197	3,903,687

Notes:

Pricing Sources:
1=Keylink, 2=Oracle, 3=Data Comm Warehouse, 4=Nbase/Xyplex

Audited by: Francois Raab , Info Sizing

Five-Year Cost of Ownership:	13,521,883
tpmC	110,434.10
$/tpmC	122.44

Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you

Numerical Quantities Summary for the IBM RS/6000 S70 Advanced

MQTH, computed Maximum Qualified Throughput: 110,434.1 tpmC

Repeatability Run: 109,768.0 tpmC 0.61% difference

Response Times (in seconds)	90th %	Average	Maximum
New Order	4.40	2.58	315.41
Payment	3.95	2.30	364.41
Order-Status	3.95	2.31	276.22
Delivery (interactive portion)	0.19	0.11	3.69
Delivery (deferred portion)	14.84	23.83	6215.33
Stock-Level	11.7	5.63	1346.82
Menu	0.01	0.01	2.23

Transaction Mix, in percent of total transactions	Percent
New Order	44.77%
Payment	43.07%
Order-Status	4.05%
Delivery	4.04%
Stock-Level	4.05%

Keying/Think Times (in seconds)	Min.	Average	Max.
New Order	18.00/0.01	18.01/15.31	18.14/153.20
Payment	3.00/0.01	3.01/15.32	3.15/153.20
Order-Status	2.00/0.01	2.01/13.32	2.14/133.20
Delivery	2.00/0.01	2.01/8.32	2.14/83.20
Stock-Level	2.00/0.01	2.01/8.33	2.14/83.20

Test Duration

Ramp-up Time	1 hr. 16 min. 0 secs
Measurement interval	2 hrs. 5 min
Transactions during measurement interval (all types)	30,831,969
Ramp-down time	20 minutes

Checkpointing

Number of checkpoints	Greater than 4 per node
Checkpoint interval	25 minutes

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification Revision 3.4 dated June 25, 1998, for measurements on the cluster of IBM RISC System/6000 Enterprise Server Model S70 Advanced . The phrase RS/6000 will be substituted for RISC System/6000 for the remainder of this document.

The software used on the cluster of RS/6000 Enterprise Server S70 Advanced includes AIX Version 4.3.2 operating system, Oracle8 Server database manager, and TXSeries 4.2 for AIX transaction manager.

IBM RISC System/6000 Enterprise SMP Server S70 Advanced

Company Name	System Name	Data Base Software	Operating System Software
IBM Corporation Oracle Corporation	RS/6000 Enterprise Server S70 Advanced	Oracle8 Enterprise Edition 8.0.5 w/ OPS & Partition Options	AIX Version 4.3.2

Total System Cost	TPC-C Throughput	Price/Performance
-Hardware -Software -5 Years Maintenance	Sustained maximum throughput of system running TPC-C expressed i transactions per minute	Total system cost/tpmC
$13,521,883	**110,434.10 tpm-C**	**$122.44 per tpm-C**

Preface

TPC Benchmark™ C Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on August 13, 1992 and updated with revision 3.4 on June 25, 1998.

This is the full disclosure report for benchmark testing of a cluster of IBM RS/6000 Enterprise server S70 Advanced according to the TPC Benchmark™ C Standard Specification.

TPC Benchmark™ C exercises the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing a mixture of read-only and update intensive transactions. This is a complex OLTP application environment exercising a breadth of system components associated by such environments characterized by:

- The simultaneous execution of multiple transaction types that span a breadth of complexity
- On-line and deferred transaction execution modes
- Multiple on-line terminal sessions
- Moderate system and application execution time
- Significant disk input/output
- Transaction integrity (ACID properties)
- Non-uniform distribution of data access through primary and secondary keys
- Data bases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Contention on data access and update

This benchmark defines four on-line transactions and one deferred transaction, intended to emulate functions that are common to many OLTP applications. However, this benchmark does not reflect the entire range of OLTP requirements. The extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-C should not be used as a substitute for a specific customer application benchmarks when critical capacity planning and/or product evaluation decisions are contemplated.

The performance metric reported by TPC-C is a “business throughput” measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of processing an order, and each transaction is subject to a response time constraint. The performance metric for this benchmark is expressed in transactions-per-minute-C (tpmC). To be compliant with the TPC-C standard, all references to tpmC results must include the tpmC rate, the associated price-per-tpmC, and the availability date of the priced configuration.

1. General Items

1.1 Application Code Disclosure

The application program (as defined in Clause 2.1.7) must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input and output functions.

Appendix A contains the RS/6000 application code for the five TPC Benchmark™ C transactions. Appendix D contains the terminal functions and layouts.

1.2 Benchmark Sponsor

A statement identifying the benchmark sponsor(s) and other participating companies must be provided.

This benchmark was sponsored by **International Business Machines Corporation**, **Transarc Corporation** (wholly owned subsidiary of IBM Corp.) and **Oracle Corporation.**

1.3 Parameter Settings

Settings must be provided for all customer-tunable parameters and options which have been changed from the defaults found in actual products, including but not limited to:

- *Data Base tuning options*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and application configuration parameters.*

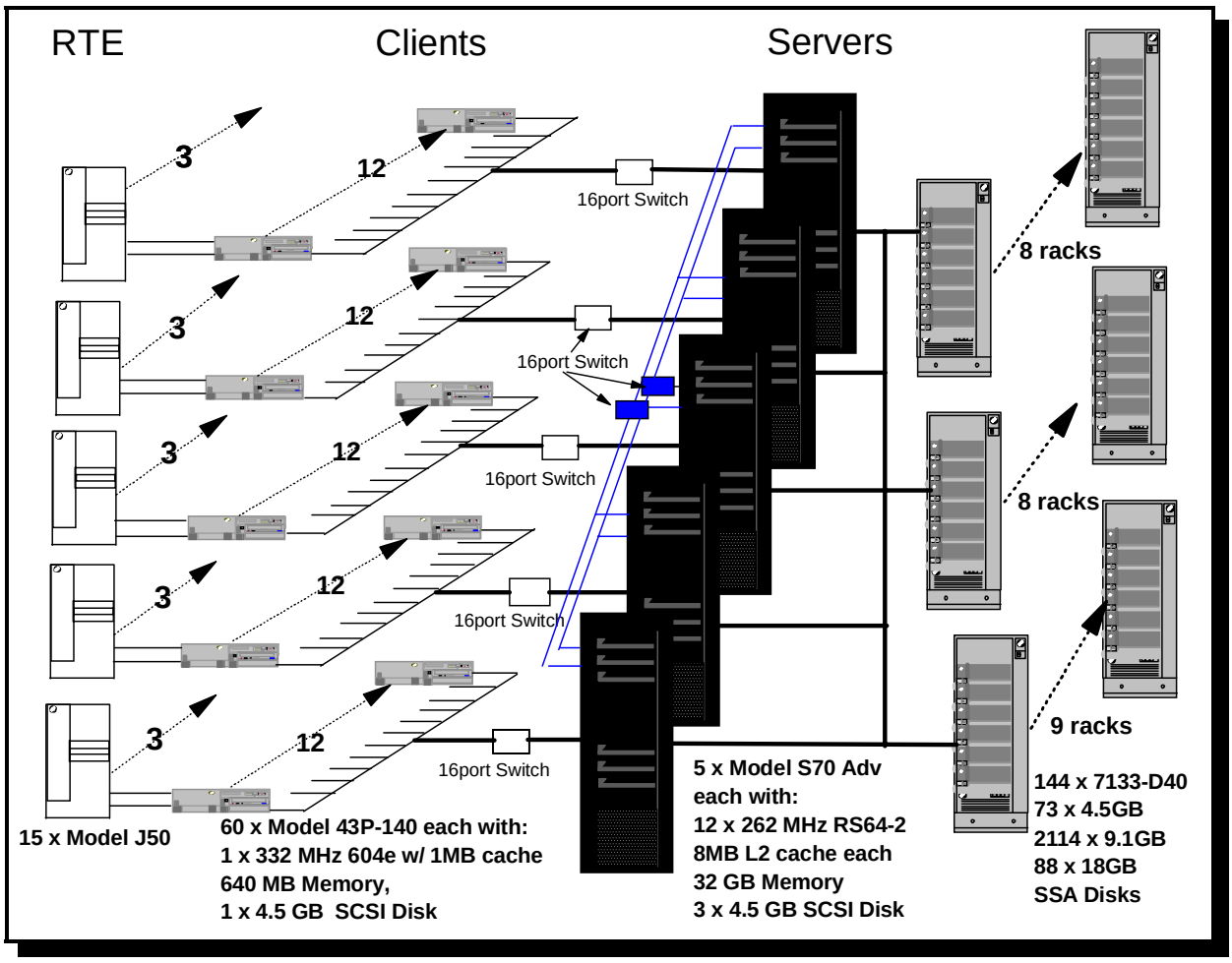
Appendix B contains the system, data base, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

1.4 Configuration Diagrams

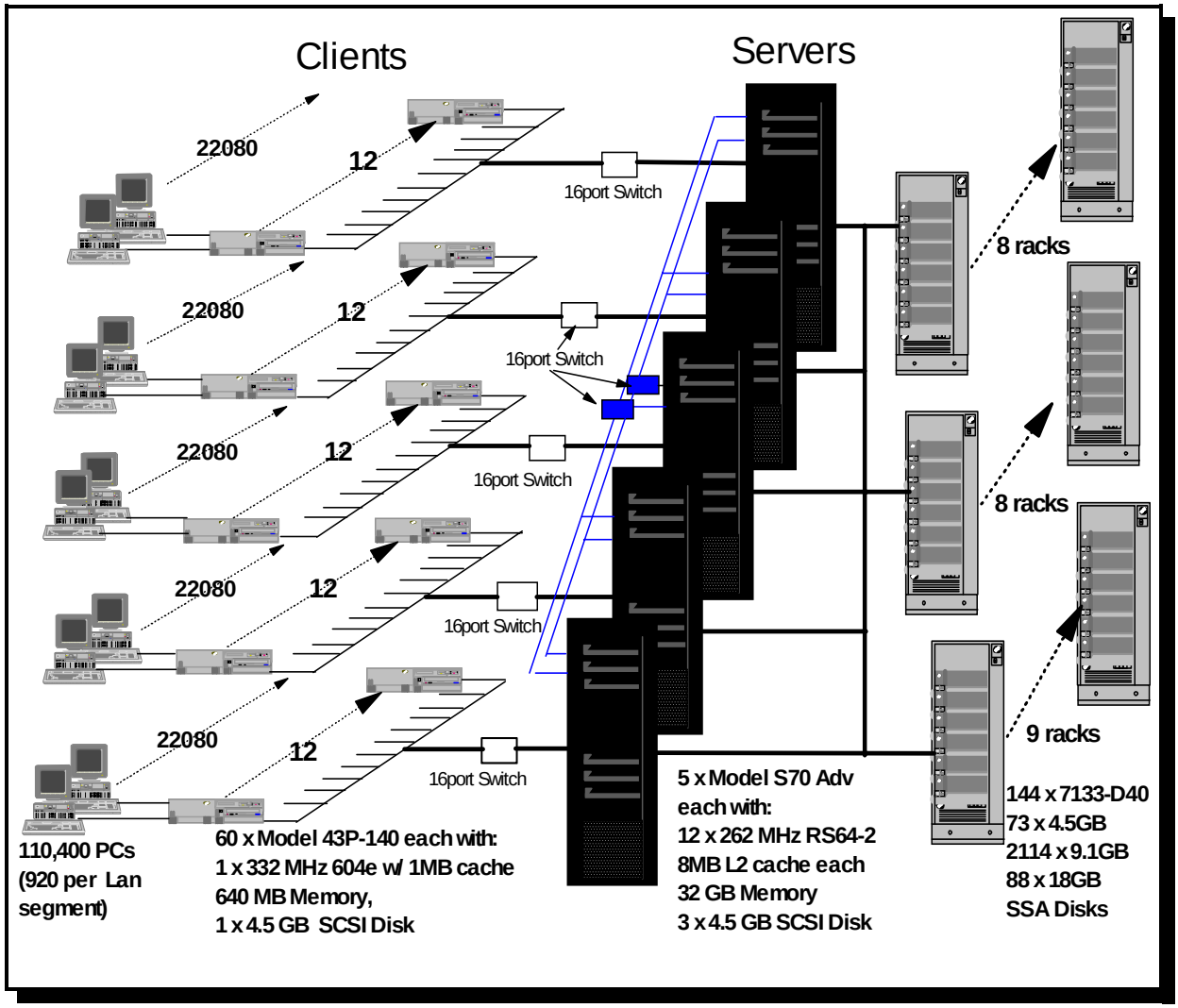
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *Number and type of processors*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test*
- *Number and type of disk units (and controllers, if applicable)*
- *Number of channels or bus connections to disk units, including the protocol type*
- *Number of LAN (e.g. Ethernet) connections, including routers, work stations, terminals, etc, that were physically used in the test or are incorporated into the pricing structure (see Clause 8.1.8)*
- *Type and run-time execution location of software components (e.g. DBMS, client processes, transaction monitors, software drivers, etc)*

RISC System/6000 Enterprise Server S70 Advanced Cluster Benchmark Configuration



RISC System/6000 Enterprise Server S70 Advanced Cluster Priced Configuration



2. Clause 1: Logical Data Base Design Related Items

2.1 Table Definitions

Listings must be provided for all table definition statements and all other statements used to setup the data base.

Appendix C contains the table definitions and the database load programs used to build the data base.

2.2 Database Organization

The physical organization of tables and indices, within the data base, must be disclosed.

Physical space was allocated to Oracle8 Server on the server disks according to the details provided in section C.1 in Appendix C. The size of the space segments on each disk was calculated to provide even distribution of data across the disk subsystem. Clustered indices were defined on all tables except the history table. In addition, a non-clustered index was defined on the Customer table. The indices were defined at table definition and were built at the initial table load by executing the database build script in section C.2 of Appendix C.

2.3 Insert and/or Delete Operations

It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C transaction mix. Furthermore, any restriction in the SUT data base implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows.

There were no restrictions on insert and/or delete operations to any of the tables. The space required for an additional five percent of the initial table cardinality was allocated to Oracle8 Server and priced as static space.

2.4 Horizontal or Vertical Partitioning

While there are few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark, any such partitioning must be disclosed.

Horizontal partitioning was used for the History table, the New Order table, the Order Line table and the Order table along with their respective indexes using the functionality provided by Oracle8.

3. Clause 2: Transaction and Terminal Profiles Related Items

3.1 Verification for the Random Number Generator

The method of verification for the random number generation must be disclosed.

The srandom(), getpid() and gettimeofday() functions are used to produce unique random seeds for each driver. The drivers use these seeds to seed the srand(), srandom() and srand48() functions. Random numbers are produced using wrappers around the standard system random number generators.

The negative exponential distribution uses the following function to generate the distribution. This function has the property of producing a negative exponential curve with a specified average and a maximum value 4 times the average.

```
        const double RANDOM_4_Z = 0.89837799236185
const double RANDOM_4_K = 0.97249842407114

        double neg_exp_4(double average {
                return - average * (1/RANDOM_4_Z * log (1 - RANDOM_4_K * drand48())));
}
```

The random functions used by the driver system and the data base generation program were verified. The C_LAST column was queried to verify the random values produced by the database generation program. After a measurement, the HISTORY, ORDER, and ORDER_LINE tables were queried to verify the randomness of values generated by the driver. The rows were counted and grouped by customer and item numbers.

Here is an example of one SQL query used to verify the random number generation functions:

- create table TEMP (W_ID int, D_ID, C_LAST char(16), CNTR int);
- insert into TEMP select C_W_ID, C_D_ID, C_LAST, COUNT(*) from CUSTOMER group by C_W_ID, C_D_ID, C_LAST;
- select CNTR, COUNT(*) from TEMP group by CNTR order by 1;

3.2 Input/Output Screens

The actual layouts of the terminal input/output screens must be disclosed.

The screen layouts corresponds exactly to the layout corresponding in clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3 and 2.8.3 of the TPC-C specifications.

3.3 Priced Terminal Features

The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained. Although not specifically priced, the type and model of the terminals used for the demonstration in 8.1.3.3 must be disclosed and commercially available (including supporting software and maintenance).

The emulated workstations, IBM RS/6000 Model 43P-140s, are commercially available and support all of the requirements in Clause 2.2.2.4.

3.4 Presentation Managers

Any usage of presentation managers or intelligent terminals must be explained.

The RS/6000 Model 43P-140 workstations did not involve screen presentations, message bundling or local storage of TPC-C rows. All screen processing was handled by the client system. All data manipulation was handled by the server system.

3.5 Home and Remote Order-lines

The percentage of home and remote order-lines in the New-Order transactions must be disclosed.

Table 3-1 show the percentage of home and remote transactions that occurred during the measurement period for the New-Order transactions.

3.6 New-Order Rollback Transactions

The percentage of New-Order transactions that were rolled back as a result of an illegal item number must be disclosed.

Table 3-1 show the percentage of New-Order transactions that were rolled back due to an illegal item being entered.

3.7 Number of Items per Order

The number of items per order entered by New-Order transactions must be disclosed.

Table 3-1 show the average number of items ordered per New-Order transaction.

3.8 Home and Remote Payment Transactions

The percentage of home and remote Payment transactions must be disclosed.

Table 3-1 show the percentage of home and remote transactions that occurred during the measurement period for the Payment transactions.

3.9 Non-Primary Key Transactions

The percentage of Payment and Order-Status transactions that used non-primary key (C_LAST) access to the data base must be disclosed.

Table 3-1 show the percentage of non-primary key accesses to the data base by the Payment and Order-Status transactions.

3.10 Skipped Delivery Transactions

The percentage of Delivery transactions that were skipped as a result of an insufficient number of rows in the NEW-ORDER table must be disclosed.

Table 3-1 show the percentage of Delivery transactions missed due to a shortage of supply of rows in the NEW-ORDER table.

3.11 Mix of Transaction Types

The mix (i.e. percentages) of transaction types seen by the SUT must be disclosed.

Table 3-1 show the mix percentage for each of the transaction types executed by the SUT.

3.12 Queueing Mechanism of Delivery

The queuing mechanism used to defer execution of the Delivery transaction must be disclosed.

The Delivery transaction was submitted using an RPC call to an IBM TXSeries version 4.2, Encina interface transaction manager (TM). TXSeries returns an immediate response to the calling program and schedules the work to be performed. This allows the Delivery transaction to be submitted, obtain an interactive response and queue the actual data base transaction for deferred execution. Please see the application code in Appendix A for details.

Table 3-1 Numerical Quantities for Transaction and Terminal Profiles

New Order	RS/6000 Enterprise Server S70 Advanced
Percentage of Home order lines	99.01%
Percentage of Remote order lines	0.99%
Percentage of Rolled Back Transactions	0.99%
Average Number of Items per order	10
Payment	
Percentage of Home transactions	84.99%
Percentage of Remote transactions	15.01%
Non-Primary Key Access	
Percentage of Payment using C_LAST	60.02%
Percentage of Order-Status using C_LAST	60.01%
Delivery	
Delivery transactions skipped	0
Transaction Mix	
New-Order	44.77%
Payment	43.07%
Order-Status	4.05%
Delivery	4.04%
Stock-Level	4.05%

4. Clause 3: Transaction and System Properties

The results of the ACID test must be disclosed along with a description of how the ACID requirements were met.

All ACID tests were conducted according to specification.

4.1 Atomicity Requirements

The system under test must guarantee that data base transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially-completed operations leave any effects on the data.

4.1.1 Atomicity of Completed Transaction

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number) and verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have been changed appropriately.

The following steps were performed to verify the Atomicity of completed transactions.

1. The balance was retrieved from the CUSTOMER table for a random Customer, District and Warehouse giving BALANCE_1.
2. The Payment transaction was executed for the Customer, District and Warehouse used in step 1
3. The balance was retrieved again for the Customer used in step 1 and step 2 giving BALANCE_2. It was verified that BALANCE_1 was greater than BALANCE_2 by AMT.

4.1.2 Atomicity of Aborted Transactions

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number) and substitute a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have NOT been changed.

The following steps were performed to verify the Atomicity of the aborted Payment transaction:

1. The Payment application code was changed to execute a rollback of the transaction instead of performing the commit.
2. Using the balance, BALANCE_2, from the CUSTOMER table retrieved for the completed transaction, the Payment transaction was executed for the Customer, District, and Warehouse used in step 1 of the section 4.1.1, using a payment amount (AMT) of 410.00. The transaction rolled back due to the change in the application code from step 1.
3. The balance was retrieved again for the Customer used for step 2 giving BALANCE_3. It was verified that BALANCE_2 was equal to BALANCE_3.

4.2 Consistency Requirements

Consistency is the property of the application that requires any execution of a data base transaction to take the data base from one consistent state to another, assuming that the data base is initially in a consistent state.

Verify that the data base is initially consistent by verifying that it meets the consistency conditions defined in Clauses 3.3.2.1 to 3.3.2.4. Describe the steps used to do this in sufficient detail so that the steps are independently repeatable.

4.2.1 Consistency Condition 1

Entries in the WAREHOUSE and DISTRICT tables must satisfy the relationship:

- *W_YTD = sum(D_YTD)*

for each warehouse defined by (W_ID = D_W_ID)

4.2.2 Consistency Condition 2

Entries in the DISTRICT, ORDER, and NEW-ORDER tables must satisfy the relationship:

- *D_NEXT_O_ID - 1 = max(O_ID) = max(NO_O_ID)*

for each district defined by (D_W_ID = O_W_ID = NO_W_ID) and (D_ID = O_D_ID = NO_D_ID). This condition does not apply to the NEW-ORDER table for any districts which have no outstanding new orders.

4.2.3 Consistency Condition 3

Entries in the New-Order table must satisfy the relationship:

- *max(NO_O_ID) - min(NO_O_ID) + 1 = [number of rows in the New-Order table for this district]*

for each district defined by NO_W_ID and NO_D_ID. This condition does not apply to any districts which have no outstanding new orders.

4.2.4 Consistency Condition 4

Entries in the ORDER and ORDER-LINE tables must satisfy the relationship:

- *sum(O_OL_CNT) = [number of rows in the ORDER-LINE table for this district]*

for each district defined by (O_W_ID = OL_W_ID) and (O_D_ID = OL_D_ID).

4.2.5 Consistency Tests

Verify that the data base is initially consistent by verifying that it meets the consistency conditions defined in Clauses 3.3.2.1 to 3.3.2.4. Describe the steps used to do this in sufficient detail so that the steps are independently repeatable.

The consistency conditions defined in 4.2.1 through 4.2.4 were tested using a shell script to issue queries to the database. All queries showed that the data base was in a consistent state.

After executing transactions for the full measurement intervals the shell script was executed again. All queries show that the database was still in a consistent state.

4.3 Isolation Requirements

Operations of concurrent data base transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order.

Because the database is shared among transaction processing nodes, each of the tests validating isolations was performed twice, first with both transactions processing on same node and next with each transaction processing on different node.

4.3.1 Isolation Test 1

This test demonstrates isolation for read-write conflicts of Order-Status and New-Order transactions.

1. An Order status transaction T0 was executed for a randomly selected customer, and the order returned was as recorded. Transaction T0 was committed.
2. A new-order transaction T1 was started for the same customer used in T0. T1 was stopped immediately prior to commit.
3. An order-status transaction T2 was started for the same customer used in T1. Transaction T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 completed and was committed.
5. An order-status transaction T3 was started for the same customer used in T1. T3 returned the order inserted by T1.

This result demonstrates serialization of T2 before T1. It has equivalent validity to the outcome specified in the Standard which supposes T1 to be serialized before T2.

4.3.2 Isolation Test 2

This test demonstrates isolation for read-write conflicts of Order-Status and New-Order transactions when the New-Order transaction is rolled back.

The following steps were performed to satisfy the test of isolation for Order-Status and a rolled back New-Order transactions:

1. An Order status transaction T0 was executed for a randomly selected customer, and the order returned was recorded. Transaction T0 was committed.
2. A new-order transaction T1 with an invalid item was started for the same customer used in T0. Transaction T1 was stopped prior to rollback.
3. An order-status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. Transaction T2 returned the same order that T0 had returned.
4. T1 was rollback.
5. An order-status transaction T3 was started for the same customer used in T1. T3 returned the same order that T0 returned.

4.3.3 Isolation Test 3

This test demonstrates isolation for write-write conflicts of two New-Order transactions.

The following steps were performed to verify isolation of two New-Order transactions:

1. The D_NEXT_O_ID of a randomly selected district was retrieved.
2. A new-order transaction T1 was started for a randomly selected customer within the district used in step1. T1 was stopped immediately prior to commit.
3. Another new-order transaction was started for the same customer used in T1. Transaction T2 waited.
4. T1 completed. T2 completed and was committed.
5. The order number returned by T1 was the same as the D_NEXT_O_ID retrieved in step 1. The order number returned by T2 was one greater that the order number returned by T1.

6. The D_NEXT_O_ID of the same district was retrieved again. It had been incremented by two (it was one greater that the order number returned by T2).

4.3.4 Isolation Test 4

This test demonstrates isolation for write-write conflicts of two New-Order transactions when one transaction is rolled back.

The following steps were performed to verify the isolation of two New-Order transactions after one is rolled back:

1. The D_NEXT_O_ID of a randomly selected district was retrieved.
2. A new-order transaction T1 with an invalid item was started for a randomly selected customer with the district used in step1. T1 was stopped immediately prior to rollback.
3. Another new-order transaction was started for the same customer used in T1. T2 waited.
4. T1 was allowed to rollback. T2 completed and was committed.
5. The order number returned by T2 was the same as the D_NEXT_O_ID retrieved in step 1.
6. The D_NEXT_O_ID of the same district was retrieved again. It had been incremented by one (it was one greater that the order number returned by T2).

4.3.5 Isolation Test 5

This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions.

The following steps were performed to successfully conduct this test:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C_BALANCE of the customer found in step 1 is retrieved.
3. A delivery transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the commit of the database transaction corresponding to the district used in step 1.
4. A payment transaction T2 was started for the same customer found in step 1. T2 waited.
5. T1 was allowed to complete. T2 completed and was committed.
6. The C_BALANCE of the customer found in step 1 was retrieved again. The C_BALANCE reflected the results of both T1 and T2.

4.3.6 Isolation Test 6

This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions when the Delivery transaction is rolled back.

The following steps were performed to successfully conduct this test:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C_BALANCE of the customer found in step 1 is retrieved
3. A delivery transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the rollback of the database transaction corresponding to the district used in step 1.
4. A payment transaction T2 was started for the same customer found in step 1. Transaction T2 waited.
5. T1 was allowed to rollback. T2 completed and was committed.

6. The C_BALANCE of the customer found in step 1 was retrieved again. The C_BALANCE reflected the results of only Transaction T2.

4.3.7 Isolation Test 7

This test demonstrates repeatable reads for the New-Order transaction while an interactive transaction updates the price of an item.

The following steps were performed to successfully conduct this test:

1. The I_PRICE of two randomly selected items were retrieved.
2. A new-order transaction T2 with a group of items X and Y was started. T2 was stopped immediately, after retrieving the prices of all items. The prices of items X and Y retrieved matched those values retrieved in step 1.
3. A transaction T3 was started to increase the price of items X and Y by 10%.
4. T3 did not stall and no transaction was rolled back. T3 was committed.
5. T2 was resumed, and the prices of all items were retrieved again within T2. The prices of items X and Y matched those retrieved in step 1.
6. T2 was committed.
7. The prices of items X and Y were retrieved again. The values matched the values set by T3.

4.3.8 Isolation Test 8

This test demonstrates isolation for phantom protection between a Delivery and a New-Order transaction.

The following steps were performed to successfully conduct this test:

1. The NO_D_ID of all new order rows for a randomly selected warehouse and district was changed. The changes were committed.
2. A delivery transaction T1 was started for the selected customer.
3. T1 was stopped immediately after reading the new order table for the selected warehouse and district . No qualifying rows were found.
4. A new order transaction T2 was started for the same warehouse and district. T2 completed and was committed without being blocked by T1.
5. T1 was resumed and the new order table was read again. No qualifying row was found.
6. T1 completed and was committed.
7. The NO_D_ID of all new order rows for the selected warehouse and district was restored to the original value. The changes were committed.

4.3.9 Isolation Test 9

This test demonstrates isolation for phantom protection between an Order-Status and a New-Order transaction.

The following steps were performed to successfully conduct this test:

1. An order status transaction T1 was started for a randomly selected customer.

2. T1 was stopped immediately after reading the order table for the selected customer The most recent order for that customer was found.
3. A new order transaction T2 was started for the same customer. T2 completed and was committed without being blocked by T1.
4. T1 was resumed and the order table was read again to determine the most recent order for the same customer. The order found was the same as the one found in step 2.
5. T1 completed and was committed.

4.4 Durability Requirements

The tested system must guarantee durability: the ability to preserve the effects of committed transactions and insure data base consistency after recovery from any one of the failures listed in Clause 3.5.3

4.4.1 Permanent Unrecoverable Failure of any Single Durable Medium

Permanent irrecoverable failure of any single durable medium containing TPC-C data base tables or recovery log data.

Failure of Durable Medium containing recovery log data and Instantaneous Interruption and Memory Failure.

This test was conducted on a fully scaled database. The following steps were performed successfully.

1. The current count of the total number of orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving SUM_1.
2. A test was started and allowed to run for twelve minutes.
3. One of the disks containing the Oracle8 transaction log data was powered off. Since the log was on a Oracle mirrored-log disk, Oracle8 continued to process the transactions successfully.
4. The test continued for another 1 1/2 minutes.
5. The Node5 Server of the cluster was immediately shut down by switching the Emergency Power Off , thereby removing system power. The failure of instance on node5 of the cluster was recognized by the system and one of the nodes started an instantaneous recovery.
6. The disk from step 3 was powered back on.
7. Node 3 began recover process and Node 5 remained powered-off.
8. Step 1 is performed returning the value for SUM_2. It was verified that SUM_2 was equal to SUM_1 plus the completed New_Order transactions recorded by the RTE and that no entries existed for rolled-back transactions.
9. Consistency condition 3 was verified.

Failure of Durable Medium containing TPC-C data base tables and failure of inter-node routing switch

The following steps were successfully performed to pass the Durability test of failure of a disk unit with data base tables:

1. The contents of a disk containing a TPCC table was backed up by copying it to another disk.
2. The current count of the total number of orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving SUM_1.
3. A scaled-down test was started and allowed to run until steady state.
4. One of the two Ethernet Switches between the nodes was powered-off.

5. The heartbeat function of HACMP switched to the backup routing on the second switch and processing continued without interruption.
6. Approximately five minutes later, the disk containing the TPCC table was powered off.
7. The run was stopped.
8. The disk from step 4 was powered back on and was restored from the backup copy in step 1.
9. Oracle8 was restarted and its transaction log was used to roll forward through the transactions that had completed since the run had started.
10. Step 2 was performed returning SUM_2. It was verified that SUM_2 was equal to SUM_1 plus the completed New_Order transactions recorded by the RTE and that no entries existed for rolled-back transactions.
11. Consistency condition 3 was verified.

.

5. Clause 4: Scaling and Data Base Population Related Items

5.1 Cardinality of Tables

The cardinality (e.g., the number of rows) of each table, as it existed at the start of the benchmark run, must be disclosed.

Table 5-1 portray the TPC Benchmark™ C defined tables and the number of rows for each table as they were built initially.

Table 5-1 Initial Cardinality of Tables (RS/6000 S70 Advanced)

Table Name	Number of Rows
Warehouse	11,000
District	110,000
Customer	330,000,000
History	330,000,000
Order	330,000,000
New Order	99,000,000
Order Line	3,299,747,564
Stock	1,100,000,000
Item	100,000

5.2 Distribution of Tables and Logs

The distribution of tables and logs across all media must be explicitly depicted for the tested and priced systems.

The following table depicts the data base configuration of the system tested.

Table 5-2. RS/6000 S70 Advanced Data Distribution Benchmark Configuration

Controller	Disk	Contents	Capacity
ssa	hdisk3-23,25-50	log*i1a,log*i2a,log*i3a,log*i4a,log*i5a	18GB
ssa	hdisk71-75	icu1021-025	9GB
ssa	hdisk99-142	cust001-044	9GB
ssa	hdisk143-253	stoc001-111	9GB
ssa	hdisk254-258	icu2001-005	9GB
ssa	hdisk259-260	hist001-004	9GB
ssa	hdisk261-265	nord001-005	9GB
ssa	hdisk266-270	ords001-005	9GB
ssa	hdisk271-290	ordl001-020,ordl201-220	9GB

Controller	Disk	Contents	Capacity
ssa	hdisk291-295	inor001-005	9GB
ssa	hdisk296-300	ior1001-005	9GB
ssa	hdsik301-310	ior2001-010	9GB
ssa	hdisk311-321	iorl001-011	9GB
ssa	hdisk322-323	Item001-002,icu1001-002,istk001-002, root,root1	9GB
ssa	hdisk-324-325	tpcc_cntl1,tpcc_cntl2,rol1	9GB
ssa	hdiska326,2199,328-369	cust045-088	9GB
ssa	hdisk370-480	stoc112-222	9GB
ssa	hdisk481-485	icu2006-10	9GB
ssa	hdisk486-487	hist005-008	9GB
ssa	hdisk488-492	nord006-010	9GB
ssa	hdisk493-497	ords006-010	9GB
ssa	hdisk498-517	ordl021-040,ordl221-240	9GB
ssa	hdisk518-521,2198	inor006-010	9GB
ssa	hdisk523-527	ior1006-010	9GB
ssa	hdsik528-537	ior2011-020	9GB
ssa	hdisk538-548	ior1012-022	9GB
ssa	hdisk549-550	Item003-004,root,icu1003-004, istk003-004	9GB
ssa	hdisk551-552	tpcc_cntl3,root	9GB
ssa	hdiska553-596	cust089-132	9GB
ssa	hdisk597-682,2186,684-707	stoc223-333	9GB
ssa	hdisk708-712	icu2011-015	9GB
ssa	hdisk713-714	hist009-012	9GB
ssa	hdisk715-719	nord011-015	9GB
ssa	hdisk720-724	ords011-015	9GB
ssa	hdisk725-733,2197,735-744	ordl041-060,ordl241-260	9GB
ssa	hdisk745-749	inor011-015	9GB
ssa	hdisk750-754	ior1011-015	9GB
ssa	hdsik755-765	ior2021-030	9GB
ssa	hdisk765-775	ior1023-033	9GB
ssa	hdisk776-777	Item005-006,root,icu1005-006, istk005-006	9GB

Controller	Disk	Contents	Capacity
ssa	hdisk778-779	rol2	9GB
ssa	hdisk780-823-	cust133-176	9GB
ssa	hdisk824-842,2202,844-934	stoc334-444	9GB
ssa	hdisk935-939	icu2016-020	9GB
ssa	hdisk940-941	hist013-016	9GB
ssa	hdisk942-946	nord016-020	9GB
ssa	hdisk947-951	ords016-020	9GB
ssa	hdisk952-958,2191,960-971	ordl061-080,ordl261-280	9GB
ssa	hdisk972-976	inor016-020	9GB
ssa	hdisk977-981	ior1016-020	9GB
ssa	hdsik982-991	ior2031-040	9GB
ssa	hdisk992-1002	ior1034-044	9GB
ssa	hdisk1003-1004	Item007-008,root,icu1007-008, istk007-008	9GB
ssa	hdisk1005-1006	rol2	9GB
ssa	hdisk1007-1050	cus177-220	9GB
ssa	hdisk1051-1106,2187,1108-1161	stoc445-555	9GB
ssa	hdisk1162-1166	icu2021-025	9GB
ssa	hdisk1167-1168	hist017-020	9GB
ssa	hdisk1169-1173	nord021-025	9GB
ssa	hdisk1174-1178	ords021-025	9GB
ssa	hdisk1179-1185,2195,1187-1195 ,2192,1197,1198	ordl081-100,ordl281-300	9GB
ssa	hdisk1199-1203	inor021-025	9GB
ssa	hdisk1204-1208	ior1021-025	9GB
ssa	hdsik1209-1218	ior2041-050	9GB
ssa	hdisk1219-1229	ior1045-055	9GB
ssa	hdisk1230-1231	Item009-010,root,icu1009-010, istk009-010	9GB
ssa	hdisk1232-1233	rol3,tpcc_gms	9GB
ssa	hdisk1234-1277	cus221-264	9GB
ssa	hdisk1278-1388	stoc557-666	9GB
ssa	hdisk1389-1393	icu2026-030	9GB

Controller	Disk	Contents	Capacity
ssa	hdisk1394-1395	hist021-024	9GB
ssa	hdisk1396-1400	nord026-030	9GB
ssa	hdisk1401-1405	ords026-030	9GB
ssa	hdisk1406-1425	ordl1101-120, ordl301-320	9GB
ssa	hdisk1426-1430	inor026-230	9GB
ssa	hdisk1431,1432,2200,1434,1435	ior1026-030	9GB
ssa	hdsik1436-1445	ior2051-060	9GB
ssa	hdisk1446-1451,2207,1453-1456	ior1056-066	9GB
ssa	hdisk1457-1458	Item011-012,root,icu1011-012, istk011-012	9GB
ssa	hdisk1459-1460	rol3	9GB
ssa	hdisk1461-1476,2188,1478-1504	cus265-308	9GB
ssa	hdisk1505-1544,2204,1546-1567,2203,1569-1576,2205,2206,1579-1615	stoc667-777	9GB
ssa	hdisk1616-1620	icu2031-035	9GB
ssa	hdisk1621-1622	his025-028	9GB
ssa	hdisk1623-1627	nord031-035	9GB
ssa	hdisk1628-1632	ords031-035	9GB
ssa	hdisk1633-1652	ordl1121-140, ordl321-340	9GB
ssa	hdisk1653-1657	inor031-035	9GB
ssa	hdisk1658-1662	ior1031-035	9GB
ssa	hdsik1663-1672	ior2061-070	9GB
ssa	hdisk1673-1683	ior1067-077	9GB
ssa	hdisk1684-1685	Item013-014,root,item,icu1013-014, istk013-014	9GB
ssa	hdisk1686-1687	rol4	9GB
ssa	hdisk1688-1697,2189,1699-1731	cus309-352	9GB
ssa	hdisk1732-1842	stoc778-888	9GB
ssa	hdisk1843-1847	icu2036-040	9GB
ssa	hdisk1848-1849	hist029-032	9GB
ssa	hdisk1850-1854	nord036-040	9GB
ssa	hdisk1855-1859	ords036-040	9GB

Controller	Disk	Contents	Capacity
ssa	hdisk1860,2193,1862,2194,1864 -1879	ordl141-160, ordl341-360	9GB
ssa	hdisk1880-1884	inor036-040	9GB
ssa	hdisk1885-1889	ior1036-040	9GB
ssa	hdsik1890-1899	ior2071-080	9GB
ssa	hdisk1900-1910	ior1078-088	9GB
ssa	hdisk1911-1912	Item015-016,root,icu1015-016, istk015-016	9GB
ssa	hdisk1913-1914	rol4	9GB
ssa	hdisk1915-1958	cus353-396	9GB
ssa	hdisk1959-2069	stoc889-999	9GB
ssa	hdisk2070-2074	icu2041-045	9GB
ssa	hdisk2075-2076	hist033-036	9GB
ssa	hdisk2077-2081	nord041-045	9GB
ssa	hdisk2082-2086	ords041-045	9GB
ssa	hdisk2087-2106	ordl161-180, ordl361-380	9GB
ssa	hdisk2107-2111	inor041-045	9GB
ssa	hdisk2112-2116	ior1041-045	9GB
ssa	hdsik2117-2126	ior2081-090	9GB
ssa	hdisk2127-2137	iorl089-099	9GB
ssa	hdisk2138-2139	Item017-018,root,icu1017-018, istk017-018	9GB
ssa	hdisk2140-2141	rol5	9GB
ssa	hdisk2142-2185	cus397-440	9GB
ssa	hdisk2297-2301	icu2046-050	4GB
ssa	hdisk2302-2303	hist037-040	4GB
ssa	hdisk2304-2308	nord046-050	4GB
ssa	hdisk2309-2313	ords046-050	4GB
ssa	hdisk2314-2333	ordl181-200	4GB
ssa	hdisk2334-2338	inor046-050	4GB
ssa	hdisk2339-2343	ior1046-050	4GB
ssa	hdsik2344-2353	ior2091-100	4GB
ssa	hdisk2354-2364	iorl100-110	4GB

Controller	Disk	Contents	Capacity
ssa	hdisk2365-2366	Item019-020,root,icu1019-020, istk019-020	4GB
ssa	hdisk2367-2368	rol5	4GB
ssa	hdisk2369-2380	dist001,ware021	
ssa	hdisk2403-2434,2436-2450	Log*i1b,log*i2b,log*i3b,log*i4b,log*i5b	18GB

5.3 Data Base Model Implemented

A statement must be provided that describes the data base model implemented by the DBMS used.

The database manager used for this testing was Oracle8 Enterprise Edition 8.0.5 from Oracle Inc. For these tests, Oracle Parallel Server and Partition Options were used. Oracle8 Enterprise Edition 8.0 is a relational DBMS.

5.4 Partitions/Replications Mapping

The mapping of data base partitions/replications must be explictly described.

IBM did not use replication. Horizontal partitioning was used for all tables and indexes except customer, item, stock, icustomer, icustomer2, iitems and istock. Refer to appendix C for more details.

5.5 180 day space calculations

RS/6000 Enterprise Server S70 Advanced

TPM	110,434.10
Warehouses	11,000

SEGMENT	TYPE	TSPACE	BLOCKS	FIVE_PCT	DAILY_GROW	TOTAL
CUSTOMER	TABLE	CUST	82,510,001	4,125,500.05	0.00	86,635,501.05
DISTRICT	TABLE	WARE	27,915	1,395.75	0.00	29,310.75
HISTORY	TABLE	HIST	5,830,049	0.00	936,489.04	6,766,538.04
ICUSTOMER	INDEX	ICUST1	1,824,287	91,214.35	0.00	1,915,501.35
ICUSTOMER2	INDEX	ICUST2	4,101,504	205,075.20	0.00	4,306,579.20
IDISTRICT	INDEX	WARE	1,280	64.00	0.00	1,344.00
IITEM	INDEX	ITEMS	1,540	77.00	0.00	1,617.00
INEW_ORDERS	INDEX	INORD	601,729	30,086.45	0.00	631,815.45
IORDERS	INDEX	IORD1	1,824,400	91,220.00	0.00	1,915,620.00
IORDERS2	INDEX	IORD2	2,911,933	145,596.65	0.00	3,057,529.65
IORDER_LINE	INDEX	IORDL	21,000,198	1,050,009.90	0.00	22,050,207.90
ISTOCK	INDEX	ISTK	5,486,865	274,343.25	0.00	5,761,208.25
ITEM	TABLE	ITEMS	3,031	151.55	0.00	3,182.55
IWAREHOUSE	INDEX	WARE	515	25.75	0.00	540.75
NEW_ORDER	TABLE	NORD	516,276	25,813.80	0.00	542,089.80
ORDERS	TABLE	ORD	3,531,283	0.00	567,235.00	4,098,518.00
ORDER_LINE	TABLE	ORDL	60,055,120	0.00	9,646,739.09	69,701,859.09
ROLL_SEG	SYS	ROLL	326,400	0.00	0.00	326,400.00
STOCK	TABLE	STOCKS	100,000,001	5,000,000.05	0.00	105,000,001.05
SYSTEM	SYS	SYSTEM	267,008	0.00	0.00	267,008.00
WAREHOUSE	TABLE	WARE	11,000	550.00	0.00	11,550.00
Total			290,832,335	11,041,124	11,150,463	313,023,922

Dynamic space	69,416,452		
Static space	232,457,007		
Free space	11,150,463		
Daily growth	11,150,463		
Daily spread	0	Oracle may be configured such that daily spread is 0	
180-day space (blk.)	2,239,540,370		
Block size (bytes)	4,096		
180-day (GB)	8,543.17		
		new_order	20,988,778
Log block size	512	Redo blocks writt	611,430,894
Log blocks/tpmC	29.13	Number of log blocks used in one tpmC	
8-hour log (GB)	736.33		

Disk Type	Disk Formatted Capa	SUT # of disks	SUT Capacity(GB)	Priced # of disks	Priced Capacity(GB)	Space usage (GB)	
						180-day	8,543.17
4.5	4,288.00	88	368.50	88	368.50	os+paging	62.93
9.1	8,672.00	2,114	17,902.94	2,114	17,902.94		
18	17,376.00	88	1,493.25	88	1,493.25	Total Space	8,606.10
				2,290	19,764.69		

6. Clause 5: Performance Metrics and Response Time Related Items

6.1 Response Times

Ninetieth percentile, maximum and average response times must be reported for all transaction types as well as for the Menu response time.

Table 6-1 list the response times and the ninetieth percentiles for each of the transaction types for the measured system.

6.2 Keying and Think Times

The minimum, the average, and the maximum keying and think times must be reported for each transaction type.

Table 6-1 list the TPC-C keying and think times for the measured system.

Table 6-1. RS/6000 Enterprise Server S70 Advanced Response, Think and Keying Times

Response Times	New Order	Payment	Order Status	Delivery (int./def.)	Stock Level	Menus
90 %	4.4	3.95	3.95	0.19/14.84	11.7	0.01
Average	2.58	2.3	2.31	0.11/23.83	5.63	0
Maximum	315.41	364.41	276.22	3.69/6245.33	1,346.82	0.4
			Think Times			
Minimum	0.01	0.01	0.01	0.01	0.01	N/A
Average	15.31	15.32	13.32	8.32	8.33	N/A
Maximum	153.2	153.2	133.2	83.2	83.2	N/A
			Keying Times			
Minimum	18	3	2	2	2	N/A
Average	18.01	3.01	2.01	2.01	2.01	N/A
Maximum	18.14	3.15	2.14	2.14	2.14	N/A

6.3 Response Time Frequency Distribution

Response time frequency distribution curves must be reported for each transaction type.

Figure 6-3-1. RS/6000 S70 Advanced New-Order Response Time Distribution

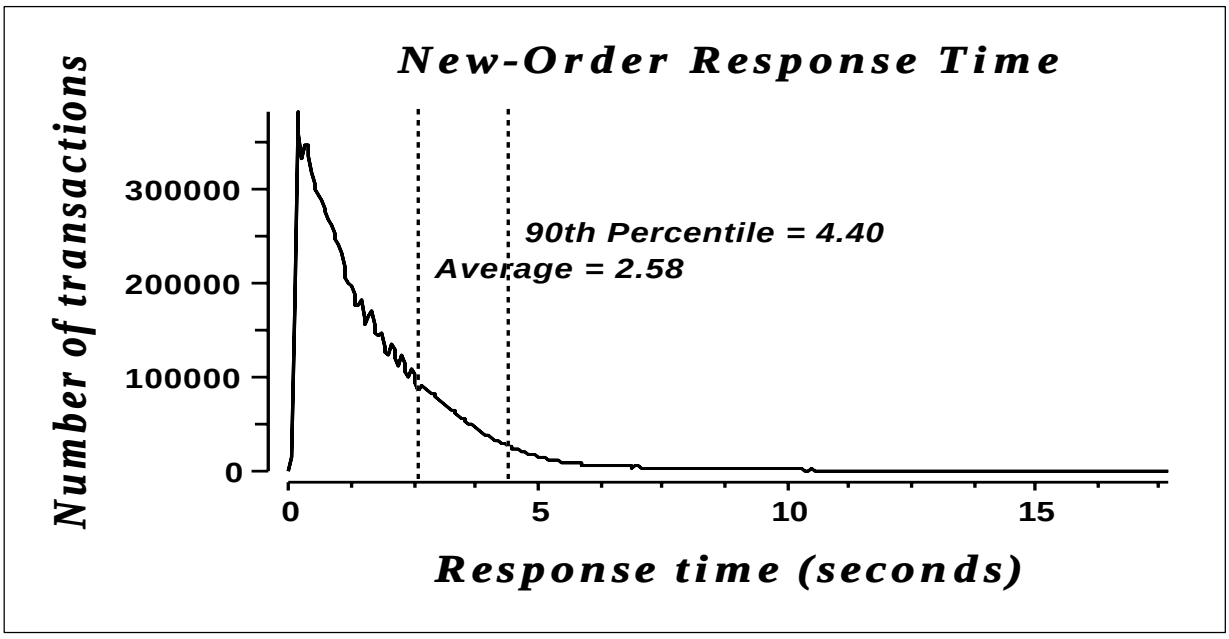


Figure 6-3-2. RS/6000 S70 Advanced Payment Response Time Distribution

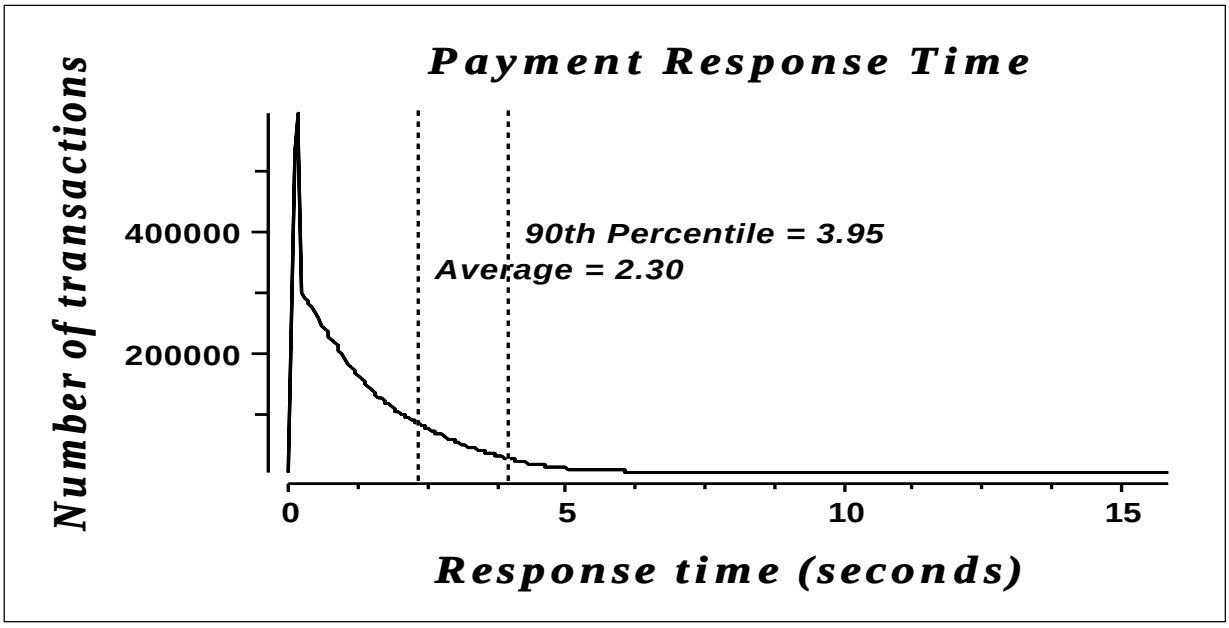


Figure 6-3-3. RS/6000 S70 Advanced Order-Status Response Time Distribution

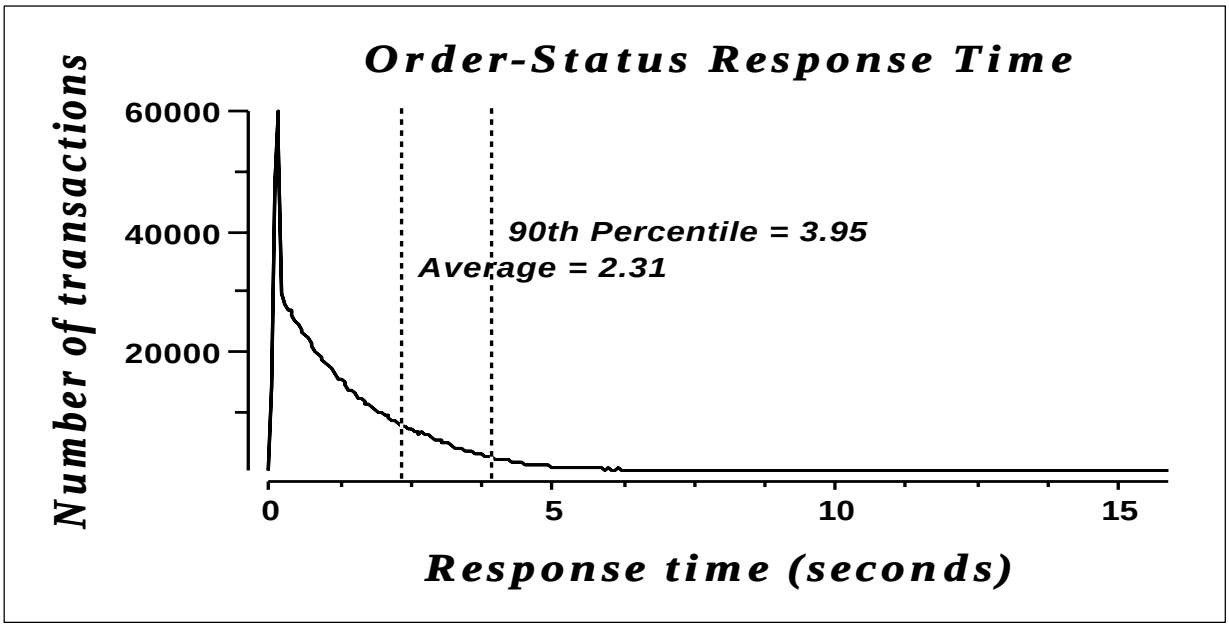


Figure 6-3-4. RS/6000 S70 Advanced Delivery (Interactive) Response Time Distribution

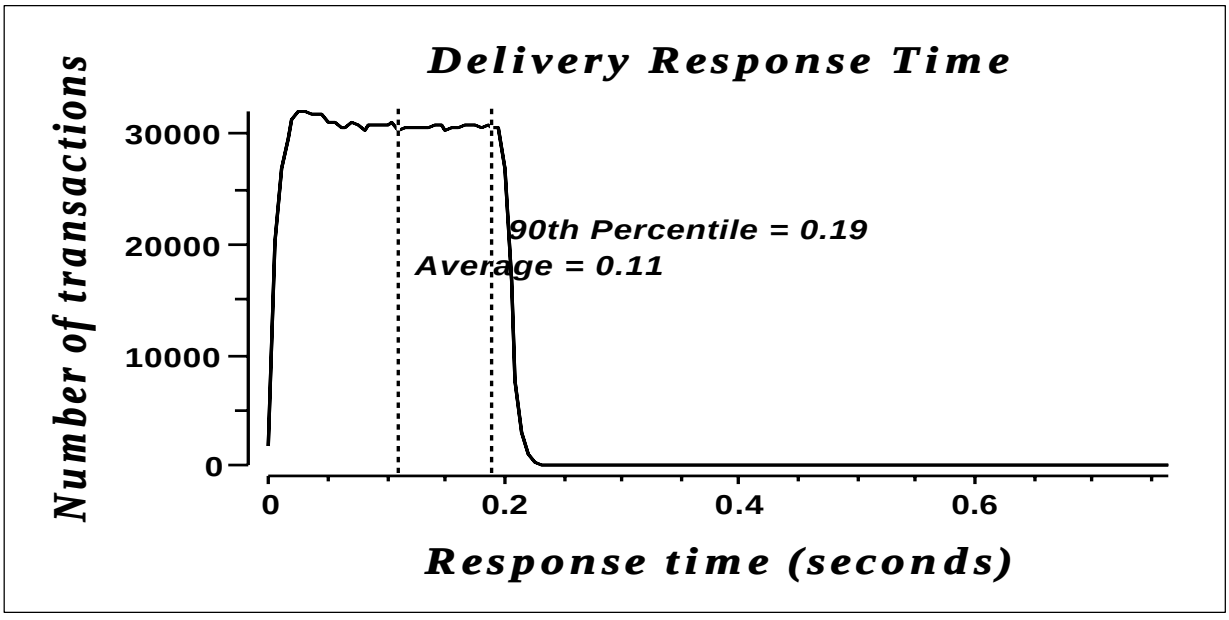


Figure 6-3-5. RS/6000 S70 Advanced Delivery (Deferred) Response Time Distribution

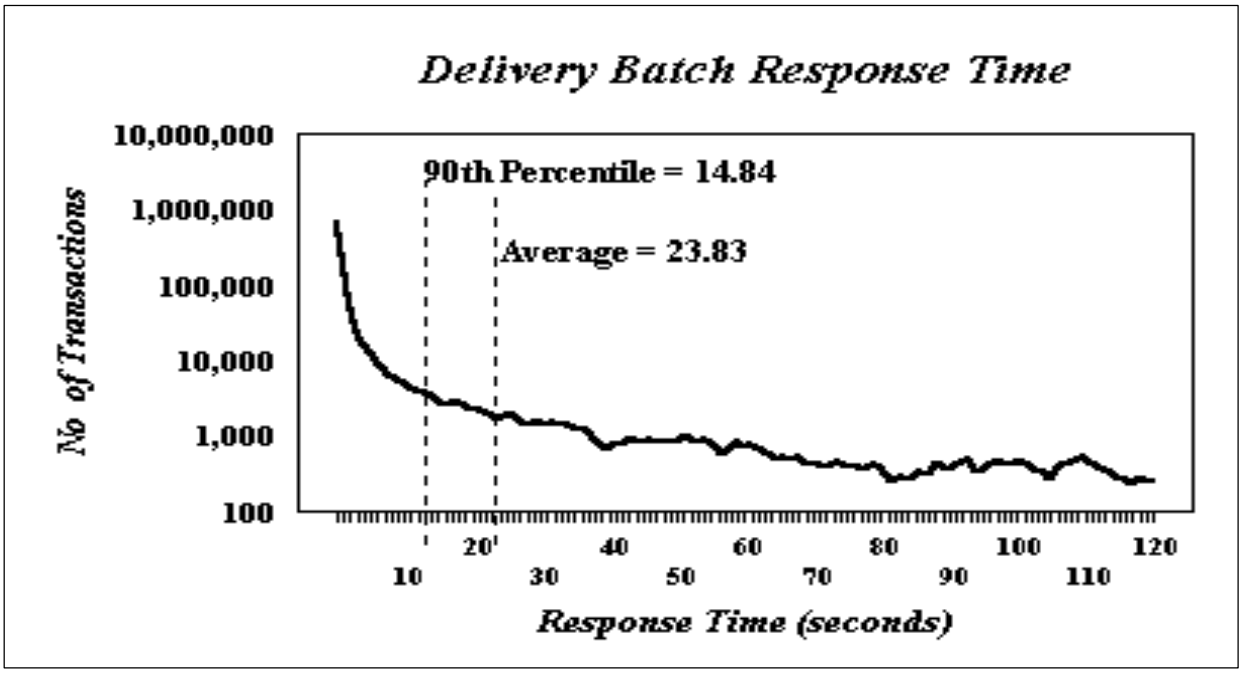
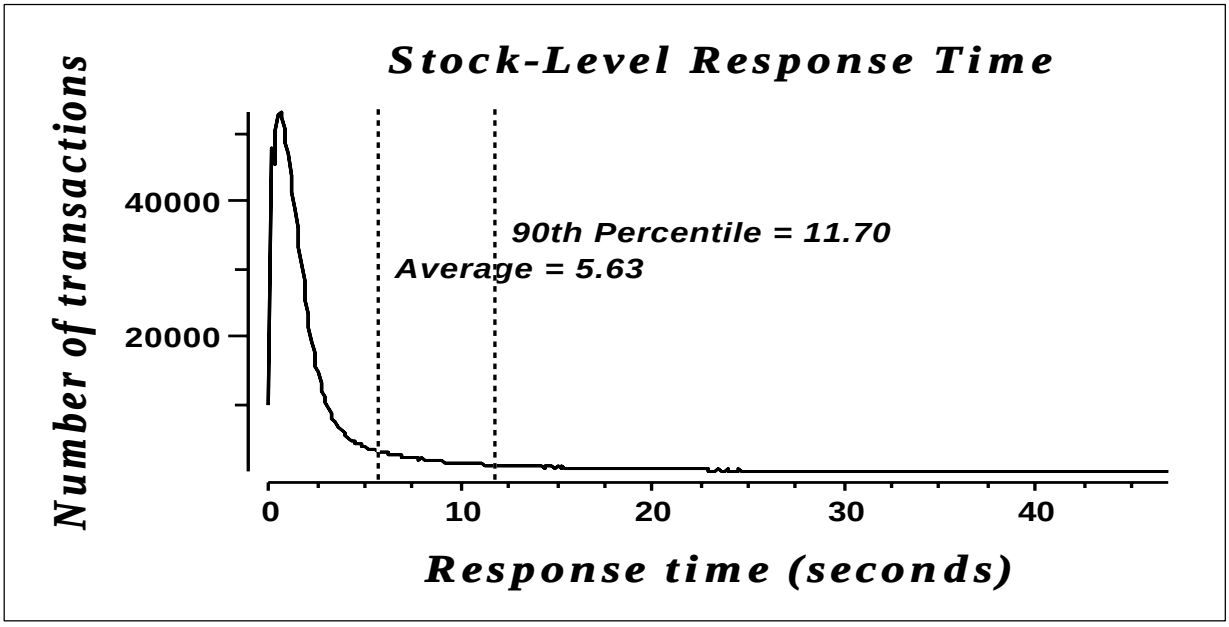


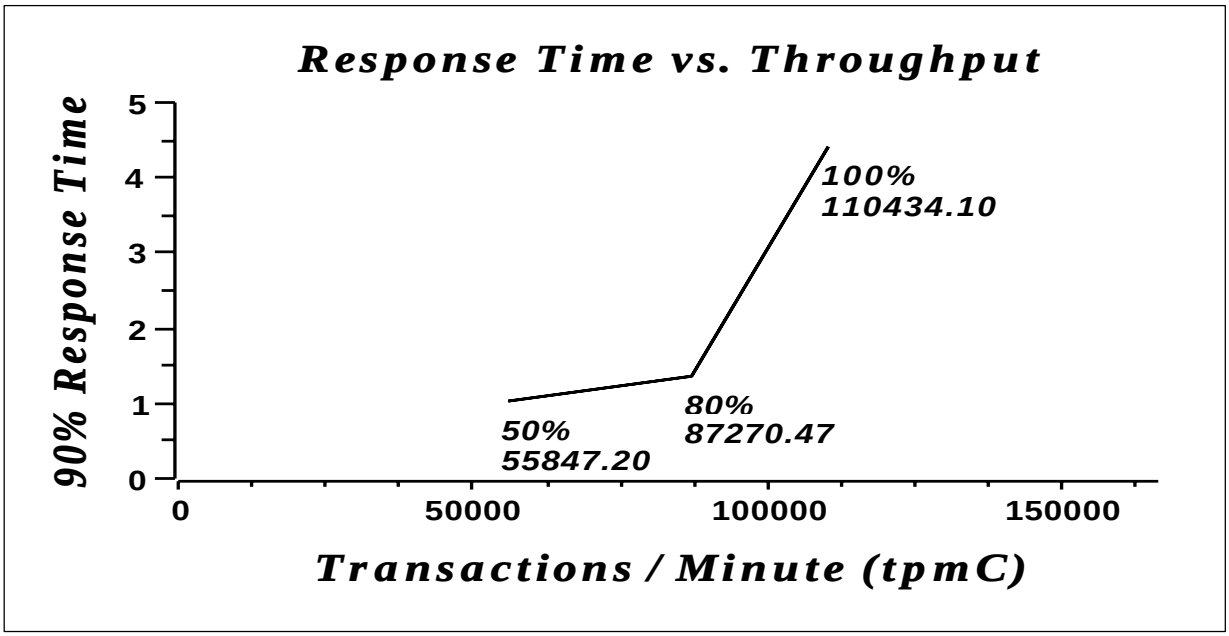
Figure 6-3-6. RS/6000 S70 Advanced Stock Level Response Time Distribution



6.4 Performance Curve for Response Time versus Throughput

The performance curve for response times versus throughput must be reported for the New-Order transaction.

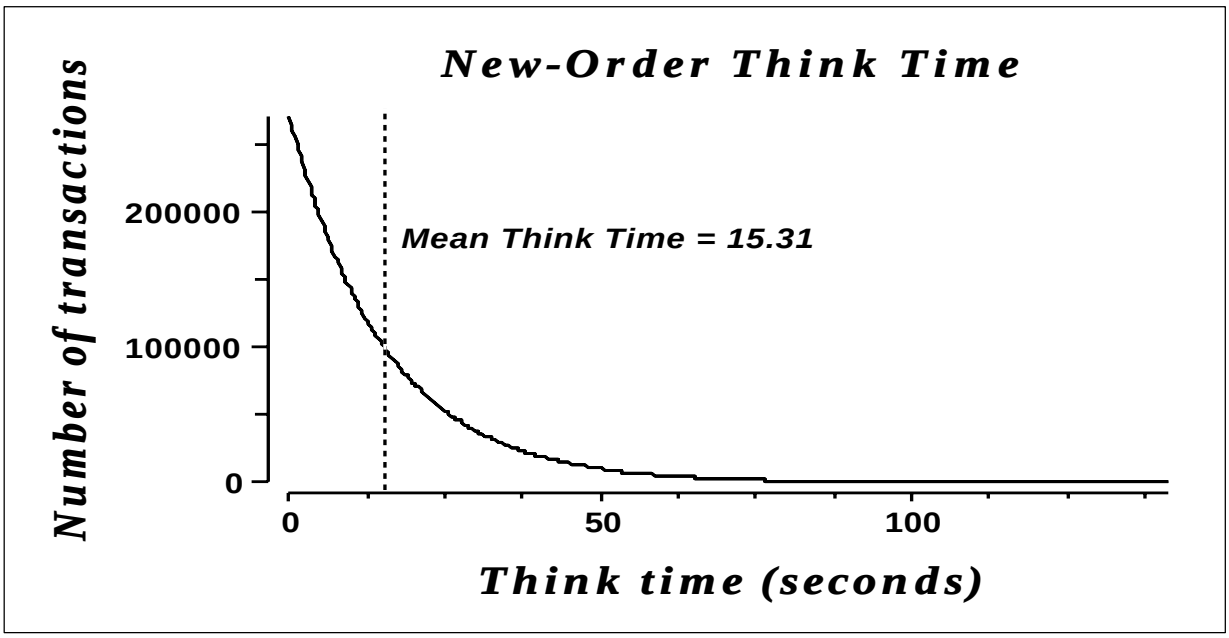
Figure 6-4-1. RS/6000 S70 Advanced New-Order Response Time vs. Throughput



6.5 Think Time Frequency Distribution

A graph of the think time frequency distribution must be reported for the New-Order transaction.

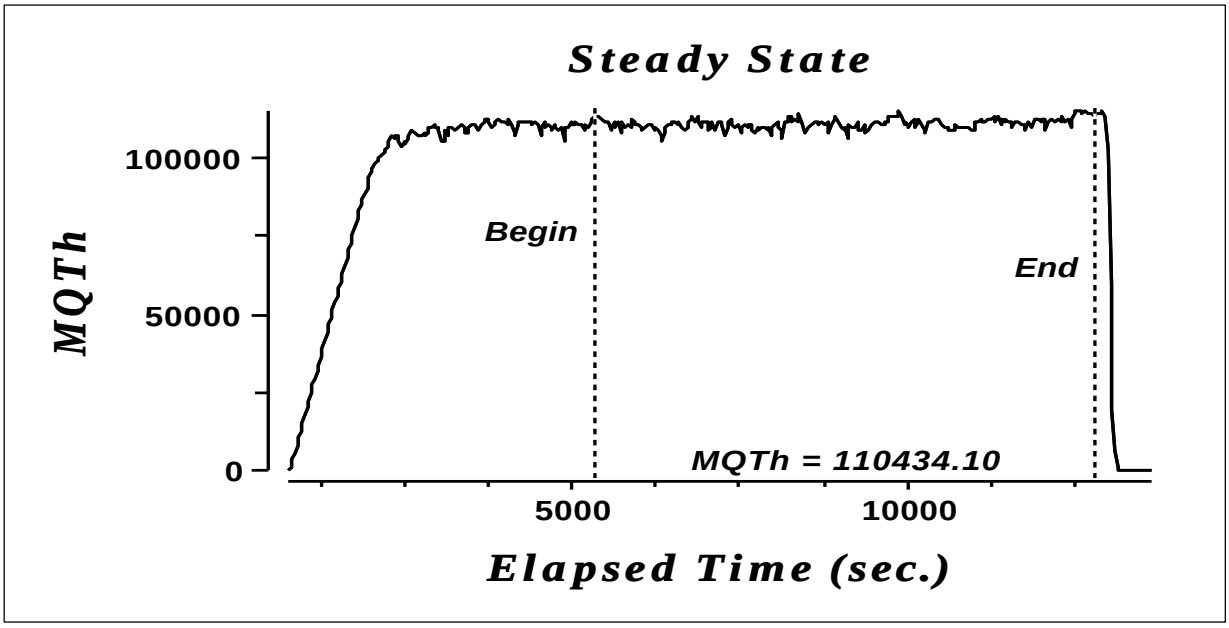
Figure 6-5-1. RS/6000 S70 Advanced New-Order Think Time Distribution



6.6 Throughput versus Elapsed Time

A graph of throughput versus elapsed time must be reported for the New-Order transaction.

Figure 6-6-1. New-Order Throughput vs. Elapsed Time



6.7 Steady State Determination

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval must be described.

All the emulated users were allowed to logon and do transactions. The time stamping interval was set to start after approximately 76 minutes of rampup. Refer to the Numerical Quantities Summary pages for the rampup time. Figure 6.7.1 New-Order throughput versus Elapsed Time graph show that the system was in steady state at the beginning of the Measurement Interval.

6.8 Work Performed During Steady State

A description of how the work normally performed during a sustained test (for example check pointing , writing redo/undo log records, etc), actually occurred during the measurement interval must be reported.

6.8.1 Transaction Flow

For each of the TPC Benchmark™C transaction types, the following steps are executed:

IBM TXSeries version 4.2, Encina interface, was used as a transaction manager (TM). Each transaction was divided into three programs. The front end program handled all screen I/O , a database client program which connected to the database and served as a TXSeries server (a back end program), and a database server program which handled all database operations at the SUT. Both the front end and back end programs ran on the client system. The front end program communicates with the database client program through DCE RPCs. The database client program communicates with the Server system over Ethernet using SQL*Net calls. Besides calling TXSeries Encina initialization code during startup, all other functions are transparent to the application code. Encina routes the transaction and balances the load according to the options defined in the configuration file in appendix B.2, The transaction flow is described below.

- The clients are divided into 10 DCE cells each with a dedicated DCE Server. Each Oracle server has 2 groups of clients each containing 6 clients.

- Each client machine is a node in an Encina Cell.

- There are two types of configurations made for the servers in the Encina nodes, depending on what Oracle server a client machine is connected to. For the Oracle servers Node1, Node2 and Node4, there are two Encina servers configured; one processes the delivery transactions and one processes all of the other transactions. For clients connected to Oracle server Node 3 and Node5, the servers are split up into 3 different types; delivery, stock level and all other transactions.

- These two types of Encina server configurations are set up differently:

- For client machines connected to Oracle server Node1,2,4, the delivery server is configured with 10 server manager DCE threads and 2 background threads to process deferred deliveries. Each background thread has one connection to the database. The Encina server handling the other transactions is set up with 17 processing agents. Each processing agent has one server managing DCE threads. Each thread has one connection to the database.

- For client machines connected to Oracle server Node 3,5, the delivery server is configured with 10 server manager DCE threads and 3 background threads. Each background thread has one connection to the database. The Encina server handling the stocklevel transactions is setup with 7 DCE threads with one background thread each having a connection to the database. The Encina server handling the remaining transactions is setup with 17 background threads each with a database connection.

- When the Encina clients are started, they connect to Encina cell. Each connection has been setup to have a maximum of 4 requests on the queue. The request queue for delivery can handle a maximum of 10,000 pending requests.

- When terminals are started, each terminal connects to the Encina client. The client spawns a thread for each connection to handle that connection. The thread executes the 'process_terminal' routine. The process_terminal displays the TPC-C transaction menu on the user ter minal.

- The TPC-C user chooses the transaction type and proceeds to fill the screen fields required for transaction.

- The process_terminal accepts all values entered by the user and transmits those values to one of the TPC_C backend programs. The transaction is performed through a DCE RPC. There is an interface for each TPC-C transaction type and each TPC-C backend program exports one or more of these interfaces. (The delivery servers export only the delivery interface, the other servers export the other four interfaces, and only those). Encina transparently routes the RPC to one of the servers exporting the corresponding inter face.

- A TPC-C backend server program receives an RPC and proceeds to execute all database operations related to the request. All information entered on the user terminal is contained in the RPC.

- Once the transaction is committed, the server program fills in the outpu t parameters. The RPC is then sent back to the client program.

- When the RPC returns to the client, the process_terminal routine writes the transaction out on the user terminal.

6.8.2 Database Transaction

All database operations are performed by the TPC-C back-end programs. The process is described below:

Using SQL*Net calls, the TPC-C back-end program interacts with Oracle8 Server to perform SQL data manipulations such as update, select, delete and insert, as required by the transaction. After all database operations are performed for a transaction, the transaction is committed.

Oracle8 Server proceeds to update the database as follows:

When Oracle8 Server changes a database table with an update, insert, or delete operation, the change is initially made in memory, not on disk. When there is not enough space in the memory buffer to read in or write additional data pages, Oracle8 Server will make space by flushing some modified pages to disk. Modified pages are also written to disk when a checkpoint occurs. Before a change is made to the database, it is first recorded in the transaction log. This ensures that the database can be recovered completely in the event of a failure. Using the transaction log, transactions that started but did not complete prior to a failure can be undone, and transactions recorded as complete in the transaction log but not yet written to disk can be redone.

6.8.3 Checkpoints

A checkpoint is the process of writing all modified data pages to disk. The TPC-C benchmark was setup to checkpoint every 25 minutes on each node. Two checkpoints were initiated during the rampup period, and four were initiated during the measurement interval for each node.

6.9 Reproducibility

A description of the method used to determine the reproducibility of the measurement results must be reported.

A repeatability measurement was taken for the same length of time as the measured run. The repeatability measurement was 109,768.0 tpmC.

6.10 Measurement Interval

A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC) must be included.

A 125 minute Measurement Interval was used. Further, the measurement interval is a multiple of the checkpoint interval, and since four checkpoints were executed on each node during the measurement interval, the protected zones were not required (as stated in Clause 5.5.2.2). This demonstrates that a different measurement interval over the eight hour period would yield similar throughput results.

7. Clause 6: SUT, Driver, and Communication Definition Related Items

7.1 RTE Availability

If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of what inputs to the RTE had been used.

IBM used an internally developed RTE for these tests. Appendix D contains the scripts used in the testing.

7.2 Functionality and Performance of Emulated Components

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system.

In the benchmark configuration the Remote Terminal Emulator (RTE) communicates with the client system over Ethernet. One RS/6000 Model J50 emulates a network of 7360 RS/6000 Model 43P-140 workstations. The communications mechanism used in the benchmark and priced configurations are the same. In the benchmark configuration two Ethernet LANs for each client were used to connect to one driver system. Each driver system supported four client systems for a total of eight Ethernet LANs. Each LAN segment in the priced configuration is used to connect 920 workstations.

7.3 Network Bandwidth

The bandwidth of the network(s) used in the tested/priced configuration must be disclosed.

The Ethernet used in the LAN complies with the IEEE 802.3 standard and has a bandwidth of 10 Megabits per second. Each LAN segment in the RS/6000 Enterprise Server S70 Advanced configuration connected 920 workstations.

7.4 Operator Intervention

If the configuration requires operator intervention, the mechanism and the frequency of this intervention must be disclosed.

No operator intervention is required to sustain the reported throughput during the eight hour period.

8. Clause 7: Pricing Related Items

8.1 Hardware and Programs Used

A detailed list of the hardware and software used in the priced system must be reported. Each item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package-pricing is used, contents of the package must be disclosed. Pricing source(s) and effective date(s) must also be reported.

The detailed list of all hardware and programs for the priced configuration is listed in the pricing sheets (please refer to Section 8.2 for details) for each system reported. The prices for all products and features that are provided by IBM are available the same day as product or feature availability.

Prices were quoted by Oracle, Inc. based on machine-type category which Oracle has classified as an Enterprise System.

Pricing for 8 port Ethernet Hubs was quoted by Data Comm Warehouse. These have a life-time warranty.

Pricing for the 16 port Ethernet Switches was quoted by NBase-Xyplex. These are warranted for 12 months from receipt of delivery. Extended warrantee is provided at a cost of 3% per year per unit after the 12th month. This cost is included in the 5 year Maintenance. NBase-Xyplex provides 40% discount on these Switches.

8.2 Five Year Cost of System Configuration

The total 5-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

The price sheets for the RS/6000 are contained on the following page. The basis for the discounts used are:

- Extended Maintenance Option (EMO):
  - This is a discount for prepayment of maintenance costs for the system unit, disk, and the terminals. A discount of seventeen percent is available for this configuration based on payment for five years maintenance at time of purchase.
- Mid-Range Service Option (MRSO):
  - This discount is available for customers when agreement is reached for the customer to perform specified service duties (consult marketing representative for details). This discount is applied to the balance after the Extended Maintenance Option Discount is applied. For the TPC Benchmark™C configurations the MRSO discount is seventeen percent for the system, disk and terminals.
- KeyLink, a Pioneer-Standard company, provides complete hardware and software solutions to end-users and offers customers dollar volume discounts based on the total system price (total 5 year system cost, including all hardware, all software and maintenance charges). For this configuration the discount is based on the hardware and software totals; the maintenance is discounted with using EMO and MRSO.

8.3 Availability Dates

The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available.

All products are generally available today.

8.4 Statement of tpmC and Price/Performance

A statement of the measured tpmC, as well as the respective calculations for 5-year pricing, price/performance (price/tpmC), and the availability date must be disclosed.

.

System	tpmC	5-year System Cost	$/tpmC	Availability Date
RS/6000 Enterprise Server S70 Advanced	110,434.1	$13,521,883	$122.44	All HS/SW available as shown in Section 8.3

RS/6000 Enterprise Server S70 Advanced Five Year System Price Configuration

Description	Part Number	Source	Unit Price	Qty	Extended Price	5 yr. Maint Price
Server Hardware						
RS/6000 Server Model S7A	7017-S7A	1	15,036	5	75,180	244,800
3 SCSI Adapters, Rack, CD-ROM, SCSI Hot Swap 6-pack						
4.5 GB 1" Ultra SCSI Hot Swap	2913	1	1,400	15	21,000	0
Async/Terminal Cable	2934	1	45	5	225	0
10/100 Mbs Ethernet PCI Adapter	2968	1	275	20	5,500	0
4 x 2048 MB R1 Memory	4179	1	98,304	20	1,966,080	0
4way RS64 II 262 Mhz Proc, 8MB L2	5312	1	75,000	10	750,000	806,400
4way RS64 II 262 Mhz Proc, 8MB L2	5313	1	75,000	5	375,000	201,600
Advanced SerialRAID Adapter	6225	1	3,000	130	390,000	0
I/O Drawer	6320	1	23,766	20	475,320	0
Primary I/O Drawer Group	6321	1	600	5	3,000	0
Support Processor Group	6322	1	2,700	15	40,500	0
Secondary I/O Drawer Group	6323	1	734	15	11,010	0
I/O Drawer Rack-Rack, Drawer Cabl	3126	1	3,175	5	15,875	0
I/O Rack	7000	1	3,500	10	35,000	0
System Rack Model R00	7015-R00	1	3,110	25	77,750	37,200
Additional Power Distribution Unit	6171	1	1,000	25	25,000	0
SSA Disk Subsystem, Black Cover	7133-D40	1	13,000	144	1,872,000	1,382,400
50/60 Hz AC, 300 VDC Power Supply	8022	1	2,000	144	288,000	0
4.5 GB Disk Drive Modules	8204	1	1,900	73	138,700	0
9.1 GB Disk Drive Modules	8209	1	2,760	2,114	5,834,640	0
18 GB Disk Drive Modules	8218	1	4,500	88	396,000	0
SSA Cables	8801	1	95	576	54,720	0
			Subtotal		12,850,500	2,672,400
Server Software						
AIX 4.3 for S7A	5765-C34	1	300	5	1,500	0
AIX C compiler	5765-C64	1	1,578	1	1,578	0
Performance Toolbox	5765-654	1	1,300	1	1,300	0
HACMP Conc Resource Mgr	5765-A86	1	20,000	5	100,000	0
Oracle Version 8.0.5 OPS		2	1,850,480	1	1,850,480	1,850,480
			Subtotal		1,954,858	1,850,480
Client Hardware						
RS/6000 Model 43P-140, 332 MHz 4.5gb Disk, Intg SCSI-2 FW, Intg Enet, 1MB L2 cache	7043-140	1	7,295	60	437,700	201,600
4.5 GB SCSI Internal Disk	2900	1	1,300	1	1,300	0
128MB DIMM Memory	4102	1	448	60	26,880	0
128MB DIMM Memory Expansion	4115	1	896	240	215,040	0
Async Terminal/Printer Cable	2934	1	45	60	2,700	0
10/100 Mbps Ethernet Adapter, PCI	2986	1	275	180	49,500	0
IBM ASCII Terminal, Keyboard	3153-BG3	1	577	65	37,505	0
			Subtotal		770,625	201,600
Client Software						
AIX 4.3.2 Unlimited Users	5756-C34	1	4,750	60	285,000	0
AIX C compiler	5765-C64	1	1,578	1	1,578	0
IBM TXSeries 4.2 for AIX	5697-D17	1	3,353	60	201,156	70,500
			Subtotal		487,734	70,500
User Connectivity						
16port Ethernet Switch w/40% disc	Nbase	4	1,560	9	14,040	2,808
Ethernet Hub (8port) + 10% Spares	DEH2924	3	36.00	15180	546,480	0
			Subtotal		560,520	2,808
			Discounts		(7,006,041)	(894,101)
				Total	9,618,197	3,903,687

Notes:

Pricing Sources:
1=Keylink, 2=Oracle, 3=Data Comm Warehouse, 4=Nbase/Xyplex

Audited by: Francois Raab , Info Sizing

Five-Year Cost of Ownership: 13,521,883

tpmC 110,434.10

$/tpmC 122.44

9. Clause 9: Audit Related Items

If the benchmark has been independently audited, then the auditor's name, address, phone number, and a brief audit summary report indicating compliance must be included in the Full Disclosure Report. A statement should be included, specifying when the complete audit report will become available and who to contact in order to obtain a copy.

The auditor's attestation letter is included in this section of the report.

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Sponsors:

Jeff Garelick
IBM RS/6000 Performance
11400 Burnet Road
Austin, Texas 78758

Herve Lejeune
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065

June 28, 1999

I verified the TPC Benchmark™ C performance of the following Client Server configuration:

Platform: IBM RS/6000 Enterprise Server S70 Advanced Cluster
Operating system: AIX 4.3.2
Database Manager: Oracle8 Enterprise Edition 8.0.5 Parallel Edition
Transaction Manager: IBM TX Series 4.2 for AIX

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Five node Server: IBM RS/6000 Enterprise Server S70				
12 x RS64-II Per Node (262 MHz)	32 GB per node (8 MB L2 Cache per processor)	88 x 4.5 GB 2114 x 9.1 GB 88 x 18 GB	4.4 Seconds	110,434.10
Sixty Clients: RS6000 Workgroup Server 43P-140 (specification for each)				
1 x PowerPC 604e (332 MHz)	640 MB	(59) with 1 x 4.5 GB (1) with 2 x 4.5 GB	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for Revision 3.4 of the benchmark. The following verification items were given special attention:

- The transactions were correctly implemented
- The database records were the proper size
- The database was properly scaled and populated
- The ACID properties were met
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.

- At least 90% of all delivery transactions met the 80 Second completion time limit
- All 90% response times were under the specified maximums
- The measurement interval was representative of steady state conditions
- The reported measurement interval was 125 minutes
- At least four checkpoints per node were taken during the measurement interval
- Measurement repeatability was verified
- The 180 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

None.

Respectfully Yours,

François Raab
President

1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office: 719/473-7555 • Fax: 719/473-7554

Appendix A: TPC-C Application Source

A.1 Client/Terminal Handler code

Callora.c

```
/*
 * null.c
 *
 * $Revision: 1.4 $
 * $Date: 1998/01/23 15:07:42 $
 * $Log:         server.c,v $
 *
 * $TALog: callora.c,v $
 * Revision 1.4  1998/01/23  15:07:42  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 * Revision 1.1  1997/07/22  21:17:14   radha
 * [added by delta radha-20360-TPCC-integrate-with-Oracle-7322-drivers, r1.1]
 *
 *
 */

#if 1
#define NULL_WITH_SLEEP
#endif

#include <stdio.h>
#include <time.h>
#include <string.h>
#include "serverDebug.h"

#if defined (solaris)
#include <dce/pthread.h>
#else   /* solaris */
#include <pthread.h>
#endif  /* solaris */

#include "tpcc_type.h"
#include "databuf.h"
#include "server.h"

#define ROWS 14
#define COLS 14

#ifndef COMPILE_WITHOUT_ORA
#include "tpcc_info.h"
#endif

#define SIM_ERROR_CODE TPCC_SUCCESS

extern int server_null_test;

#ifdef DEBUG_SERVER
#define PRINT_NEW_IN(a, b) fprintf(stderr, "%s\n", b); print_new_in(a)
#define PRINT_NEW_ORDER(a, b) fprintf(stderr, "%s\n", b); print_new_order(a)
#define PRINT_NEW_RES(rc, a) \
   fprintf(stderr, "<R do_new_order, rc=%d, transtatus=%d, duplicates=%d, all_local=%d\n", \
         rc, (a)->s_transtatus, (a)->s_all_local, (a)->duplicate_items)

#else
#define PRINT_NEW_RES(rc, a)
#define PRINT_NEW_ORDER(a, b)
#define PRINT_NEW_IN(a, b)
#define PRINT_DIST_NEW_ORDER(a, b)
#endif

void mat_mult(int);

float matrix_a[ROWS][COLS]= {
  {1.2, 3.4, 2.3, 4.6, 5.2, 3.5, 4.3, 4.5, 1.8, 2.5, 4.3, 4.5, 1.8, 2.5},
  {2.3, 4.5, 1.2, 9.4, 3.1, 6.5, 1.0, 9.2, 4.5, 2.9, 1.0, 9.2, 4.5, 2.9},
  {3.4, 5.2, 3.8, 6.5, 1.6, 2.3, 4.5, 2.0, 3.4, 3.7, 4.5, 2.0, 3.4, 3.7},
  {1.2, 5.3, 6.1, 2.9, 3.8, 4.6, 2.1, 3.4, 9.0, 7.3, 2.1, 3.4, 9.0, 7.3},
  {2.4, 1.2, 3.4, 7.2, 1.0, 3.2, 3.4, 5.2, 3.8, 7.5, 3.4, 5.2, 3.8, 7.5},
  {2.3, 4.5, 2.1, 3.9, 8.4, 5.2, 3.8, 4.5, 0.2, 9.3, 3.8, 4.5, 0.2, 9.3},
  {4.5, 6.9, 7.2, 1.8, 3.4, 5.1, 3.2, 4.9, 5.2, 3.4, 3.2, 4.9, 5.2, 3.4},
  {7.6, 2.1, 0.9, 3.7, 4.5, 1.0, 3.4, 5.1, 2.3, 4.5, 3.4, 5.1, 2.3, 4.5},
  {9.8, 1.3, 2.0, 6.5, 1.3, 2.5, 4.1, 9.5, 2.3, 4.9, 4.1, 9.5, 2.3, 4.9},
  {2.8, 3.4, 6.5, 0.3, 4.5, 6.7, 2.3, 4.8, 5.2, 1.6, 2.3, 4.8, 5.2, 1.6},
  {4.5, 6.9, 7.2, 1.8, 3.4, 5.1, 3.2, 4.9, 5.2, 3.4, 3.2, 4.9, 5.2, 3.4},
  {7.6, 2.1, 0.9, 3.7, 4.5, 1.0, 3.4, 5.1, 2.3, 4.5, 3.4, 5.1, 2.3, 4.5},
  {9.8, 1.3, 2.0, 6.5, 1.3, 2.5, 4.1, 9.5, 2.3, 4.9, 4.1, 9.5, 2.3, 4.9},
  {2.8, 3.4, 6.5, 0.3, 4.5, 6.7, 2.3, 4.8, 5.2, 1.6, 2.3, 4.8, 5.2, 1.6}
};
float matrix_b[ROWS][COLS]= {
  {3.4, 5.9, 2.8, 3.4, 5.6, 1.3, 4.5, 6.1, 2.4, 3.8, 4.5, 6.1, 2.4, 3.8},
  {7.2, 9.3, 4.6, 5.2, 1.3, 6.4, 1.2, 3.5, 4.1, 2.7, 1.2, 3.5, 4.1, 2.7},
  {6.4, 5.2, 8.3, 9.4, 2.3, 4.5, 2.6, 3.0, 4.8, 5.1, 2.6, 3.0, 4.8, 5.1},
  {7.2, 3.4, 6.9, 8.1, 2.3, 4.6, 2.8, 3.4, 7.5, 3.2, 2.8, 3.4, 7.5, 3.2},
  {2.3, 4.5, 7.2, 3.4, 5.8, 2.3, 9.4, 7.5, 2.9, 3.8, 9.4, 7.5, 2.9, 3.8},
  {4.5, 2.9, 3.4, 5.6, 2.8, 3.4, 5.6, 2.3, 4.5, 3.4, 5.6, 2.3, 4.5, 3.4},
  {9.5, 6.4, 5.6, 7.5, 6.8, 7.3, 9.0, 6.3, 4.7, 5.1, 9.0, 6.3, 4.7, 5.1},
  {2.5, 6.3, 4.6, 5.3, 4.5, 6.3, 4.5, 6.3, 4.5, 8.6, 4.5, 6.3, 4.5, 8.6},
  {3.4, 5.6, 3.7, 4.5, 6.2, 3.4, 5.6, 2.1, 3.4, 5.1, 5.6, 2.1, 3.4, 5.1},
  {2.5, 3.4, 1.2, 3.4, 1.2, 3.4, 2.1, 3.4, 5.2, 4.3, 2.1, 3.4, 5.2, 4.0},
  {9.5, 6.4, 5.6, 7.5, 6.8, 7.3, 9.0, 6.3, 4.7, 5.1, 9.0, 6.3, 4.7, 5.1},
  {2.5, 6.3, 4.6, 5.3, 4.5, 6.3, 4.5, 6.3, 4.5, 8.6, 4.5, 6.3, 4.5, 8.6},
  {3.4, 5.6, 3.7, 4.5, 6.2, 3.4, 5.6, 2.1, 3.4, 5.1, 5.6, 2.1, 3.4, 5.1},
  {2.5, 3.4, 1.2, 3.4, 1.2, 3.4, 2.1, 3.4, 5.2, 4.3, 2.1, 3.4, 5.2, 4.0}
};

static struct timespec *get_wait_time(struct timespec *timeP, int tran)
{
   int ran = random() % 1000;
   int wait;

   if (ran > 998) {
                timeP->tv_sec = 10;
   } else if (ran > 990) {
                timeP->tv_sec = 5;
   } else if (ran > 970) {
      timeP->tv_sec = 1;
   } else {
                timeP->tv_sec = 0;
   }
   timeP->tv_nsec = 50000000;
   if (tran == NEWO_TRANS) {
                timeP->tv_nsec *= 2;
                timeP->tv_sec *= 2;
   }
   return(timeP);
}

void sim_new_order(dataP)
  newOrder_data_t *dataP;
{
  int i;
  extern int num_mults;
  static int next_id = 100;
  struct timespec wait_time;

#ifdef NULL_WITH_SLEEP
  pthread_delay_np(get_wait_time(&wait_time, NEWO_TRANS));
#endif
  mat_mult(num_mults);

  sprintf((char *)dataP->c_last, "BARBARBAR");
  sprintf((char *)dataP->c_credit, "GC");
  dataP->c_discount = 0.33;
  dataP->o_id = next_id++;
  sprintf((char *)dataP->entry_date, "17-12-1995.12:33:56");
  dataP->total = 99.1;
  dataP->w_tax = 0.729;
  dataP->d_tax = 0.15;
  for (i=0; i<dataP->o_ol_cnt; i++) {
    dataP->item[i].price = dataP->item[i].ol_i_id % 1000;
    sprintf((char *)dataP->item[i].name_i, "item %d", i);
    dataP->item[i].s_quantity = i;
    dataP->item[i].brand_generic[0] = i%2 ? 'O' : 'E',
    dataP->item[i].brand_generic[1] = '\0';
    dataP->item[i].ol_amount =
      dataP->item[i].price * dataP->item[i].ol_quantity;
  }

  if ((dataP->item[dataP->o_ol_cnt - 1].ol_i_id < 1) ||
      (dataP->item[dataP->o_ol_cnt - 1].ol_i_id > 100000)) {
    dataP->header.returncode = INVALID_NEWO;
  } else if (random() % 90 == 0) {
    dataP->header.returncode = SIM_ERROR_CODE;
  } else {
    dataP->header.returncode = TPCC_SUCCESS;
  }
  return;
}

void sim_payment(dataP)
    payment_data_t *dataP;
{
  extern int num_mults;
  struct timespec wait_time;

#ifdef NULL_WITH_SLEEP
  pthread_delay_np(get_wait_time(&wait_time, PAYMENT_TRANS));
#endif
  mat_mult(num_mults);

    dataP->c_id = 1;
    dataP->c_credit_lim = 100.9;
    dataP->c_discount = 0.2;
    dataP->c_balance = 11.1;

    sprintf((char *)dataP->c_first, "%-16s", "c_first");
    sprintf((char *)dataP->c_middle, "%-2s", "MI");
    sprintf((char *)dataP->c_last, "%-16s", "c_last");
```

```
  sprintf((char *)dataP->c_street_1, "%-20s", "c_street_1");
  sprintf((char *)dataP->c_street_2, "%-20s", "c_street_2");
  sprintf((char *)dataP->c_city, "%-20s", "c_city");
  sprintf((char *)dataP->c_state, "%-2s", "PA");
  sprintf((char *)dataP->c_zip, "%-9s", "152111111");
  sprintf((char *)dataP->c_phone, "%-16s", "6522573904218222");
  sprintf((char *)dataP->c_date, "%-19s", "28-11-1995");
  sprintf((char *)dataP->c_credit, "%-2s", "GC");
  sprintf((char *)dataP->pay_date, "%-19s", "17-12-1995.12:39:13");
  sprintf((char *)dataP->d_street_1, "%-20s", "d_street_1");
  sprintf((char *)dataP->d_street_2, "%-20s", "d_street_2");
  sprintf((char *)dataP->d_city, "%-20s", "d_city");
  sprintf((char *)dataP->d_state, "%-2s", "PA");
  sprintf((char *)dataP->d_zip, "%-9s", "152111111");
  sprintf((char *)dataP->w_street_1, "%-20s", "w_street_1");
  sprintf((char *)dataP->w_street_2, "%-20s", "w_street_2");
  sprintf((char *)dataP->w_city, "%-20s", "w_city");
  sprintf((char *)dataP->w_state, "%-2s", "OH");
  sprintf((char *)dataP->w_zip, "%-9s", "142411111");

  if (random() % 70 == 0) {
     dataP->header.returncode = SIM_ERROR_CODE;
  } else {
     dataP->header.returncode = TPCC_SUCCESS;
  }
}


void sim_stock_level(dataP)
    stockLevel_data_t *dataP;
{
 extern int num_mults;
 struct timespec wait_time;

#ifdef NULL_WITH_SLEEP
 pthread_delay_np(get_wait_time(&wait_time, STOCK_TRANS));
#endif
 mat_mult(num_mults);

 dataP->stock_count = 12;
 if (random() % 80 == 0) {
  dataP->header.returncode = SIM_ERROR_CODE;
 } else {
  dataP->header.returncode = TPCC_SUCCESS;
 }
}


void sim_delivery(dataP)
    delivery_data_t *dataP;
{
 extern int num_mults;
 struct timespec wait_time;

#ifdef NULL_WITH_SLEEP
 pthread_delay_np(get_wait_time(&wait_time, DELIVERY_TRANS));
#endif

 dataP->start_queue = 2.2;
 dataP->header.returncode = TPCC_SUCCESS;
}


void sim_order_status(dataP)
    orderStatus_data_t *dataP;
{
 extern int num_mults;
 int i;
 struct timespec wait_time;

#ifdef NULL_WITH_SLEEP
 pthread_delay_np(get_wait_time(&wait_time, ORDER_STAT_TRANS));
#endif
 mat_mult(num_mults);


  dataP->c_id = dataP->c_id ? dataP->c_id : 99;
  strcpy((char *)dataP->c_first, "Jerome");
  strcpy((char *)dataP->c_middle, "LB");
  strcpy((char *)dataP->c_last, "Trevoe");
  dataP->c_balance = 90.78;
  dataP->o_id = 99;
  strcpy((char *)dataP->entry_date, "06-12-1995.16:42:28");
  dataP->o_carrier_id = 9;
  dataP->o_ol_cnt = 7;

  for (i=0; i<dataP->o_ol_cnt; i++) {
                dataP->item[i].ol_supply_w_id = 1;
                dataP->item[i].ol_i_id = dataP->w_id * 10 + dataP->d_id;
                dataP->item[i].ol_quantity = 10 * (i+1);
                dataP->item[i].ol_amount = dataP->item[i].ol_quantity * 10.1;
                strcpy((char *)dataP->item[i].delivery_date, "NOT DELIVR");
  }


 if (random() % 90 == 0) {
  dataP->header.returncode = SIM_ERROR_CODE;
 } else {
  dataP->header.returncode = 0;
  }
}

/*
 *  mat_mult
 *              Multiply the above two matrices
 */
void mat_mult(iter)
    int iter;
{
 float res[ROWS][COLS];
 int i, j, k;
 int a_num_rows = ROWS;
 int a_num_columns = COLS;
 int b_num_rows = ROWS;
 int b_num_columns = COLS;

 for (; iter>0; iter--) {
  for (i=0; i<a_num_rows; i++) {
   for (j=0; j<b_num_columns; j++) {
                res[i][j] = 0;
                for (k=0; k<b_num_rows; k++) {
                 res[i][j] += matrix_a[i][k] * matrix_b[k][j];
                }
    matrix_a[i][j] = res[i][0];
   }
  }
  pthread_yield();
 }
}

#define TPCC_RET_SCP(a,b,len) \
    strncpy((char *)dataP->b, (char *)oraStruct.a, len); \
    (char *)dataP->b[(len)-1] = '\0'
#define TPCC_CP(a,b) oraStruct.a = dataP->b
#define TPCC_SCP(a,b,len) strncpy((char *)oraStruct.a, (char *)dataP->b, len)
#define TPCC_RET_CP(a,b) dataP->b = oraStruct.a

#define TPCCP_RET_SCP(a,b,len) \
    strncpy((char *)dataP->b, (char *)oraStructP->a, len); \
    dataP->b[(len)-1] = '\0'
#define TPCCP_CP(a,b) oraStructP->a = dataP->b
#define TPCCP_SCP(a,b,len) strncpy((char *)oraStructP->a, (char *)dataP->b, len)
#define TPCCP_RET_CP(a,b) dataP->b = oraStructP->a

/*
 * Talk to Oracle
 */
#ifndef COMPILE_WITHOUT_ORA

int get_db_ready(dbName, flag, numCn, dvryFileName)
  char *dbName;
  int flag;
  int numCn;
  char *dvryFileName;
{
   int rc;
   AUDITLOG(("> get_db_ready to %s flag %d\n", dbName, flag));
   if (server_null_test) return(0);

   fprintf(stderr, ">> get_db_ready, db: %s, flag %d\n", dbName, flag);

#ifdef SINGLE_THREAD_ONLY
   rc = TPCinit (serverIdNumber, "tpcc", "tpcc");
#else
   rc = TPCinit (serverIdNumber, "tpcc", "tpcc", numCn, dvryFileName);
#endif
   if (rc) {
      fprintf(stderr, "TPCinit(%d, tpcc, tpcc) returned %d\n",
                                serverIdNumber, rc);
   }

   AUDITLOG(("< get_db_ready rc %d\n", rc));
   return(rc);
}

void do_delivery(dataP)
    delivery_data_t *dataP;
{
   struct delstruct oraStruct;
   int rc;

   AUDITLOG(("> do_delivery\n"));

   if (server_null_test) {
      sim_delivery(dataP);
      return;
   }

   TPCC_CP(delin.w_id, w_id);
   TPCC_CP(delin.o_carrier_id, o_carrier_id);
   TPCC_CP(delin.qtime, start_queue);
   TPCC_CP(delin.in_timing_int, queued_time);

   DPRINT(("Calling TPCdel: w_id %d, o_carrier_id %d, %f qtime, %d in_timing_int\n",
                oraStruct.delin.w_id, oraStruct.delin.o_carrier_id,
                oraStruct.delin.qtime, oraStruct.delin.in_timing_int));

   rc = TPCdel(&oraStruct);
```

```
  if ((rc != 0) && (rc != -666)) {
    err_printf("Error TPCdel: terror %d, rc %d, retry %d, w_id %d, o_carrier_id %d, %fqtime,
%din_timing_int\n",
                oraStruct.delout.terror, rc, oraStruct.delout.retry,
                oraStruct.delin.w_id, oraStruct.delin.o_carrier_id,
                oraStruct.delin.qtime, oraStruct.delin.in_timing_int);
  }
  dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.delout.terror;
  AUDITLOG(("< do_delivery rc %d\n", rc));
}

void copyout_order_status(orderStatus_data_t *dataP,
                                        struct ordstruct *oraStructP)
{
  int i;
  TPCCP_RET_CP(ordout.c_balance, c_balance);
  TPCCP_RET_CP(ordout.o_id, o_id);
  TPCCP_RET_CP(ordout.o_carrier_id, o_carrier_id);
  TPCCP_RET_CP(ordout.o_ol_cnt, o_ol_cnt);
  TPCCP_RET_CP(ordout.c_id, c_id);
#define I_CP(ind, a,b) dataP->item[ind].b = oraStructP->ordout.a[ind]
#define I_SCP(ind, a, b, len) \
 strncpy((char *)dataP->item[ind].b, (char *)oraStructP->ordout.a[ind], len); \
 dataP->item[ind].b[(len) - 1] = '\0'
  for (i=0; i<oraStructP->ordout.o_ol_cnt && i < 15; i++) {
    I_CP(i, ol_amount, ol_amount);
    I_CP(i, ol_i_id, ol_i_id);
    I_CP(i, ol_supply_w_id, ol_supply_w_id);
    I_CP(i, ol_quantity, ol_quantity);
    I_SCP(i, ol_delivery_d, delivery_date, 11);
  }
#undef I_CP
#undef I_SCP
  TPCCP_RET_SCP(ordout.c_first, c_first, 17);
  TPCCP_RET_SCP(ordout.c_middle, c_middle, 3);
  TPCCP_RET_SCP(ordout.c_last, c_last, 17);
  TPCCP_RET_SCP(ordout.o_entry_d, entry_date, 20);
}

void do_order_status(dataP)
   orderStatus_data_t *dataP;
{
  struct ordstruct oraStruct;
  int i, rc;

  AUDITLOG(("> do_order_status\n"));

  if (server_null_test) {
    sim_order_status(dataP);
    return;
  }

  TPCC_CP(ordin.w_id, w_id);
  TPCC_CP(ordin.d_id, d_id);
  TPCC_CP(ordin.c_id, c_id);
  oraStruct.ordin.bylastname = ((dataP->c_id == 0) ? 1 : 0);
  TPCC_SCP(ordin.c_last, c_last, 17);

  DEBUGP(("Calling TPCord: w_id %d, d_id %d, c_id %d, bylastname %d, c_last %s\n",
oraStruct.ordin.w_id, oraStruct.ordin.d_id, oraStruct.ordin.c_id, oraStruct.ordin.bylastname,
oraStruct.ordin.c_last));

  rc = TPCord(&oraStruct);
  if ( rc != 0 ) {
    err_printf("Error TPCord: terror %d, rc %d, retry %d, w_id %d, d_id %d, c_id %d,
bylastname %d, c_last %s\n ",
                oraStruct.ordout.terror, rc, oraStruct.ordout.retry,
                oraStruct.ordin.w_id, oraStruct.ordin.d_id, oraStruct.ordin.c_id,
                oraStruct.ordin.bylastname, oraStruct.ordin.c_last);
  }

  copyout_order_status(dataP, &oraStruct);

  dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.ordout.terror;
  AUDITLOG(("< do_order_stats rc %d\n", dataP->header.returncode));
}

void do_stock_level(dataP)
   stockLevel_data_t *dataP;
{
  struct stostruct oraStruct;
  /* What's this comment?? -- srs: i only did this one to check the links */
  int rc;

  AUDITLOG(("> do_stock_level\n"));

  if (server_null_test) {
    sim_stock_level(dataP);
    return;
  }

  TPCC_CP(stoin.w_id, w_id);
  TPCC_CP(stoin.d_id, d_id);
  TPCC_CP(stoin.threshold, threshold);

  DEBUGP(("Calling TPCsto: w_id %d, d_id %d, threashold %d\n",
                oraStruct.stoin.w_id, oraStruct.stoin.d_id,
                oraStruct.stoin.threshold));

  rc = TPCsto( &oraStruct );
  if ( rc != 0 ) {
    err_printf("Error TPCsto : terror %d, rc %d, retry %d, w_id %d, d_id %d, threashold %d\n",
                oraStruct.stoout.terror, rc, oraStruct.stoout.retry,
                oraStruct.stoin.w_id, oraStruct.stoin.d_id,
                oraStruct.stoin.threshold);
  }

  TPCC_RET_CP(stoout.low_stock, stock_count);

  dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.stoout.terror;

  DEBUGP(("do_stock_lev returning %d\n", dataP->header.returncode));
  AUDITLOG(("< do_stock_level rc %d\n", dataP->header.returncode));

}

void copyin_payment(dataP, oraStructP)
   payment_data_t *dataP;
   struct paystruct *oraStructP;
{
  TPCCP_CP(payin.w_id, w_id);
  TPCCP_CP(payin.d_id, d_id);
  TPCCP_CP(payin.c_w_id, c_w_id);
  TPCCP_CP(payin.c_d_id, c_d_id);
  TPCCP_CP(payin.c_id, c_id);
  oraStructP->payin.bylastname = ((dataP->c_id == 0) ? 1 : 0);
  TPCCP_CP(payin.h_amount, h_amount);
  TPCCP_SCP(payin.c_last, c_last, 17);
}

void copyout_payment(dataP, oraStructP)
   payment_data_t *dataP;
   struct paystruct *oraStructP;
{
  TPCCP_RET_SCP(payout.w_street_1, w_street_1, 21);
  TPCCP_RET_SCP(payout.w_street_2, w_street_2, 21);
  TPCCP_RET_SCP(payout.w_city, w_city, 21);
  TPCCP_RET_SCP(payout.w_state, w_state, 3);
  TPCCP_RET_SCP(payout.w_zip, w_zip, 10);
  TPCCP_RET_SCP(payout.d_street_1, d_street_1, 21);
  TPCCP_RET_SCP(payout.d_street_2, d_street_2, 21);
  TPCCP_RET_SCP(payout.d_city, d_city, 21);
  TPCCP_RET_SCP(payout.d_state, d_state, 3);
  TPCCP_RET_SCP(payout.d_zip, d_zip, 10);
  TPCCP_RET_CP(payout.c_id, c_id);
  TPCCP_RET_SCP(payout.c_first, c_first, 17);
  TPCCP_RET_SCP(payout.c_middle, c_middle, 3);
  TPCCP_RET_SCP(payout.c_last, c_last, 17);
  TPCCP_RET_SCP(payout.c_street_1, c_street_1, 21);
  TPCCP_RET_SCP(payout.c_street_2, c_street_2, 21);
  TPCCP_RET_SCP(payout.c_city, c_city, 21);
  TPCCP_RET_SCP(payout.c_state, c_state, 3);
  TPCCP_RET_SCP(payout.c_zip, c_zip, 10);
  TPCCP_RET_SCP(payout.c_phone, c_phone, 17);
  TPCCP_RET_SCP(payout.c_since, c_date, 11);
  TPCCP_RET_SCP(payout.c_credit, c_credit, 3);
  TPCCP_RET_CP(payout.c_credit_lim, c_credit_lim);
  TPCCP_RET_CP(payout.c_discount, c_discount);
  TPCCP_RET_CP(payout.c_balance, c_balance);
  TPCCP_RET_SCP(payout.c_data, c_data, 201);
  TPCCP_RET_SCP(payout.h_date, pay_date, 20);
  strcpy((char *)dataP->w_name, "W_NAME");
  strcpy((char *)dataP->d_name, "D_NAME");
  /* Ignore c_ytd_payment, c_payment_cnt */
}

void do_payment(dataP, xaFlag)
   payment_data_t *dataP;
   int xaFlag;
{

  struct paystruct oraStruct;
  int firstWh, secondWh;
  int rc;

  AUDITLOG(("> do_payment\n"));
  if (server_null_test) {
    sim_payment(dataP);
    return;
  }

  copyin_payment(dataP, &oraStruct);

#if 0
    err_printf("TPCpay: w_id %d, D_id %d, C_w_id %d, c_id %d, bylastname %d, amount %.2f,
c_last %s (%s)\n",
               oraStruct.payin.w_id,
               oraStruct.payin.d_id,
               oraStruct.payin.c_w_id,
               oraStruct.payin.c_id,
               oraStruct.payin.bylastname,
               oraStruct.payin.h_amount,
               oraStruct.payin.c_last,
               dataP->c_last);
#endif


  rc = TPCpay(&oraStruct);
```

```
#if 0
  err_printf("< TPCpay terror %d, rc %d, retry %d\n",
             oraStruct.payout.terror,rc, oraStruct.payout.retry);
#endif


  dataP->header.num_rms = 1;
  if (rc != 0 ) {
     err_printf("Error TPCpay: terror %d, rc %d, retry %d,w_id %d, D_id %d, C_w_id %d, c_id
%d, bylastname %d, amount %.2f,c_last %s (%s)\n",
             oraStruct.payout.terror,rc, oraStruct.payout.retry,
                                oraStruct.payin.w_id,
             oraStruct.payin.d_id,
             oraStruct.payin.c_w_id,
             oraStruct.payin.c_id,
             oraStruct.payin.bylastname,
             oraStruct.payin.h_amount,
             oraStruct.payin.c_last ? oraStruct.payin.c_last : "-NULL-",
                                (char *)dataP->c_last ? (char *)dataP->c_last : "-NULL-");
  }

  copyout_payment(dataP, &oraStruct);

  dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.payout.terror;
  AUDITLOG(("< do_payment rc %d\n", dataP->header.returncode));
}

static void copyin_new_order(dataP, oraStructP)
    newOrder_data_t *dataP;
    struct newstruct *oraStructP;
{
   int i;

   TPCCP_CP(newin.w_id, w_id);
   TPCCP_CP(newin.d_id, d_id);
   TPCCP_CP(newin.c_id, c_id);

#define NO_I_CP(ind,a,b) oraStructP->a[ind] = dataP->item[ind].b
#define NO_I_SCP(ind,a,b,len) strncpy((char *)oraStructP->a[ind], (char *)dataP->item[ind].b, len)

   /* tpccpl.c loops over 15 items, we do the same */
   for (i=0; i<15; i++) {
      NO_I_CP(i, newin.ol_i_id, ol_i_id);
      NO_I_CP(i, newin.ol_supply_w_id, ol_supply_w_id);
      NO_I_CP(i, newin.ol_quantity, ol_quantity);
#ifdef DEBUG_SERVER
      fprintf(stderr, "NewOrder: Item %d, supplyWh %d (local %d)\n",
            i,
            oraStructP->newin.ol_supply_w_id[i],
            oraStructP->newin.w_id);
#endif
   }
   /* Ignore all_local field, total_items,
    * tpccpl.c doesnt use them
    */
#undef NO_I_CP
#undef NO_I_SCP
}

void copyout_new_order(dataP, oraStructP)
    newOrder_data_t *dataP;
    struct newstruct *oraStructP;
{
   int i;

   TPCCP_RET_CP(newout.o_id, o_id);
   TPCCP_RET_CP(newout.o_ol_cnt, o_ol_cnt);
   TPCCP_RET_SCP(newout.c_last, c_last, 17);
   TPCCP_RET_SCP(newout.c_credit, c_credit, 3);
   TPCCP_RET_CP(newout.c_discount, c_discount);
   TPCCP_RET_CP(newout.w_tax, w_tax);
   TPCCP_RET_CP(newout.d_tax, d_tax);
   TPCCP_RET_SCP(newout.o_entry_d, entry_date, 20);
   TPCCP_RET_CP(newout.total_amount, total);
   TPCCP_RET_SCP(newout.status, statusline, 26);

#define NO_RET_CP(ind,a,b) dataP->item[ind].b = oraStructP->newout.a[ind]
#define NO_RET_SCP(ind,a,b,len) strncpy((char *)dataP->item[ind].b, (char
*)oraStructP->newout.a[ind], len)

   for (i=0; i<oraStructP->newout.o_ol_cnt && i<15; i++) {
      NO_RET_SCP(i, i_name, name_i, 25);
      NO_RET_CP(i, s_quantity, s_quantity);
                dataP->item[i].brand_generic[0] = oraStructP->newout.brand_generic[i];
                dataP->item[i].brand_generic[1] = '\0';
      NO_RET_CP(i, i_price, price);
      NO_RET_CP(i, ol_amount, ol_amount);
                /* Ignore s_idx and s_dist */
   }
   if (oraStructP->newout.status[0] != '\0') {
                DEBUGP(("TPCnew: status -- %s\n", oraStructP->newout.status));
                dataP->items_valid = 0;
   } else {
                dataP->items_valid = 1;
   }

#undef NO_RET_CP
#undef NO_RET_SCP
}

void do_new_order(dataP, xaFlag)
  newOrder_data_t *dataP;
  int xaFlag;
{
   static int num_calls = 0;
   int i;
   struct newstruct oraStruct;
   int rc;

   AUDITLOG(("> do_new_order\n"));

   if (server_null_test) {
      sim_new_order(dataP);
      return;
   }

   /*  Copy the structure into the TPCC structure. */
   copyin_new_order(dataP, &oraStruct);

   DEBUGP(("-> TPCnew %d items to wh %d\n",
                dataP->o_ol_cnt, dataP->w_id));
   dataP->header.num_rms = 1;

#if 0
     err_printf("Error TPCnew : w_id %d, d_id %d, c_id %d, o_ol_cnt %d (out cnt %d)\n",
        oraStruct.newin.w_id, oraStruct.newin.d_id,
        oraStruct.newin.c_id, dataP->o_ol_cnt, oraStruct.newout.o_ol_cnt);
      for (i=0; i<15; i++) {
        err_printf("ol_i_id %d, ol_supply_w_id %d, ol_quantity %d\n",
          oraStruct.newin.ol_i_id[i], oraStruct.newin.ol_supply_w_id[i],
          oraStruct.newin.ol_quantity[i]);
      }
#endif

   rc = TPCnew(&oraStruct);

#if 0
  err_printf("< TPCnew terror %d, rc %d, retry %d\n",
          oraStruct.newout.terror,rc, oraStruct.newout.retry);
#endif

   if (rc != 0) {
      err_printf("Error TPCnew : terror %d, rc %d, retry %d,w_id %d, d_id %d, c_id %d, o_ol_cnt
%d (out cnt %d)\n",
                oraStruct.newout.terror,rc, oraStruct.newout.retry,
                oraStruct.newin.w_id, oraStruct.newin.d_id,
                oraStruct.newin.c_id, dataP->o_ol_cnt, oraStruct.newout.o_ol_cnt);
              for (i=0; i<15; i++) {
                err_printf("ol_i_id %d, ol_supply_w_id %d, ol_quantity %d\n",
                        oraStruct.newin.ol_i_id[i], oraStruct.newin.ol_supply_w_id[i],
                        oraStruct.newin.ol_quantity[i]);
              }
   }
   DEBUGP(("-< TPCnew %d\n", rc));

   /* copy out results */
   copyout_new_order(dataP, &oraStruct);

   if (rc == 0) {
              dataP->header.returncode =
                        dataP->items_valid ? TPCC_SUCCESS : INVALID_NEWO;
#if 0
              if (dataP->items_valid && (++num_calls % 500) == 0) {
                int i;
                err_printf("TPCnew Success: w_id %d, d_id %d, c_id %d, o_ol_cnt %d, Oid
%d\n",
                        oraStruct.newin.w_id, oraStruct.newin.d_id,
                        oraStruct.newin.c_id, oraStruct.newout.o_ol_cnt,
                  oraStruct.newout.o_id);
                for (i=0; i<15 && i<oraStruct.newout.o_ol_cnt; i++) {
                        err_printf("   %2d: i_id %i5d, sw_id %4d, qty %d, price %.2f amt
%.2f\n",
                                i, oraStruct.newin.ol_i_id[i],
                                oraStruct.newin.ol_supply_w_id[i],
                                oraStruct.newin.ol_quantity[i],
                  oraStruct.newout.i_price[i],
                  oraStruct.newout.ol_amount[i]);
                }
              }
#endif
   } else {
              dataP->header.returncode = oraStruct.newout.terror;
   }

   AUDITLOG(("< do_new_order rc %d\n", dataP->header.returncode));
}


#else

void TPCexit()
{
}

void do_delivery(dataP)
    delivery_data_t *dataP;
{
```

```
   sim_delivery(dataP);
}

int get_db_ready(dbName, flag, num_cn, dvryFileName)
   char *dbName;
   int flag;
   int num_cn;
   char *dvryFileName;
{
   return(0);
}

void do_order_status(dataP)
    orderStatus_data_t *dataP;
{
   sim_order_status(dataP);
}

void do_stock_level(dataP)
    stockLevel_data_t *dataP;
{
   sim_stock_level(dataP);
}

void do_payment(dataP, xaFlag)
    payment_data_t *dataP;
    int xaFlag;
{
   sim_payment(dataP);
}

void do_new_order(dataP, xaFlag)
  newOrder_data_t *dataP;
  int xaFlag;
{
   sim_new_order(dataP);
}

int get_warehouses(firstWhP, lastWhP, if_xa)
  long int *firstWhP;
  long int *lastWhP;
  long int if_xa;
{
   *firstWhP = 1;
   *lastWhP = 100;
   return(0);
}


#endif
```

client_bg_thread.c

```
/*
 *                  mon_client.c
 *
 * $Revision: 1.15 $
 * $Date: 1998/07/02 18:28:51 $
 * $Log:            $
 *
 *
 * $TALog: client_bg_thread.c,v $
 * Revision 1.15  1998/07/02  18:28:51   wenjian
 * Change client_status_report to send more information of the
 * client process. These changes are matched with changes in
 * tpcc_monitor.c.
 * [from r1.9 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.5]
 *
 * Revision 1.7  1998/04/29  19:47:40   wenjian
 * - Add client_status_report to comunnicate with tpcc_monitor
 * - Add socket_print_rt_avg
 * [from r1.6 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.1]
 *
 * Revision 1.6  1998/02/17  22:12:40   wenjian
 * [merge of changes from 1.3 to 1.4 into 1.5]
 *
 * Revision 1.4  1998/02/17  16:04:40  oz
 * - Split the login into two parts to allow for speciallogins
 * - If the warehouse ID is 0, this is a special login to
 *   query the client for status
 *
 * - check_threads: Return the number of threads
 * - New function: client_report
 * [from r1.3 by delta oz-21864-TPCC-split-client-login-screen, r1.1]
 *
 * Revision 1.5  1998/02/17  22:06:59   wenjian
 * Add necessary head files for win32
 * [from r1.3 by delta wenjian-21750-TPCC-changes-for-porting-on-NT,r1.1]
 *
 * Revision 1.3  1998/01/29  22:53:34  oz
 * - Use pthread delay instead of sleep
 * [from r1.2 by delta oz-21749-TPCC-use-pthread-delay-for-bg-thread,r1.1]
 *
 * Revision 1.2  1998/01/26  20:37:34  oz
 * - Remove all the code associated with explicit binding
 *
 * - Removed include of mon_client_utils.h
 * [from r1.1 by delta oz-21697-TPCC-remove-explicit-binding-code, r1.1]
 *
 * Revision 1.1  1998/01/26  16:19:22  oz
 * - moved all the code pertaining to the background
 *   thread to its own file and all the data structures
 *   to client_utils.h
 * [added by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file,r1.1]
 *
 *
 */

/*
 *  client_bg_thread
 *                  A file used for debug purposes only.
 *
 *  It implements a background thread that once a minute checks the
 *  state of all the threads and reports the state of the client.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <time.h>
#if defined (solaris)
#include <dce/pthread.h>
#else   /* solaris */
#include <pthread.h>
#endif  /* solaris */
#include <tpm/mon/mon.h>
#include <tpm/adl/adl.h>
#include <utils/trace.h>
#ifdef MULTIPLE_INTERFACE
#include "neworder.h"
#include "payment.h"
#include "stocklevel.h"
#include "orderstatus.h"
#else
#include "tpcc_trans.h"
#endif
#include "delivery.h"
#include "utilities.h"
#include "client_utils.h"
#include "do_tpcc.h"
#include "client.h"
#include "encina_client.h"
#ifdef WIN32
#include "io.h"
#include "tran_stat.h"
#endif

#ifdef WIN32
#define   read(A,B,C)     recv(A,B,C,0)
#define   write(A,B,C)    send(A,B,C,0)
#endif

#if 1
#define PRINT_AV(total, num, str)                   \
{                                                   \
   if ((num) > 0) {                                 \
      fprintf(stderr, " %s %.0f,", str, (double)(total)/(num));     \
   }                                                \
}
#else
#define PRINT_AV(a,b,c)
#endif


static void check_threads(total_tran_count_t *tran_ctP, int *numP, int *numInitP);
static struct timeval *client_last_time(thread_descr_t *descrP);
void getUserSysTime(struct timeval *user_time, struct timeval *sys_time);

/*
 *  client_last_time
 *
 *  Each thread maintains the current state it is in and the time
 *  it entered this state.
 *  This routined returns a pointer to the structure in the thread
 *  data that contains the time corresponding to the threads current
 *  state.
 *  Typical use:
 *                  - Set the state, then call gettime on the pointer
 *                    returned by this function.
 */
static struct timeval *client_last_time(thread_descr_t *descrP)
{
   struct timeval *lastTimeP = &descrP->done;
   switch (descrP->state) {
                  case thread_state_init:          /*  Thread is initializing - no trans yet */
                     lastTimeP = &descrP->init;
                     break;
      case thread_state_called:  /*  Tran type was sent by the RTE  */
         lastTimeP = &descrP->called;
         break;
      case thread_state_returned:  /*  Final screen sent to RTE  */
         lastTimeP = &descrP->returned;
```

```
        break;
      case thread_state_sent:  /*  Sent to server  */
        lastTimeP = &descrP->sent;
        break;
      case thread_state_received:  /*  Received reply from server  */
        lastTimeP = &descrP->received;
        break;
      case thread_state_done:  /*  The thread exited */
        lastTimeP = &descrP->done;
        break;
      default:
                err_printf("client_last_time: bad state: %d\n", descrP->state);
        lastTimeP = &descrP->done;
        break;
    }
    return(lastTimeP);
}

void set_client_debug_state(void *contextP, int state, int tran)
{
    thread_info_t *thread_context = (thread_info_t *)contextP;
    struct timezone tz;
    thread_descr_t *descrP = &thread_context->descr;

    descrP->state = state;

    gettimeofday(client_last_time(descrP), &tz);
    if (state == thread_state_called) descrP->tran = tran;
}



/*  How often to report the state of a thread:
 *  If it is in the thread_state_init phase: report if it has been in
 *  that state for more than 5 minutes.
 *  Report if it takes the terminal more thatn 3 minutes to generate the next
 *  transaction.  Otherwise, report if anything takes longer than 60 seconds.
 */
#define THREAD_STATE_REPORT_DELTA(state) \
   ((state) == thread_state_init ? 300 : \
    (state) == thread_state_returned ? 180 : 60)

static char *thread_state_to_str(int state)
{
    char *ret_val = "-Unknown-";
    switch(state) {
                case thread_state_init:     ret_val = "state_init"; break;
                case thread_state_called:   ret_val = "state_called"; break;
                case thread_state_sent:     ret_val = "state_sent"; break;
                case thread_state_received: ret_val = "state_received"; break;
                case thread_state_done:     ret_val = "state_done"; break;
                case thread_state_returned: ret_val = "state_returned"; break;
    }
    return(ret_val);
}

static void print_rt_avg(total_tran_count_t *curP,
                                                total_tran_count_t *prevP,
                                                int type)
{
    int i;
    static char *names[] = {"0", "no", "pa", "os", "dl", "sl"};
    err_printf("%s RT avg: ", type ? "server" : "client");

    for (i=1; i<=MAX_TRAN_TYPE; i++) {
                int num_trans = curP->tran[i].num - prevP->tran[i].num;
                double rt_diff = curP->tran[i].RT[type] - prevP->tran[i].RT[type];
                PRINT_AV(rt_diff, num_trans, names[i]);
    }
    fprintf(stderr, "\n");
}

/*
 *  A background thread that keeps tabs on the state of all the
 *  threads of the client.  (For Debug)
 */
static void *bg_thread(void *argP)
{
    static struct timespec time_wait = {60, 0};

    total_tran_count_t tran_ct, tran_reported[2];
    int total_newo, total_tran_err;
    struct timeval cur_time;
    struct timezone tz;
    struct timeval time_reported[2];

    gettimeofday(&time_reported[0], &tz);
    time_reported[1] = time_reported[0];

    memset(&tran_reported[0], '\0', 2 * sizeof(tran_reported[0]));

    while (1) {
                double time_diff1, time_diff2;
                double tran_diff1, tran_diff2;
                double err_diff1, err_diff2;

                check_threads(&tran_ct, NULL, NULL);

                total_tran_err = tran_ct.errors;
        total_newo = tran_ct.tran[NEWO_TRANS].num;

                gettimeofday(&cur_time, &tz);
                time_diff1 = time_diff_ms(&cur_time, &time_reported[0]);
                tran_diff1 = total_newo - tran_reported[0].tran[NEWO_TRANS].num;
                err_diff1 = total_tran_err - tran_reported[0].errors;
                time_diff2 = time_diff_ms(&cur_time, &time_reported[1]);
                tran_diff2 = total_newo - tran_reported[1].tran[NEWO_TRANS].num;
                err_diff2 = total_tran_err - tran_reported[1].errors;
                if (total_newo != 0 && tran_diff2 > 0) {
                    err_printf("bg_thread: TPM: %.0f (last %.0f sec), %.0f (last %.0fsec)\n",
                                tran_diff1 / time_diff1 * 60000, time_diff1 / 1000.,
                                tran_diff2 / time_diff2 * 60000, time_diff2 / 1000.);
            /* print av server response time for all transactions */
                    print_rt_avg(&tran_ct, &tran_reported[1], 0);
                    print_rt_avg(&tran_ct, &tran_reported[1], 1);
                }

                if (err_diff2 != 0) {
                    err_printf("bg_thread: errPM %.1f (last %.0f sec)\n",
                                                err_diff2 / time_diff2 * 60000, time_diff2 / 1000.);
                }
                tran_reported[0] = tran_reported[1];
                tran_reported[1] = tran_ct;
                time_reported[0] = time_reported[1];
                time_reported[1] = cur_time;
                pthread_delay_np(&time_wait);
    }
}


/*
 *  client_report
 *
 *  Report a summary of the state of the clients
 *  It is cumulative, since the beginning of time
 */
void client_report(int fileno)
{
    int i;
    int num_threads;
    tran_info_t *curP;
    total_tran_count_t tran_ct;
    char cmd = '\0';
    char buf[128];

    while (cmd != 'q') {
                memset(&tran_ct, 0, sizeof(tran_ct));
                check_threads(&tran_ct, &num_threads,NULL);

                sprintf(buf, "Threads %d, Errs %d\n", num_threads, tran_ct.errors);
                write(fileno, buf, strlen(buf));
                for (i=0, curP=tran_ct.tran; i<=MAX_TRAN_TYPE; i++, curP++) {
                    prefix_sprintf(buf, "Type %d, num %d, errs %d, RT %.3f %.3f\n",
                                i, curP->num, curP->errs, curP->RT[0]/1000,
                                curP->RT[1]/1000);
                    write(fileno, buf, strlen(buf));
                }
                /* ENDMSG is a mark to indicate the end of this message */
                write(fileno, ENDMSG, strlen(ENDMSG));
                read(fileno, buf, 2);
                cmd = buf[0];
    }
}


static void check_threads(total_tran_count_t *tran_ctP, int *num_threadsP, int *num_threadsInitP)
{
    struct timezone tz;
    int num_per_state[NUM_STATES];
    int total_stuck = 0;
    static int init_printed = 0;
    int total_tran_err;
    int num_active = 0;
    struct timeval cur_time;

    MUTEX_LOCK(&init_lock);

    if (info_list && (info_list_len > 0)) {
        int i,j;
                int num_init = 0, num_done = 0;
                for (i=0; i<NUM_STATES; i++) num_per_state[i] = 0;

        memset(tran_ctP, '\0', sizeof(*tran_ctP));
                gettimeofday(&cur_time, &tz);
        for (i=0; i<info_list_len; i++) {
                    struct timeval *client_timeP;
                    int time_diff;
                    thread_descr_t *descrP;
                    int delta;
            if (info_list[i] == NULL || !info_list[i]->initialized) {
                                continue;
                    }
                    if (!info_list[i]->done) num_active++;
                    descrP = &info_list[i]->descr;
                    delta = THREAD_STATE_REPORT_DELTA(descrP->state);
                    client_timeP = client_last_time(descrP);

                    for (j=1; j<=MAX_TRAN_TYPE; j++) {
```

```
                tran_ctP->tran[j].num += info_list[i]->tran[j].num;
                tran_ctP->tran[j].errs += info_list[i]->tran[j].errs;
                tran_ctP->tran[j].RT[0]+= info_list[i]->tran[j].RT[0];
                tran_ctP->tran[j].RT[1]+= info_list[i]->tran[j].RT[1];
                                tran_ctP->errors += info_list[i]->tran[j].errs;
                  }

  /*      gettimeofday(&cur_time, &tz); */
                  time_diff = cur_time.tv_sec - client_timeP->tv_sec;
                  DPRINT(("bg_thread: thread %d (index %d) state %s tran %d for %d sec\n",
              info_list[i]->thread_id, i,
              thread_state_to_str(descrP->state),
              descrP->tran,
              time_diff));
                  if (descrP->state == thread_state_init) {
                                num_init++;
                  } else if (descrP->state == thread_state_done) {
                                num_done++;
                                if (!descrP->done_printed) {
                                   err_printf("bg_thread: thread %d (index %d) done.\n",
                    info_list[i]->thread_id, i);
                                   descrP->done_printed = 1;
                                }
                  } else if (time_diff > delta) {
           num_per_state[descrP->state] ++;
                                total_stuck++;
                                if (!descrP->printed) {
                                   err_printf("bg_thread: thread %d (index %d) state %s tran %d stuck
for %d sec\n",
                     info_list[i]->thread_id, i,
                     thread_state_to_str(descrP->state),
                     descrP->tran,
                     time_diff);

                                   descrP->printed = 1;
                                }
                  } else if (descrP->printed) {
                                err_printf("bg_thread: thread %d (index %d) state %s tran %d
unstuck.\n",
                     info_list[i]->thread_id, i,
                     thread_state_to_str(descrP->state),
                     descrP->tran);

                                descrP->printed = 0;
        }
      }
                if (num_threadsP)
                  *num_threadsP = num_active;
                if (num_threadsInitP)
                  *num_threadsInitP = num_init;

                if (num_init > 0) {
                  err_printf("bg_thread: %d threads still in the init state\n",
                                        num_init);
                } else if (!init_printed) {
                  err_printf("bg_thread: All %d threads are running\n",
                                info_list_len);
                  init_printed = 1;
                }
                if (num_active != info_list_len)
                  err_printf("%d threads of %d are still active\n",
                                num_active, info_list_len);

                if (num_done > 0) {
                  err_printf("bg_thread: %d threads done so far.\n", num_done);
                }
                if (total_stuck > 0) {
                  err_printf("bg_thread: Summary %d stuck: ", total_stuck);
                  for (i=0; i<NUM_STATES; i++) {
                    if (num_per_state[i] > 0) {
                                fprintf(stderr, "%d %s, ",
                                        num_per_state[i], thread_state_to_str(i));
                    }
                  }
                  fprintf(stderr, "\n");
                }
                total_tran_err = 0;
    for (i=0; i<=MAX_TRAN_TYPE; i++)
                total_tran_err += tran_ctP->tran[i].errs;
    if (total_tran_err > 0) {
                err_printf("bg_thread: %d errs: %d no, %d pa, %d os, %d sl\n",
                                total_tran_err,
                                tran_ctP->tran[NEWO_TRANS].errs,
                                tran_ctP->tran[PAYMENT_TRANS].errs,
                                tran_ctP->tran[ORDER_STAT_TRANS].errs,
                                tran_ctP->tran[STOCK_TRANS].errs);
    }
  }
  MUTEX_UNLOCK(&init_lock);
}

void start_bg_debug_thread()
{
  int rc;
  pthread_attr_t attr;
  pthread_t thread;

  if (rc = pthread_attr_create(&attr)){
    err_printf("start_bg_debug_thread: pthread_attr_create failed: %d\n", rc);
                return;
  }
  if ((rc = pthread_create(&thread,
                  attr,
                  bg_thread,
                  (pthread_addr_t)NULL)) != 0) {
    err_printf("start_bg_debug_thread: pthread_create failed: %d\n", rc);
                return;
  }
  if ( rc = pthread_detach(&thread) != 0) {
                err_printf("start_bd_debug_thread: pthread_detach failed %d\n", rc);
                return;
  }
}


/* socket_print_rt_avg:
 * mainly copied from print_rt_avg
 */
static void socket_print_rt_avg(int fileno,
                                                total_tran_count_t *curP,
                total_tran_count_t *prevP,
                int type)
{
  int i;
  static char *names[] = {"0", "no", "pa", "os", "dl", "sl"};
  char buf1[1024], buf2[256];

  prefix_sprintf(buf1, "%s RT avg: ", type ? "server" : "client");

  for (i=1; i<=MAX_TRAN_TYPE; i++) {
    int num_trans = curP->tran[i].num - prevP->tran[i].num;
    double rt_diff = curP->tran[i].RT[type] - prevP->tran[i].RT[type];
                if (num_trans>0)
                  sprintf(buf2," %s %.0f,", names[i], (double)(rt_diff)/(num_trans));
                else
                  sprintf(buf2," %s 0.0,", names[i]);
                strcat(buf1,buf2);
  }
  strcat(buf1, "\n");
  write(fileno, buf1, strlen(buf1));
}

/* client_status_report:
 * mainly copied from bg_thread
 */
void *client_status_report(int fileno)
{
  static struct timespec time_wait = {60, 0};

  total_tran_count_t tran_ct;
  tran_info_t  *curP;
  struct timeval cur_time;
  struct timezone tz;
  char  buf[1024], cmd='\0';
  int   i, cnt=0;

  /* a loop for communication with tpcc_monitor */
  while ( cmd != 'q' ) {
    struct timeval cur_time;
    struct timeval user_time={0,0}, sys_time={0,0};
    struct timezone tz;
                int num_threads, num_threadsInit;

    memset(&tran_ct, 0, sizeof(tran_ct));

      /* read next cmd from the socket */
                read(fileno, buf, 1);
                cmd = buf[0];
                /* DPRINT(("%c\n",cmd)); */
                if (cmd=='q') {
                        break;
                }

                check_threads(&tran_ct, &num_threads, &num_threadsInit);
                gettimeofday(&cur_time, &tz);
#ifdef GET_USER_SYS_TIME
        getUserSysTime(&user_time, &sys_time);
#endif

                /* Protocol:
                 *        thread id and time stamp
         *                time_diff1 time_diff2 total_newo total_erro
                 *              type num_trans error RT-C RT-S
        */
                prefix_sprintf(buf,"\n");
                write(fileno,buf,strlen(buf));
                sprintf(buf, "%d %d %d %d %d %d\n",
                        cur_time.tv_sec, cur_time.tv_usec,
                        tran_ct.tran[NEWO_TRANS].num,
                        tran_ct.errors, num_threads, num_threadsInit);
                write(fileno, buf, strlen(buf));
                sprintf(buf, "%d %d %d %d\n", user_time.tv_sec,user_time.tv_usec,
                                        sys_time.tv_sec, sys_time.tv_usec);
                write(fileno, buf, strlen(buf));
        for (i=0,curP=tran_ct.tran; i<=MAX_TRAN_TYPE; i++, curP++) {
                                if (i==0) continue;
          sprintf(buf, "%d %d %d %.3f %.3f\n",
            i, curP->num, curP->errs, curP->RT[0],
```

```
            curP->RT[1]);
          write(fileno, buf, strlen(buf));
                }
                write(fileno, ENDMSG, strlen(ENDMSG));

  }
}


/* for AIX only */
void getUserSysTime(struct timeval *user_time, struct timeval *sys_time)
{
  struct rusage rubuff;

#ifdef GET_CLIENT_USAGE
  if (getrusage(RUSAGE_SELF,&rubuff) == 0) {
                  user_time->tv_sec = rubuff.ru_utime.tv_sec;
    user_time->tv_usec = rubuff.ru_utime.tv_usec;

    sys_time->tv_sec = rubuff.ru_stime.tv_sec;
    sys_time->tv_usec = rubuff.ru_stime.tv_usec;

  } else {
    user_time->tv_sec = user_time->tv_usec = 0;
    sys_time->tv_sec = sys_time->tv_usec = 0;
  }
#else
  user_time->tv_sec = user_time->tv_usec = 0;
  sys_time->tv_sec = sys_time->tv_usec = 0;
#endif
}
```

client.C

```
/* (C)1997 IBM Corporation */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

#include "screen.h"
#include "encina.h"

extern "C" void set_client_debug_state(void *contextP, int state, int tran);

Encina encina;

extern "C" int client_login(int infd, int outfd, int *w_idP, int *d_idP)
{
  Thread_data thread(infd, outfd, NULL);
  User_data user_data;
  Login log(&user_data, &thread);
  log.handle();
  *w_idP = user_data.warehouse;
  *d_idP = user_data.district;
  return 0;
}

extern "C" int client_init (int infd, int outfd, int w_id, int d_id,
                                                  void *contextP) {
  int rc = 0;
  Thread_data *threadP = new Thread_data(infd, outfd, contextP);
  Field *menuField = new IntField(threadP, 8);
  User_data user_data;
  Menu  menu(&user_data, threadP);

  user_data.warehouse = w_id;
  user_data.district = d_id;
  menu.present();

  Payment pay(&user_data, threadP);
  Delivery del(&user_data, threadP);
  OrderStatus os(&user_data, threadP);
  StockLevel sl(&user_data, threadP);
  NewOrder no(&user_data, threadP);

  while (rc == 0) {
                int key = menuField->get_key();
                set_client_debug_state(contextP, 1, key - '0');
                switch (key) {
                  case EOF: rc = -1; break;
                  case '1': case 'N': case 'n': rc = no.handle();  break;
                  case '2': case 'P': case 'p': rc = pay.handle(); break;
                  case '3': case 'O': case 'o': rc = os.handle();  break;
                  case '4': case 'D': case 'd': rc = del.handle(); break;
                  case '5': case 'S': case 's': rc = sl.handle();  break;
                  case '\030':
                    position(threadP, 1, 1);
                    threadP->flush(); break;
                  case '9': case 'Q': case 'q': case 'E': case 'e':
                            return(0);
                  default: threadP->write("\a", 1); break;
                }
                set_client_debug_state(contextP, 4, key - '0');
    }
  return 0;
}
```

client.h

```
/*
 *                client.h
 *
 * $Revision: 1.5 $
 * $Date: 1998/01/26 16:19:22 $
 * $Log:              $
 *
 *
 * $TALog: client.h,v $
 * Revision 1.5  1998/01/26  16:19:22  oz
 * - moved all the code pertaining to the background
 *   thread to its own file and all the data structures
 *   to client_utils.h
 * [from r1.4 by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file,r1.1]
 *
 * Revision 1.4  1998/01/23  15:07:43  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 *
 *
 */

#ifndef TPCC_CLIENT_H
#define TPCC_CLIENT_H

#if defined (solaris)
#include <dce/pthread.h>
#else   /* solaris */
#include <pthread.h>
#endif  /* solaris */


#define MUTEX_T pthread_mutex_t
#define COND_T  pthread_cond_t
#define MUTEX_LOCK(a) pthread_mutex_lock(a)
#define MUTEX_UNLOCK(a) pthread_mutex_unlock(a)
#define COND_WAIT(cond,mut) pthread_cond_wait(cond,mut)
#define COND_SIGNAL(cond) pthread_cond_signal(cond)
#define COND_BROADCAST(cond) pthread_cond_broadcast(cond)
#define MUTEX_INIT(mut)   pthread_mutex_init(mut, pthread_mutexattr_default)
#define COND_INIT(cond)   pthread_cond_init(cond, pthread_condattr_default)
#define MUTEX_DESTROY(mut) pthread_mutex_destroy(mut)
#define COND_DESTROY(cond) pthread_cond_destroy(cond)

/*
 *  Routines and declarations that are common to all clients
 */
void *clnt_thread_init(void);

void thread_done(void *);

void send_new_order(void *, newOrder_data_t *);

void send_payment(void *, payment_data_t *);

void send_order_status(void  *, orderStatus_data_t *);

void send_delivery(void *, delivery_data_t *);

void send_stock_level(void  *, stockLevel_data_t *);

void send_batch_request(void *contextP, int num, tpcc_data_t *dataP);

void send_unmarshalled(void *contextP,
                                        int tran_type,
                                        int size,
                                        ndr_byte *dataP);

void enroll_client(int id);

void *clnt_thread_init(void);

#endif  /*  TPCC_CLIENT_H */
```

client_listen.c

```
/*
 *                client_listen.c
 *
 * $Revision: 1.18 $
 * $Date: 1998/04/29 19:47:41 $
 * $Lo:               $
 *
 *
 * $TALog: client_listen.c,v $
 * Revision 1.18  1998/04/29  19:47:41   wenjian
```

```
* - Use fd instead of stream on NT
* - Add code to consider tpcc_monitor as a special client login
* - Use TRY and CATCH_ALL to deal with exceptions
* [from r1.17 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.1]
*
* Revision 1.17  1998/02/17  22:13:28   wenjian
* [merge of changes from 1.14 to 1.15 into 1.16]
*
* Revision 1.15  1998/02/17  16:04:41  oz
* - Split the login into two parts to allow for speciallogins
* - If the warehouse ID is 0, this is a special login to
*   query the client for status
*
* - First, login
*   If the w_id is bigger than 0: normal thread.
*   Otherwise, call client_report.
* [from r1.14 by delta oz-21864-TPCC-split-client-login-screen, r1.1]
*
* Revision 1.16  1998/02/17  22:06:59   wenjian
* Add head files and define macros for win32
* [from r1.14 by delta wenjian-21750-TPCC-changes-for-porting-on-NT,r1.1]
*
* Revision 1.14  1998/01/28  22:24:48  oz
* [from r1.13 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.4]
*
* Revision 1.13  1998/01/26  16:19:22  oz
* - moved all the code pertaining to the background
*   thread to its own file and all the data structures
*   to client_utils.h
* [from r1.12 by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file,r1.1]
*
* Revision 1.12  1998/01/24  14:17:04  oz
* - User server name to identify server and name delivery file
* - Use env variable HOME instead of /home/encina if HOME is set
*
* - Print the thread ID on thread exit as well
* [from r1.11 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]
*
* Revision 1.11  1998/01/23  15:07:44  oz
* - Updated the SP TPCC directory to the latest files used
*   during the SP tpcc audit.
* [from r1.10 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*
*  Exported functions:
*               make_connections
*
*  Private functions:
*               process_terminal
*
*
*/

/*  client_listen.c
*               Code in the client that listens for requests from the
*               terminal processes and submits them for processing.
*
*               There is one listening function: make_connection.
*               That function calls cnm_ManageConnection which never returns
*               and so it is best to call it in its own independent thread.
*
*               As soon as cnm_ManageConnections receives a connection it
*  starts a new thread and calls process_terminal in that
*  thread passing in the file descriptor for the new connection.
*
*               Note that the client does not need to know in advance how many
*               terminals it will talk to.
*
*               The function process_terminal reads initializes the thread
*  and then calls client_init to process all the requests from
*  that terminal.
*/


#include <stdlib.h>
#ifdef WIN32
#include <io.h>
#endif
#include <stdio.h>
#include <sys/types.h>
#include <tc/tc.h>
#include <do_tpcc.h>
#include <tpcc_type.h>

#if defined (solaris)
#include <dce/pthread.h>
#else   /* solaris */
#include <pthread.h>
#endif  /* solaris */

#include "client_utils.h"
#ifdef WIN32
#include "cnm.h"
#else
#include <cnm/cnm.h>
#endif

#ifdef WIN32
#define         close           _close
#define         fileno          _fileno
#endif
/*
 *  State about the terminal stored by the terminal thread
 *  work_entry: The work entry to be used by this terminal thread.
 */
typedef struct {
   int profiling;
   int terminal_id;
   void *handle_contextP;
} terminal_context_t;

/**
 **  Function Prototypes
 **/
static void process_terminal(cnm_arg_t *argP);

extern void client_init(int, int, int, int, void *);
extern void client_login(int, int, int *, int *);
```

```
/*
 *  process_terminal
 *
 *                  The argument we get is a file descriptor for a terminal
 *                  process.  We read from that file to receive input and send
 *                  output back to that file.
 */
static void process_terminal(cnm_arg_t *argP)
{
   int w_id, d_id;
   terminal_context_t terminal_context;
   tpcc_data_t tran_data;
   int fdIn;
   pthread_t thread = pthread_self();
   int thread_id = pthread_getunique_np(&thread);
   struct timespec rand_sleep;
#if defined(_AIX)
   tid_t tid = thread_self();
#else
   int tid = thread_id;
#endif

#ifdef WIN32
   fdIn = argP->fd;
#else
   fdIn = fileno(argP->stream);
#endif /* WIN32 */

   /*
    *  Default terminal context
    *  This may be updated later by the terminal
    */
   terminal_context.terminal_id = -1;
   terminal_context.profiling = 0;

   TRY {
      client_login(fdIn, fdIn, &w_id, &d_id);
      if (w_id > 0) {
        /*  Initialize the server handle and other thread structures  */
        terminal_context.handle_contextP = (void *)clnt_thread_init();

        logprintf("Tid: %d (0x%x) w_id %d, d_id %d\n", tid, tid, w_id, d_id);
                  client_init(fdIn, fdIn, w_id, d_id, terminal_context.handle_contextP);
                  logprintf("Thread done - Tid %d (0x%x)\n", tid, tid);
                  thread_done(terminal_context.handle_contextP);
      } else {
                  logprintf("Starting Auxiliary Thread, Tid %d (0x%x)\n", tid, tid);
                  client_status_report(fdIn);
                  logprintf("End of Auxiliary Thread, Tid %d (0x%x)\n", tid, tid);
      }
   } CATCH_ALL {
                  err_printf("An exception happened\n");
                  logprintf("End of Auxiliary Thread, Tid %d (0x%x)\n", tid, tid);
   }
   ENDTRY

   close(fdIn);
}

/*
 *  make_connections
 *
 *    Listen for connections on a socket.
 *    Whenever a connection is made, start a thread to talk
 *    to the terminal.
 *                  This functions is spawned on its own thread.
 */
void make_connections(argP)
  void *argP;
{
   int port = (int)argP;
   char port_descr[28];
   int rc;

   DPRINT(("Using socket %d\n", port));
   err_printf("Using thread stack size default\n");
   sprintf(port_descr, "ncacn_ip_tcp[%d]", port);
   rc = cnm_ManageConnections(port_descr,
                                          (cnm_userRountine_t)process_terminal,
                                          NULL,
                                          0,          /*  Max Connections  */
                                          1);  /*  Spawn threads   */
   err_printf("cnm_ManageConnections returned %d\n", rc);
}
```

client_listen.h

```
/*
 *                  client_listen.h
 *
 * $Revision: 1.1 $
 * $Date: 1997/04/20 11:57:55 $
 * $Log:            $
 *
 *
 * $TALog: client_listen.h,v $
 * Revision 1.1  1997/04/20  11:57:55  oz
 * - This is the code base modified at IBM Poughkeepsie
 *   by Ofer Zajicek and Radha Sivaramakrishnan for the
 *   SP scaling test for TPCC.
 * [added by delta oz-19782-TPCC-add-ibm-sp-code, r1.1]
 *
 * Revision 1.1  1995/07/09  18:12:10  oz
 * - Modified the client side of the TPCC benchmark to have multithreaded
 *   clients.  There is a terminal process for each terminal -- when
 *   not using the terminal emulator each terminal process emulates one
 *   terminal.  The terminal processes communication with the client
 *   process using a unix socket.
 *
 *   On the client side there is a thread for each terminal process.
 *   That thread receives the request from the terminal and puts it on
 *   a queue.  There is one processing thread that dequeues the requests
 *   and sends them to the server for processing.
 * [added by delta oz-15875-TPCC-reduce-the-number-of-clients, r1.1]
 *
 *
 *
 */

/*   client_listen.h
 */

#ifndef TPCC_CLIENT_LISTEN_H
#define TPCC_CLIENT_LISTEN_H


void make_connections(void *argP);

#endif /* TPCC_CLIENT_LISTEN_H */
```

client_main.c

```
#include "string.h"
#include "tpcc.h"

extern void client_init(int infd, int outfd, int w_id, int d_id, void *conP);
extern void client_login(int infd, int outfd, int *w_idP, int *d_idP);

main()
{
   int w_id, d_id;
   client_login(0, 1, &w_id, &d_id);
   client_init(0, 1, w_id, d_id, (void *)0);
}


int send_new_order(void *contextP, NewOrder_data *data) {
   int          i;

   data->s_W_ID = 11;
   data->s_D_ID = 22;
   data->s_C_ID = 3333;
   strcpy((char *)data->s_C_LAST, "1234567890123456");
   strcpy((char *)data->s_C_CREDIT, "BC");
   data->s_C_DISCOUNT = 0.1556;
   data->s_O_OL_CNT = 10;
   data->s_O_ID = 4444;
   strcpy((char *)data->s_O_ENTRY_D, "1992-10-2 12:33:11");
   strcpy((char *)data->s_status_line, "12345678901234567890123456");
   data->s_total_amount = 12.98;
   data->s_transtatus = 0;
   data->s_W_TAX = 0.1234;
   data->s_D_TAX = 0.5678;

   for (i=0; i < data->s_O_OL_CNT; i++) {
                data->item[i].s_OL_SUPPLY_W_ID = i + 1;
                data->item[i].s_OL_I_ID = i + 1;
                strcpy((char *)data->item[i].s_I_NAME, "12345678901234567890123456");
                data->item[i].s_OL_QUANTITY = i + 1;
                data->item[i].s_S_QUANTITY = i + 1;
                data->item[i].s_brand_generic[0] = 'B';
                data->item[i].s_I_PRICE = i + 1;
                data->item[i].s_OL_AMOUNT = i + 1;
   }
   return 0;
}

int send_payment(void *contextP, Payment_data *data) {
   data->s_W_ID = 11;
   data->s_D_ID = 22;
   data->s_C_ID = 3333;
   data->s_C_W_ID = 44;
   data->s_C_D_ID = 55;
   data->s_H_AMOUNT = 9.55;
   strcpy((char *)data->s_W_STREET_1, "12345678901234567890");
   strcpy((char *)data->s_W_STREET_2, "12345678901234567890");
   strcpy((char *)data->s_W_CITY, "12345678901234567890");
   strcpy((char *)data->s_W_STATE, "PR");
   strcpy((char *)data->s_W_ZIP, "123456789");
   strcpy((char *)data->s_D_STREET_1, "12345678901234567890");
```

```
    strcpy((char *)data->s_D_STREET_2, "12345678901234567890");
    strcpy((char *)data->s_D_CITY, "1234567890123456789O");
    strcpy((char *)data->s_D_STATE, "PR");
    strcpy((char *)data->s_D_ZIP, "123456789");
    strcpy((char *)data->s_C_FIRST, "1234567890123456");
    strcpy((char *)data->s_C_MIDDLE, "12");
    strcpy((char *)data->s_C_LAST, "1234567890123456");
    strcpy((char *)data->s_C_STREET_1, "12345678901234567890");
    strcpy((char *)data->s_C_STREET_2, "12345678901234567890");
    strcpy((char *)data->s_C_CITY, "1234567890123456789O");
    strcpy((char *)data->s_C_STATE, "PR");
    strcpy((char *)data->s_C_ZIP, "123456789");
    strcpy((char *)data->s_C_PHONE, "1234567890123456");
    strcpy((char *)data->s_C_SINCE, "1992-23-22 21:11:11");
    strcpy((char *)data->s_H_DATE, "1992-10-2 12:33:11");
    strcpy((char *)data->s_C_CREDIT, "BC");
    data->s_C_CREDIT_LIM = 5000;
    data->s_C_DISCOUNT = 0.10;
    data->s_C_BALANCE = 122.10;
    strcpy((char *)data->s_C_DATA,
"12345678901234567890123456789012345678901234567890123456789012345678901234567
7890");
    return 0;
}

int send_order_status(void *contextP, OrderStatus_data *data) {
    int             i;

    data->s_W_ID = 11;
    data->s_D_ID = 22;
    data->s_C_ID = 3333;
    strcpy((char *)data->s_C_FIRST, "1234567890123456");
    strcpy((char *)data->s_C_MIDDLE, "12");
    strcpy((char *)data->s_C_LAST, "1234567890123456");
    data->s_C_BALANCE = 122.10;
    data->s_O_ID = 44;
    strcpy((char *)data->s_O_ENTRY_D,"1992-10-2 12:33:11");
    data->s_O_CARRIER_ID = 55;
    data->s_ol_cnt = 10;

    for (i=0; i < data->s_ol_cnt; i++) {
                data->item[i].s_OL_SUPPLY_W_ID = i + 1;
                data->item[i].s_OL_I_ID = i + 1;
                data->item[i].s_OL_QUANTITY= i + 1;
                data->item[i].s_OL_AMOUNT= i + 1;
                strcpy((char *)data->item[i].s_OL_DELIVERY_D, "1992-10-2 12:33:11");
    }
    return 0;
}

int send_delivery(void *contextP, Delivery_data *data) {
    strcpy((char *)data->s_exec_status, "Delivery has been queued");
    return 0;
}

int send_stock_level(void *contextP, StockLevel_data *data) {
    data->s_low_stock = 22;
    return 0;
}
```

client_term.c

```
/*
 *                terminal.c
 *
 * $Revision: 1.6 $
 * $Date: 1998/01/23 15:07:45 $
 * $Log:            $
 *
 *
 * $TALog: client_term.c,v $
 * Revision 1.6  1998/01/23  15:07:45  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.5 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27,r1.1]
 *
 *
 *
 */


#include "tpcc_type.h"
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include "client_utils.h"

char *tran_type_to_str(int tran_type)
{
    char *ret_val;
    switch (tran_type) {
      case NEWO_TRANS:                  ret_val = "Local-New-Order"; break;
      case PAYMENT_TRANS:               ret_val = "Local-Payment"; break;
      case ORDER_STAT_TRANS:            ret_val = "Order-Status"; break;
      case DELIVERY_TRANS:              ret_val = "Delivery"; break;
      case STOCK_TRANS:                 ret_val = "Stock-Level"; break;
      case NEWO_XA_TRANS:               ret_val = "Distributed-New-Order"; break;
      case PAYMENT_XA_TRANS:            ret_val = "Distributed-Payment"; break;
      default:                          ret_val = "unknown"; break;
    }
    return(ret_val);
}


void clientTerm_Init(int fd)
{
}

int get_transaction(int fd)
{
    char menu[] =
      "NewOrder(1) Payment(2) OrderStatus(3) Delivery(4) StockLevel(5) Quit(6): ";
    extern char *MEbuf;
    extern int MEbufsz;
    char buffer[6];
    int cc;

    write(fd, menu, sizeof(menu));
    cc = read(fd, buffer, sizeof(buffer));
    if ((buffer[0] >= '0') && (buffer[0] <= '5'))
      return(buffer[0] - '0');
    else
      return(-1);
}

#define INT_PROMPT(str, intP)                                   \
do {                                                    \
    char _buf[128];                                     \
    int cc;                                             \
    write(fd, str, strlen(str));                        \
    cc = read(fd, _buf, sizeof(_buf));                  \
    *intP = atol(_buf);                                 \
} while (0)


int get_NO_data(int fd, newOrder_data_t *dataP)
{
    INT_PROMPT("w_id: ", &dataP->w_id);
    INT_PROMPT("d_id: ", &dataP->d_id);
    INT_PROMPT("c_id: ", &dataP->c_id);
}

int get_data(int fd, tpcc_data_t *dataP)
{
    int cc;
    int success;
    cc = read(fd, dataP, sizeof(*dataP));
    if (cc < sizeof(*dataP)) {
                fprintf(stderr, "Read Error: Read %d, expected %d\n",
                        cc, sizeof(*dataP));
                success = 0;
    } else {
                DPRINT(("Read %d bytes, tran: %s (%d)\n",
                        cc, tran_type_to_str(dataP->tran_type), dataP->tran_type));
                success = (dataP->tran_type >= 1) && (dataP->tran_type <= 5);
    }
    return(success);
}

int send_data(int fd, tpcc_data_t *dataP)
{
    int cc;
    cc = write(fd, dataP, sizeof(*dataP));
    if (cc < sizeof(*dataP)) {
                fprintf(stderr, "Write Error: Wrote %d, expected %d\n",
                        cc, sizeof(*dataP));
    } else {
                DPRINT(("Wrote %d bytes, tran: %s (%d)\n",
                        cc, tran_type_to_str(dataP->tran_type), dataP->tran_type));
    }
    return(0);
}
```

client_utils.c

```
/*
 *
 *                client_utils.c
 *
 * $Revision: 1.7 $
 * $Date: 1998/04/29 19:47:42 $
 * $Log:          $
 *
 *
 *
 *
 * $TALog: client_utils.c,v $
 * Revision 1.7  1998/04/29  19:47:42   wenjian
 * - Add prefix_sprintf
 * - Remove ENCINA_C_CALLING_CONVENTION from err_printf
 * [from r1.6 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.1]
 *
 * Revision 1.6  1998/02/17  22:07:00   wenjian
```

```
* Minor changes for NT
* [from r1.5 by delta wenjian-21750-TPCC-changes-for-porting-on-NT,r1.1]
*
* Revision 1.5  1998/01/24  14:17:04  oz
* - User server name to identify server and name delivery file
* - Use env variable HOME instead of /home/encina if HOME is set
*
* - Flush the logfile after each write
* [from r1.4 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]
*
* Revision 1.4  1998/01/23  15:07:46  oz
* - Updated the SP TPCC directory to the latest files used
*   during the SP tpcc audit.
* [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*
*    client_utils.c
*                Generic utilities used by the client processes
*/

#include <stdio.h>
#include <time.h>
#include <string.h>
#include <stdarg.h>

#if defined (solaris)
#include <dce/pthread.h>
#else   /* solaris */
#include <pthread.h>
#endif

#include "databuf.h"
#include "client_utils.h"
#include "do_tpcc.h"
#include "tpcc_type.h"

#define CASE(a) case a: retVal = #a; break

int print_thread_id = 1;
extern int user_id;
extern char *user_code;
/*
 *  Translate the tpcc return code to a string value
 */
static char *TpccRcToStr(rc)
  tpcc_rc_t rc;
{
    char *retVal;
    switch (rc) {
                CASE(INVALID_NEWO);
                CASE(INVALID_HANDLE);
                CASE(SQL_ERROR);
                CASE(TRPC_ERROR);
                CASE(DCE_ERROR);
                CASE(NO_SUCH_LAST_NAME);
                CASE(INVALID_TRAN_TYPE);
                CASE(TPCC_ERROR_BEGIN_NEWO);
                CASE(TPCC_ERROR_DECL_NEWO_SEL_ITEM);
                CASE(TPCC_ERROR_OPEN_NEWO_SEL_ITEM);
                CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_ITEM);
                CASE(TPCC_ERROR_FETCH_NEWO_SEL_ITEM);
                CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_ITEM);
                CASE(TPCC_ERROR_PREP_NEWO_SEL_STCK);
                CASE(TPCC_ERROR_DECL_NEWO_SEL_STCK);
                CASE(TPCC_ERROR_OPEN_NEWO_SEL_STCK);
                CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_STCK);
                CASE(TPCC_ERROR_FETCH_NEWO_SEL_STCK);
                CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_STCK);
                CASE(TPCC_ERROR_NEWO_SELECT);
                CASE(TPCC_ERROR_NEWO_UPD_STCK);
                CASE(TPCC_ERROR_DIST_NEWO_UPD_STCK);
                CASE(TPCC_ERROR_NEWO_SELECT_2);
                CASE(TPCC_ERROR_DECL_NEWO_SEL_CUST);
                CASE(TPCC_ERROR_OPEN_NEWO_SEL_CUST);
                CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_CUST);
                CASE(TPCC_ERROR_FETCH_NEWO_SEL_CUST);
                CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_CUST);
                CASE(TPCC_ERROR_DECL_NEWO_SEL_DIST);
                CASE(TPCC_ERROR_OPEN_NEWO_SEL_DIST);
                CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_DIST);
                CASE(TPCC_ERROR_FETCH_NEWO_SEL_DIST);
                CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_DIST);
                CASE(TPCC_ERROR_PREP_NEWO_INS_OL);
                CASE(TPCC_ERROR_DECL_NEWO_INS_OL);
                CASE(TPCC_ERROR_OPEN_NEWO_INS_OL);
                CASE(TPCC_ERROR_OPEN_DIST_NEWO_INS_OL);
                CASE(TPCC_ERROR_PUT_NEWO_INS_OL);
                CASE(TPCC_ERROR_PUT_DIST_NEWO_INS_OL);
                CASE(TPCC_ERROR_DECL_NEWO_SEL_WARE);
                CASE(TPCC_ERROR_OPEN_NEWO_SEL_WARE);
                CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_WARE);
                CASE(TPCC_ERROR_FETCH_NEWO_SEL_WARE);
                CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_WARE);
                CASE(TPCC_ERROR_EXECUTE_NEWO_UPD_INS);
                CASE(TPCC_ERROR_UPDATE_NEWO_NEXT_OID);
                CASE(TPCC_ERROR_PREP_NEWO_INS);
                CASE(TPCC_ERROR_EXECUTE_DIST_NEWO_INS);
                CASE(TPCC_ERROR_EXECUTE_NEWO_COMMIT);
                CASE(TPCC_ERROR_ROLLBACK_NEWO);
                CASE(TPCC_ERROR_REMOTE_OL_SELECT);
                CASE(TPCC_ERROR_REMOTE_OL_UPDATE);
                CASE(TPCC_ERROR_OPEN_ORDS_CNT_CID);
                CASE(TPCC_ERROR_FETCH_ORDS_CNT_CID);
                CASE(TPCC_ERROR_OPEN_ORDS_SEL_CLAST);
                CASE(TPCC_ERROR_FETCH_ORDS_SEL_CLAST);
                CASE(TPCC_ERROR_OPEN_ORDS_SEL_CID);
                CASE(TPCC_ERROR_FETCH_ORDS_SEL_CID);
                CASE(TPCC_ERROR_OPEN_ORDS_SEL_OLDORD);
                CASE(TPCC_ERROR_FETCH_ORDS_OLDORD);
                CASE(TPCC_ERROR_OPEN_ORDS_SEL_OL);
                CASE(TPCC_ERROR_FETCH_ORDS_SEL_OL);
                CASE(TPCC_ERROR_EXECUTE_ORDS_COMMIT);
                CASE(TPCC_ERROR_OPEN_DELIVERY_OLDEST_OID);
                CASE(TPCC_ERROR_FETCH_DELIVERY_OLDEST_OID);
                CASE(TPCC_ERROR_EXECUTE_DELIVERY_COMMIT);
                CASE(TPCC_ERROR_OPEN_DELIVERY_SEL_ORD);
                CASE(TPCC_ERROR_FETCH_DELIVERY_SEL_ORD);
                CASE(TPCC_ERROR_OPEN_DELIVERY_SEL_SUM_OL);
                CASE(TPCC_ERROR_FETCH_DELIVERY_SEL_SUM_OL);
                CASE(TPCC_ERROR_EXECUTE_DELIVERY_EXEC_DVRY);
                CASE(TPCC_ERROR_SELECT_DELIVERY_ORDER_ID);
                CASE(TPCC_ERROR_SELECT_DELIVERY_CARRIER_ID);
                CASE(TPCC_ERROR_SELECT_DELIVERY_BALANCE);
                CASE(TPCC_ERROR_OPEN_STOCKLEVEL_SEL_OID);
                CASE(TPCC_ERROR_FETCH_STOCKLEVEL_SEL_OID);
                CASE(TPCC_ERROR_OPEN_STOCKLEVEL_CNT_SID);
                CASE(TPCC_ERROR_FETCH_STOCKLEVEL_CNT_SID);
                CASE(TPCC_ERROR_OPEN_STOCKLEVEL_FIND);
                CASE(TPCC_ERROR_FETCH_STOCKLEVEL_FIND);
                CASE(TPCC_ERROR_EXECUTE_STOCKLEVEL_COMMIT);
                CASE(TPCC_ERROR_OPEN_PAYMENT_CNT_CID);
                CASE(TPCC_ERROR_FETCH_PAYMENT_CNT_CID);
                CASE(TPCC_ERROR_OPEN_PAYMENT_SEL_CLAST);
                CASE(TPCC_ERROR_FETCH_PAYMENT_SEL_CLAST);
                CASE(TPCC_ERROR_OPEN_PAYMENT_SEL_CID);
                CASE(TPCC_ERROR_FETCH_PAYMENT_SEL_CID);
                CASE(TPCC_ERROR_DECL_PAYMENT_SEL_DIST);
                CASE(TPCC_ERROR_OPEN_PAYMENT_SEL_DIST);
                CASE(TPCC_ERROR_OPEN_DIST_PAYMENT_SEL_DIST);
                CASE(TPCC_ERROR_FETCH_PAYMENT_SEL_DIST);
                CASE(TPCC_ERROR_FETCH_DIST_PAYMENT_SEL_DIST);
                CASE(TPCC_ERROR_DECL_PAYMENT_SEL_WARE);
                CASE(TPCC_ERROR_OPEN_PAYMENT_SEL_WARE);
                CASE(TPCC_ERROR_OPEN_DIST_PAYMENT_SEL_WARE);
                CASE(TPCC_ERROR_FETCH_PAYMENT_SEL_WARE);
                CASE(TPCC_ERROR_FETCH_DIST_PAYMENT_SEL_WARE);
                CASE(TPCC_ERROR_EXECUTE_PAYMENT_UPD_CUST_LAST);
                CASE(TPCC_ERROR_EXECUTE_PAYMENT_UPD_CUST_ID);
                CASE(TPCC_ERROR_COMMIT_PAYMENT_UPD_CUST);
                CASE(TPCC_ERROR_SELECT_PAYMENT_W_YTD);
                CASE(TPCC_ERROR_SELECT_PAYMENT_D_YTD);
                CASE(TPCC_ERROR_BEGIN_PAYMENT);
                CASE(TPCC_ERROR_EXECUTE_PAYMENT_COMMIT);
      default: retVal = "-Unknown-"; break;
    }
    return(retVal);
}

/*
 *  get_thread_id
 *      A function that returns the thread ID of the current thread
 */
int get_thread_id()
{
    pthread_t thread = pthread_self();
    int thread_id = pthread_getunique_np(&thread);
    return(thread_id);
}

/*
 *  time_diff
 *                Return the difference in miliseconds between two times
 */
int time_diff_ms(t2, t1)
  struct timeval *t2, *t1;
{
    int t_diff;

    t_diff = (t2->tv_usec + 1000000 - t1->tv_usec + 500) / 1000 +
                        (t2->tv_sec - t1->tv_sec - 1) * 1000;

    return(t_diff);
}


#define A_CASE(a,b)  case a: retVal = b; break
/*
 *  Translate the transaction code to its name - for formatting
 */
char *clientUtils_TranCodeToName(type)
  int type;
{
    char *retVal = "-Unknown-";
    switch (type) {
                A_CASE(NEWO_TRANS,      "NEWOR");
                A_CASE(PAYMENT_TRANS,   "PAYMN");
                A_CASE(ORDER_STAT_TRANS, "ORDER");
                A_CASE(DELIVERY_TRANS,  "DELIV");
```

```
                A_CASE(STOCK_TRANS,     "STOCK");
    }
    return(retVal);
}

/*
 *  Print the return status of a TPCC transaction
 *  and the corresponding SQL codes and ISAM codes
 */
void clientUtils_ReportReturn(msg, statusP)
  char *msg;
  data_header *statusP;
{
    switch (statusP->returncode) {
      case SUCCESS_CODE:
                err_printf( "After %s, rc = %d\n", msg, statusP->returncode);
                break;
      case SQL_ERROR:
                err_printf("ERROR: After %s, rc = SQL_ERROR, SQL=%d, ISAM=%d\n", msg,
                                statusP->sql_code,
                                statusP->isam_code);
                break;
      case INVALID_NEWO:
                err_printf("After %s, rc = INVALID_NEWO\n", msg);
                break;
      case DCE_ERROR:
                err_printf("ERROR: After %s, rc = DCE_ERROR\n", msg);
                break;
      case TRPC_ERROR:
                err_printf("ERROR: After %s, rc = TRPC_ERROR\n", msg);
                break;
      case NO_SUCH_LAST_NAME:
                err_printf("After %s, rc = NO_SUCH_LAST_NAME.\n", msg);
                break;
      case DISTRIBUTED_TRAN_FAILED:
                err_printf("After %s, rc = DISTRIBUTED_TRAN_FAILED.\n", msg);
                break;
      default:
                err_printf("ERROR: After %s, rc = %s (%d), SQL=%d, ISAM=%d\n", msg,
                                TpccRcToStr(statusP->returncode),
                                statusP->returncode,
                                statusP->sql_code, statusP->isam_code);
                break;
    }
}


/*
 *  clientUtils_SetReturnCode
 *
 *  Set the return code in the dataP union.
 *  dataP is a pointer to a union of all the transaction types.
 *  Each member of the union has a header field that contains
 *  a return code.  Set the returncode value of the header field
 *  for dataP to be code.
 */
void clientUtils_SetReturnCode(dataP, code)
    tpcc_data_t *dataP;
    tpcc_rc_t code;
{
    switch (dataP->tran_type) {
      case NEWO_TRANS: {
                 newOrder_data_t *ptr = &dataP->data.new_order;
                 ptr->header.returncode = code;
                 break;
      }
      case PAYMENT_TRANS: {
                 payment_data_t *ptr = &dataP->data.payment;
                 ptr->header.returncode = code;
                 break;
      }
      case ORDER_STAT_TRANS: {
                 orderStatus_data_t *ptr = &dataP->data.order_status;
                 ptr->header.returncode = code;
                 break;
      }
      case DELIVERY_TRANS: {
                 delivery_data_t *ptr = &dataP->data.delivery;
                 ptr->header.returncode = code;
                 break;
      }
      case STOCK_TRANS: {
                 stockLevel_data_t *ptr = &dataP->data.stock_level;
                 ptr->header.returncode = code;
                 break;
      }
    }
}

/*
 *  get_prefix
 *              Format the output prefix for printing:
 *              It contains the user_id, 'C' or 'T' depending on whether it
 *              is a terminal or a client and optional a thread identifier
 *              The prefix is written in the buffer passed in by the caller.
 */
void get_prefix(buffer)
  char *buffer;
{
    if (print_thread_id) {
                int thread_id = get_thread_id();
                sprintf(buffer,  "%s(%d-%s-%d)%s",
                                user_id < 10 ? "  " : user_id < 100 ? " " : "",
                                user_id,
                                user_code,
                                thread_id,
                                thread_id < 10 ? " " : "");
    } else {
                sprintf(buffer,  "%s(%2d-%s)",
                                user_id < 10 ? " " : "", user_id, user_code);
    }
}

/*
 * err_printf
 *              A var-arg function that appends the current time and
 *              other data to the print request and sends it to stderr
 */
void err_printf(char *format, ...)
{
    time_t cur_timet;
    char time_str[30];
    char line_prefix[50];
    va_list ap;

    va_start(ap, format);

    cur_timet = time(&cur_timet);
    strftime(time_str, 29, "%X", localtime(&cur_timet));

    get_prefix(line_prefix);

    fprintf(stderr, "%s %s - ", line_prefix, time_str);
    vfprintf(stderr, format, ap);

    va_end(ap);
}


/*
 * logprintf
 *              A var-arg function that prints both to standard error toto
 *              the log file.  It prepends every line with the current time
 *              and the user id.
 */
void logprintf( char *format, ...)
{
    time_t cur_timet;
    char time_str[30];
    char line_prefix[50];
    va_list ap;

    va_start(ap, format);

    cur_timet = time(&cur_timet);
    strftime(time_str, 29, "%X", localtime(&cur_timet));

    get_prefix(line_prefix);

    fprintf(logtpcc ? logtpcc : stderr, "%s %s - ", line_prefix, time_str);
    vfprintf(logtpcc ? logtpcc : stderr, format, ap);
    if (logtpcc)
      fflush(logtpcc);

    if (debug && logtpcc) {
                fprintf(stderr, "%s %s - ", line_prefix, time_str);
                vfprintf(stderr, format, ap);
    }

    va_end(ap);
}

void prefix_sprintf(char *buf, char *format, ...)
{
    time_t cur_timet;
    char time_str[30];
    char line_prefix[50];
    char info[256];
    va_list ap;

    va_start(ap, format);

    cur_timet = time(&cur_timet);
    strftime(time_str, 29, "%X", localtime(&cur_timet));

    get_prefix(line_prefix);

    sprintf(buf, "%s %s - ", line_prefix, time_str);
    vsprintf(info, format, ap);
    strcat(buf,info);

    va_end(ap);
}
```

client_utils.h

```
/*
 *
 *                client_utils.h
 *
 * $Revision: 1.10 $
 * $Date: 1998/04/29 19:47:43 $
 * $Log:         $
 *
 *
 *
 *
 * $TALog: client_utils.h,v $
 * Revision 1.10  1998/04/29  19:47:43   wenjian
 * - Define ENDMSG marking the end of socket message between tpcc_client
 *   and tpcc_monitor
 * - Remove ENCINA_C_CALLING_CONVENTION from err_printf
 * [from r1.9 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.1]
 *
 * Revision 1.9  1998/02/17  22:13:41   wenjian
 * [merge of changes from 1.6 to 1.7 into 1.8]
 *
 * Revision 1.7  1998/02/17  16:04:41  oz
 * - Split the login into two parts to allow for speciallogins
 * - If the warehouse ID is 0, this is a special login to
 *   query the client for status
 * [from r1.6 by delta oz-21864-TPCC-split-client-login-screen, r1.1]
 *
 * Revision 1.8  1998/02/17  22:07:00   wenjian
 * Minor changes for NT
 * [from r1.6 by delta wenjian-21750-TPCC-changes-for-porting-on-NT,r1.1]
 *
 * Revision 1.6  1998/01/26  16:43:32  oz
 * - Removed the code for collectingstats in the client
 *   and dumping them before exit.
 *
 * - Removed timeP and time_allocated from thread_info_t
 * [from r1.5 by delta oz-21691-TPCC-remove-client-stats-code, r1.1]
 *
 * Revision 1.5  1998/01/26  16:19:23  oz
 * - moved all the code pertaining to the background
 *   thread to its own file and all the data structures
 *   to client_utils.h
 * [from r1.4 by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file,r1.1]
 *
 * Revision 1.4  1998/01/23  15:07:47  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 *
 *    client_utils.h
 *                Generic utilities used by the client processes
 */

#ifndef TPCC_CLIENT_UTILS_H
#define TPCC_CLIENT_UTILS_H

#include "encina/symbols.h"
#include "tpcc_type.h"
#include <stdio.h>
#include <time.h>
#include "client.h"
#ifdef WIN32
#include <winsock.h>
#endif
/*
 *  err_printf
 *                Print a string to stderr after prefixing it with the client
 *                info and the current time.
 *  logprintf
 *                Prints as above to the log file.
 */
extern FILE *logtpcc;
extern char log_file_name[];
extern void logprintf( char *format, ...);
extern void err_printf( char *format, ...);
extern void prefix_sprintf( char *buf, char *format, ...);

/*  tran_timing_t: for debug:
 *  Keep track of the timestamps of all the transactions
 *  and dump it out upon exit.  There is an array oftimestamps
 *  per thread and each thread dumps it when it exits.
 */
typedef struct {
  int server;
  int terminal;
  int tran;
  int sub_tran;  /*  Subclass: for NewOrder and payment: 1=>hasRemote  */
  struct timeval start;                         /*  Time received from terminal */
  struct timeval send;                          /*  Time the RPC was made (explicit only) */
  struct timeval srvr_start;   /*  Time received by server   */
  struct timeval srvr_done;   /*  Time sent by server         */
  struct timeval end;                           /*  Time sent to terminal       */
  int num_rms;                                                  /*  Number of RMs the tran involved */
  int tran_failed;
} tran_timing_t;

typedef enum {
   thread_state_init = 0,
   thread_state_called,
   thread_state_sent,
   thread_state_received,
   thread_state_returned,
   thread_state_done
} thread_state_t;

#define NUM_STATES thread_state_done
#define NUM_NEXT_REPORTS 10
#define ENDMSG "...."  /* a special string to mark the end of a message */

typedef struct {
   thread_state_t state;
   int tran;
   struct timeval init, called, sent, received, returned, done;
   int printed, done_printed;
} thread_descr_t;

typedef struct {
    int num;
    int errs;
    double RT[2];
} tran_info_t;
/*
 * total_tran_count_t
 *
 * structure that holds the total count of transaction of each type
 * as well as the reposne times.
 *
 */
typedef struct {
    tran_info_t tran[MAX_TRAN_TYPE + 1];
    int errors;
    double time;
} total_tran_count_t;

/*
 *  thread_info_t
 *
 *  per thread information kept by this module
 */
typedef struct {
    int thread_index;
    int thread_id;
    int initialized;
    tran_timing_t last_tran;
    int num_trans;
    int consecutive_errors;
    thread_descr_t descr;
    tran_info_t tran[MAX_TRAN_TYPE + 1];
    int done;
} thread_info_t;

int  time_diff_ms(struct timeval *t2, struct timeval *t1);

extern int debug;
#define DPRINT(args)  if (debug) err_printf args

extern MUTEX_T init_lock;
extern int info_list_len;
extern thread_info_t **info_list;  /*  List of all the thread info */

/*
 *  A global variable by which the process would like to
 *  identify itself in the prefix to output
 */
extern int user_id;

/*
 *  clientUtils_ReportReturn
 *                Called when a transaction is returned in order to error codes
 */
extern void clientUtils_ReportReturn(char *msg, data_header *statusP);

#define CHECK_ENVIRON(str,var)  if (str == NULL) { fprintf(stderr, \
                                        "%s environment variable is not defined.\n",var); exit(1); }

char *clientUtils_TranCodeToName(int type);

#endif /*  TPCC_CLIENT_UTILS_H */
```

databuf.h

```
/*
 *    databuf.h
 *
 * $Revision: 1.2 $
 * $Date: 1998/01/23 15:07:47 $
 * $Log:         databuf.h,v $
 * Revision 4.2  95/05/16  10:55:31  10:55:31   tpcc (TPCC Benchmark)
 * Added necessary RCS ident strings
 *
 * Revision 4.1  95/05/09  15:21:02  15:21:02   strue (Scott Truesdale)
 * New code from Transarc - initial version
 *
 * Revision 3.2  95/04/03  17:43:09  17:43:09   strue (Scott Truesdale)
```

```
* Changes from Transarc - added sql error handling in client; cleaned up debug handling with
macros; added check on db paramters via call to server.
*
* Revision 3.1  95/04/03  15:10:30  15:10:30   strue (Scott Truesdale)
* Base of rev 3 - shipped to transarc
*
*
*
*
*  $TALog: databuf.h,v $
* Revision 1.2  1998/01/23  15:07:47  oz
* - Updated the SP TPCC directory to the latest files used
*   during the SP tpcc audit.
* [from r1.1 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
* Revision 1.1  1997/04/20  11:57:57  oz
* - This is the code base modified at IBM Poughkeepsie
*   by Ofer Zajicek and Radha Sivaramakrishnan for the
*   SP scaling test for TPCC.
* [added by delta oz-19782-TPCC-add-ibm-sp-code, r1.1]
*
* Revision 1.31  1995/10/30  19:10:54  oz
* [merge of changes from 1.29 to 1.30 into 1.27]
*
* Revision 1.30  1995/10/27  15:41:30  oz
* - Modified the tpc-c code to work with the new informix
*   sql code that is in ex_trans.ec
* [from r1.29 by delta oz-16761-TPCC-modify-code-to-work-with-oracle,r1.1]
*
* Revision 1.27  1995/10/20  18:44:30   ctipper
* [merge of changes from 1.17 to 1.25 into 1.22]
*
* Revision 1.25  1995/10/20  18:15:34   ctipper
* Incorporate changes per code review.
*
* - add DISTRIBUTED_TRAN_FAILED, TPCC_DB_INFO_PARTIAL, and
* TPCC_DB_INFO_FAILED error codes to tpcc_rc_t
* - got rid of MAX_NUM_SERVERS variables
* [from r1.23 by delta ctipper-16547-TPCC-more-distributed-trans, r1.2]
*
* Revision 1.23  1995/10/13  17:00:26   ctipper
* This delta encompasses all changes necessary to do distributed, XA
* transactions with the TPCC benchmark. This includes the changes
* necessary to build with Informix version 6.
*
* Each client still talks to only one server, however, if a distributed
* transaction is necessary, the client sends the request to a different
* interface of that server which then forwards all or part of the
* request on to the appropriate remote server.
*
* - added new error codes to the tpcc_rc_t enumeration.
* - defined MAX_NUM_SERVERS to be 10
* [from r1.19 by delta ctipper-16547-TPCC-more-distributed-trans, r1.1]
*
* Revision 1.19  1995/09/20  21:02:39  oz
* -Corrected code for the payment transaction
* - The distributed case now no longer uses
*   stored procedures
* [from r1.18 by delta oz-16547-TPCC-add-distributed-transactions, r1.2]
*
* Revision 1.18  1995/09/20  17:51:10  oz
* - Added distributed transactions for the new order and
*   payment transaction
*
* - Added new error codes
* [from r1.17 by delta oz-16547-TPCC-add-distributed-transactions, r1.1]
*
* Revision 1.22  1995/10/02  20:31:07  oz
* - Corrected definition of ERROR()
* [from r1.21 by delta oz-16638-tpcc-modify-terminal-for-RTE, r1.3]
*
* Revision 1.21  1995/10/02  18:51:45  oz
* - Added definitions needed for utils.c and liberty.c
* [from r1.20 by delta oz-16638-tpcc-modify-terminal-for-RTE, r1.2]
*
* Revision 1.20  1995/10/02  15:52:35  oz
* - Modified the TPC-C benchmark to be compatible with the RTE.
* - There are now 3 terminal processes:
*     emulator: the old terminal process with a built in
*                  simple emulator
*     curses: An interactive terminal process using curses
*     liberty: An interactive terminal process to be used with
*                  the RTE compatible with the liberty freedom terminal.
*
* - Define TRUE and FALSE only if they are not already defined.
*   (curses.h defines TRUE)
* - Removed READ_TO_DATE and YEAR_TO_SECOND
* - Added term_type_t
* - Added
*               GOOD_INPUT (0)
*               WRONG_INPUT (10)
* [from r1.17 by delta oz-16638-tpcc-modify-terminal-for-RTE, r1.1]
*
* Revision 1.17  1995/07/28  15:28:23  oz
* - Added a -null and -no_marshalling option to TPCC
*
* - Added INVALID_TRAN_TYPE return code
* [from r1.16 by delta oz-16070-TPCC-add-null-and-marshalling-test, r1.1]
*
* Revision 1.16  1995/07/18  17:02:38  oz
* - Added a DCE_ERROR error code
* [from r1.15 by delta oz-15938-TPCC-add-dce-only-client, r1.1]
*
* Revision 1.15  1995/05/22  19:50:48   shl
* [merge of changes from 1.12 to 1.13 into 1.14]
*
* Revision 1.13  1995/05/18  15:11:27  oz
* [from r1.12 by delta oz-15290-TPCC-incorporate-hp-drop-of-05-16-95, r1.1]
*
* Revision 1.14  1995/05/22  17:26:35   ctipper
* [merge of changes from 1.5 to 1.9 into 1.11]
*
* [*** log entries omitted ***]
*
*/

#ifndef __TPCC_DATABUF_H__
#define __TPCC_DATABUF_H__

#define I_NAME_LEN      24
#define I_DATA          50
#define W_NAME_LEN      10
#define ADDR_LEN        20
#define STATE_LEN       2
#define ZIP_LEN         9
#define DIST_INFO_LEN   24
#define S_DATA_LEN      50
#define D_NAME_LEN      10
#define H_DATA_LEN      24
#define CARRIER_LEN     2
#define C_LAST_LEN      17
#define C_MID_LEN       2
#define PHONE_LEN       16
#define CREDIT_LEN      2
#define C_DATA_LEN      500
#define BC_DTA_LEN      23

#define YEAR_TO_DATE   1
#define YEAR_TO_SECOND 2

#define ERROR(x) fprintf(stderr,"Error: %s\n",#x),exit(11)

#define MAX_STR_LEN     255
#define MAX_OL                          15

#ifndef TRUE
#define TRUE    1
#endif
#ifndef FALSE
#define FALSE   0
#endif

#define CANCEL                  -1

#define DATETIME_LEN    19

#define D_PER_W         10

#define COLLECTOR       1                               /* ctipper 5/3/95 */

#define RPC_ERROR                       -2
#define SUCCESS_CODE                    0

#define CHAR_NULL       '\0'                            /* strue 1/23/95 */

typedef enum {
   liberty_term,
   curses_term,
   emulator_term
} term_type_t;

typedef enum {
   TPCC_SUCCESS = 0,
   GOOD_INPUT = 0,

   INVALID_NEWO = 100,
   SQL_ERROR = 2,
   TRPC_ERROR = 3,
   DCE_ERROR = 4,
   NO_SUCH_LAST_NAME = 5,
   INVALID_TRAN_TYPE = 6,
   INVALID_HANDLE = 7,

   WRONG_INPUT = 10,

   DISTRIBUTED_TRAN_FAILED = 15,

   TPCC_DB_INFO_PARTIAL = 20,
   TPCC_DB_INFO_FAILED,

   TPCC_ERROR_BEGIN_NEWO = 110,

   TPCC_ERROR_DECL_NEWO_SEL_ITEM,
   TPCC_ERROR_OPEN_NEWO_SEL_ITEM,
   TPCC_ERROR_OPEN_DIST_NEWO_SEL_ITEM,
   TPCC_ERROR_FETCH_NEWO_SEL_ITEM,
   TPCC_ERROR_FETCH_DIST_NEWO_SEL_ITEM,
   TPCC_ERROR_PREP_NEWO_SEL_STCK,
```

```
    TPCC_ERROR_DECL_NEWO_SEL_STCK,
    TPCC_ERROR_OPEN_NEWO_SEL_STCK,
    TPCC_ERROR_OPEN_DIST_NEWO_SEL_STCK,
    TPCC_ERROR_FETCH_NEWO_SEL_STCK,
    TPCC_ERROR_FETCH_DIST_NEWO_SEL_STCK,
    TPCC_ERROR_NEWO_SELECT,
    TPCC_ERROR_NEWO_UPD_STCK,
    TPCC_ERROR_DIST_NEWO_UPD_STCK,
    TPCC_ERROR_NEWO_SELECT_2,
    TPCC_ERROR_DECL_NEWO_SEL_CUST,
    TPCC_ERROR_OPEN_NEWO_SEL_CUST,
    TPCC_ERROR_OPEN_DIST_NEWO_SEL_CUST,
    TPCC_ERROR_FETCH_NEWO_SEL_CUST,
    TPCC_ERROR_FETCH_DIST_NEWO_SEL_CUST,
    TPCC_ERROR_DECL_NEWO_SEL_DIST,
    TPCC_ERROR_OPEN_NEWO_SEL_DIST,
    TPCC_ERROR_OPEN_DIST_NEWO_SEL_DIST,
    TPCC_ERROR_FETCH_NEWO_SEL_DIST,
    TPCC_ERROR_FETCH_DIST_NEWO_SEL_DIST,
    TPCC_ERROR_PREP_NEWO_INS_OL,
    TPCC_ERROR_DECL_NEWO_INS_OL,
    TPCC_ERROR_OPEN_NEWO_INS_OL,
    TPCC_ERROR_OPEN_DIST_NEWO_INS_OL,
    TPCC_ERROR_PUT_NEWO_INS_OL,
    TPCC_ERROR_PUT_DIST_NEWO_INS_OL,
    TPCC_ERROR_DECL_NEWO_SEL_WARE,
    TPCC_ERROR_OPEN_NEWO_SEL_WARE,
    TPCC_ERROR_OPEN_DIST_NEWO_SEL_WARE,
    TPCC_ERROR_FETCH_NEWO_SEL_WARE,
    TPCC_ERROR_FETCH_DIST_NEWO_SEL_WARE,
    TPCC_ERROR_EXECUTE_NEWO_UPD_INS,
    TPCC_ERROR_UPDATE_NEWO_NEXT_OID,
    TPCC_ERROR_PREP_NEWO_INS,
    TPCC_ERROR_EXECUTE_DIST_NEWO_INS,
    TPCC_ERROR_EXECUTE_NEWO_COMMIT,
    TPCC_ERROR_ROLLBACK_NEWO,
    TPCC_ERROR_REMOTE_OL_SELECT,
    TPCC_ERROR_REMOTE_OL_UPDATE,

    TPCC_ERROR_OPEN_ORDS_CNT_CID = 200,
    TPCC_ERROR_FETCH_ORDS_CNT_CID,
    TPCC_ERROR_OPEN_ORDS_SEL_CLAST,
    TPCC_ERROR_FETCH_ORDS_SEL_CLAST,
    TPCC_ERROR_OPEN_ORDS_SEL_CID,
    TPCC_ERROR_FETCH_ORDS_SEL_CID,
    TPCC_ERROR_OPEN_ORDS_SEL_OLDORD,
    TPCC_ERROR_FETCH_ORDS_OLDORD,
    TPCC_ERROR_OPEN_ORDS_SEL_OL,
    TPCC_ERROR_FETCH_ORDS_SEL_OL,
    TPCC_ERROR_EXECUTE_ORDS_COMMIT,

    TPCC_ERROR_OPEN_DELIVERY_OLDEST_OID = 300,
    TPCC_ERROR_FETCH_DELIVERY_OLDEST_OID,
    TPCC_ERROR_EXECUTE_DELIVERY_COMMIT,
    TPCC_ERROR_OPEN_DELIVERY_SEL_ORD,
    TPCC_ERROR_FETCH_DELIVERY_SEL_ORD,
    TPCC_ERROR_OPEN_DELIVERY_SEL_SUM_OL,
    TPCC_ERROR_FETCH_DELIVERY_SEL_SUM_OL,
    TPCC_ERROR_EXECUTE_DELIVERY_EXEC_DVRY,
    TPCC_ERROR_SELECT_DELIVERY_ORDER_ID,
    TPCC_ERROR_SELECT_DELIVERY_CARRIER_ID,
    TPCC_ERROR_SELECT_DELIVERY_BALANCE,

    TPCC_ERROR_OPEN_STOCKLEVEL_SEL_OID = 400,
    TPCC_ERROR_FETCH_STOCKLEVEL_SEL_OID,
    TPCC_ERROR_OPEN_STOCKLEVEL_CNT_SID,
    TPCC_ERROR_FETCH_STOCKLEVEL_CNT_SID,
    TPCC_ERROR_OPEN_STOCKLEVEL_FIND,
    TPCC_ERROR_FETCH_STOCKLEVEL_FIND,
    TPCC_ERROR_EXECUTE_STOCKLEVEL_COMMIT,

    TPCC_ERROR_OPEN_PAYMENT_CNT_CID = 500,
    TPCC_ERROR_FETCH_PAYMENT_CNT_CID,
    TPCC_ERROR_OPEN_PAYMENT_SEL_CLAST,
    TPCC_ERROR_FETCH_PAYMENT_SEL_CLAST,
    TPCC_ERROR_OPEN_PAYMENT_SEL_CID,
    TPCC_ERROR_FETCH_PAYMENT_SEL_CID,
    TPCC_ERROR_DECL_PAYMENT_SEL_DIST,
    TPCC_ERROR_OPEN_PAYMENT_SEL_DIST,
    TPCC_ERROR_OPEN_DIST_PAYMENT_SEL_DIST,
    TPCC_ERROR_FETCH_PAYMENT_SEL_DIST,
    TPCC_ERROR_FETCH_DIST_PAYMENT_SEL_DIST,
    TPCC_ERROR_DECL_PAYMENT_SEL_WARE,
    TPCC_ERROR_OPEN_PAYMENT_SEL_WARE,
    TPCC_ERROR_OPEN_DIST_PAYMENT_SEL_WARE,
    TPCC_ERROR_FETCH_PAYMENT_SEL_WARE,
    TPCC_ERROR_FETCH_DIST_PAYMENT_SEL_WARE,
    TPCC_ERROR_EXECUTE_PAYMENT_UPD_CUST_LAST,
    TPCC_ERROR_EXECUTE_PAYMENT_UPD_CUST_ID,
    TPCC_ERROR_COMMIT_PAYMENT_UPD_CUST,
    TPCC_ERROR_SELECT_PAYMENT_W_YTD,
    TPCC_ERROR_SELECT_PAYMENT_D_YTD,
    TPCC_ERROR_BEGIN_PAYMENT,
    TPCC_ERROR_EXECUTE_PAYMENT_COMMIT,
    TPCC_ERROR_PAYMENT_UPD_CUST_BY_NAME,
    TPCC_ERROR_PAYMENT_UPD_CUST_BY_ID,
    TPCC_ERROR_PAYMENT_UPDATE_DIST,
    TPCC_ERROR_PAYMENT_UPDATE_WH,
    TPCC_ERROR_PAYMENT_INSERT_HISTORY,
    TPCC_ERROR_EXECUTE_PAYMENT_WH_DIST

} tpcc_rc_t;

typedef enum {
    TPCC_DEADLOCK_MSG = 10,
    TPCC_RETRY_MSG
} tpcc_msg_t;

#endif /* __TPCC_DATABUF_H__ */
```

debug.c

```
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <sys/stat.h>
#include <errno.h>
#include <math.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>
#ifdef WIN32
#include <process.h>
#else
#include <termio.h>
#endif

int print_thread_id = 0;
int user_id = 1;
char *user_code = "C";

int get_thread_id()
{
    return(0);
}

/*
 *  get_prefix
 *              Format the output prefix for printing:
 *              It contains the user_id, 'C' or 'T' depending on whether it
 *              is a terminal or a client and optional a thread identifier
 *              The prefix is written in the buffer passed in by the caller.
 */
void get_prefix(buffer)
  char *buffer;
{
    if (print_thread_id) {
                int thread_id = get_thread_id();
                sprintf(buffer,  "%s(%d-%s-%d)%s",
                                user_id < 10 ? "  " : user_id < 100 ? " " : "",
                                user_id,
                                user_code,
                                thread_id,
                                thread_id < 10 ? " " : "");
    } else {
                sprintf(buffer,  "%s(%2d-%s)",
                                user_id < 10 ? " " : "", user_id, user_code);
    }
}


/*
 * err_printf
 *              A var-arg function that appends the current time and
 *              other data to the print request and sends it to stderr
 */
void err_printf(char *format, ...)
{
    static int initialized = 0;
    static FILE *debug_f = NULL;
    time_t cur_timet;
    char time_str[30];
    char line_prefix[50];
    va_list ap;

    va_start(ap, format);

    if (!initialized) {
                char fileName[45];
                initialized = 1;
                sprintf(fileName, "DebugFile.%d", getpid());
                debug_f = fopen(fileName, "w");
    }

    cur_timet = time(&cur_timet);
    strftime(time_str, 29, "%X", localtime(&cur_timet));

    get_prefix(line_prefix);

    if (debug_f) {
                fprintf(debug_f, "%s %s - ",line_prefix, time_str);
                vfprintf(debug_f, format, ap);
                fflush(debug_f);
    }
```

```
    va_end(ap);
}

void set_client_debug_state(void *contextP, int state, int tran)
{
}
```

Delivery.h

```
#ifndef TRANSARC_delivery_h
#define TRANSARC_delivery_h

#include <trpc/trpc.h>
#include "_delivery.h"

#include <encina/c_prologue.h>

#define delivery_v1_0_c_ifspec          _delivery_v1_0_c_ifspec
#define delivery_v1_0_s_ifspec          _delivery_v1_0_s_ifspec

typedef struct delivery_v1_0_epv {
void (ENCINA_DCE_FUNCTION_PTR_CALLING *impTPCCDelivery)(
#ifdef IDL_PROTOTYPES

                        delivery_data_t *dataP,
                        trpc_status_t  *trpcStatus
#endif
);

} delivery_v1_0_epv_t;

extern void impTPCCDelivery(
#ifdef IDL_PROTOTYPES

                        delivery_data_t *dataP,
                        trpc_status_t  *trpcStatus
#endif
);

trpc_handle_t  mon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
        mon_handle_t handle,
        trpc_tranInfo_t         *tranInfoP,
        trpc_ifSpec_t *ifSpecP
#endif
        );

void        mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
        mon_handle_t handle,
        trpc_handle_t  trpcHandle,
        trpc_tranInfo_t         *tranInfoP,
        trpc_ifSpec_t *ifSpecP
#endif
        );

trpc_handle_t  mon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
        mon_handle_t handle,
        trpc_tranInfo_t         *tranInfoP,
        trpc_ifSpec_t *ifSpecP
#endif
        );

void        mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
        mon_handle_t handle,
        trpc_handle_t  trpcHandle,
        trpc_tranInfo_t         *tranInfoP,
        trpc_ifSpec_t *ifSpecP
#endif
        );

extern delivery_v1_0_epv_t   delivery_v1_0_client_epv;
extern _delivery_v1_0_epv_t delivery_v1_0_manager_epv;
extern rpc_mgr_epv_t                delivery_v1_0_mgr_epv;

#include <encina/c_epilogue.h>
#endif /* TRANSARC_delivery_h */
```

Delivery.tacf

```
/*
 * Copyright (C) 1991, 1990 Transarc Corporation
 * All Rights Reserved
 */
/*
 *  neworder.tacf -- attribute configuration file fortpcc server.
 * used for transparent binding
 *
 * $Revision: 1.1 $
 * $Date: 1997/04/20 11:57:57 $
 * $Log:        tpcc.tacf,v $
 *
 * $TALog: delivery.tacf,v $
 * Revision 1.1  1997/04/20  11:57:57  oz
 * - This is the code base modified at IBM Poughkeepsie
 *   by Ofer Zajicek and Radha Sivaramakrishnan for the
 *   SP scaling test for TPCC.
 * [added by delta oz-19782-TPCC-add-ibm-sp-code, r1.1]
 *
 * Revision 1.3  1996/01/12  16:06:44  oz
 * - Added transaction specific servers:  there are 5 different interfacese
 *   one for each transaction type.
 * [added by delta oz-16955-TPCC-add-transaction-specific-servers, r1.1]
 *
 */

[implicit_handle (mon_handle_t handle)]
interface delivery
{
}
```

delivery.tidl

```
/*
 * id: $id: $
 *
 * component_name: encina benchmarks
 *
 * the following functions list may not be complete.
 * functions defined by/via macros may not be included.
 *
 * functions:
 *    <fill_me_in>
 *
 * origins: transarc corp.
 *
 * (c) copyright transarc corp. 1995, 1993
 * all rights reserved
 * licensed materials - property of transarc
 *
 * us government users restricted rights - use, duplication or
 * disclosure restricted by gsa adp schedule contract with transarc corp
 */
/*
 * history
 * $talog: $
 */

/*
 * delivery.tidl -- interface definition file fortpccserver.
 *
 * $revision: 1.11 $
 * $date: 1995/10/20 21:55:05 $
 * $log:        tpcc.tidl,v $
 */

[uuid(d714d8f8-2105-11cf-830f-0800093b9834),  version(1.0)]

interface delivery
{
import "tpm/mon/mon_handle.idl";
import "tpcc_type.idl";

[nontransactional] void
        impTPCCDelivery([in,out] delivery_data_t *dataP,
                                 [out] trpc_status_t  * trpcStatus);

}
```

do_tpcc.c

```
/*
 *      do_tpcc.c
 *
 * $Revision: 1.14 $
 * $Date: 1998/01/26 20:37:34 $
 * $Log:        do_tpcc.c,v $
 *
 * $TALog: do_tpcc.c,v $
 * Revision 1.14  1998/01/26  20:37:34  oz
 * - Remove all the code associated with explicit binding
 *
 * - Removed bindingType
 * - Removed client_first_wh and client_last_wh
 * - Removed command line args: binding, offset, ware
 * [from r1.13 by delta oz-21697-TPCC-remove-explicit-binding-code, r1.1]
 *
```

```
* Revision 1.13  1998/01/26  15:33:31  oz
* - Changed default binding to transparent
* [from r1.12 by delta oz-21671-TPCC-merge-online-transaction-interfaces, r1.2]
*
* Revision 1.12  1998/01/24  14:17:05  oz
* - User server name to identify server and name delivery file
* - Use env variable HOME instead of /home/encina if HOME is set
* [from r1.11 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]
*
* Revision 1.11  1998/01/23  15:07:48  oz
* - Updated the SP TPCC directory to the latest files used
*   during the SP tpcc audit.
* [from r1.10 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
* Revision 1.8  1997/08/04  19:50:41  oz
* [from r1.7 by delta oz-20506-TPCC-convert-to-new-format-of-connection-manager,r1.2]
*
*
*
*/
/*=========================================================================
* do_tpcc.c
*
*
*  This is the main client program for the TPCC benchmark using Encina.
*
*  The client program is multi-threaded: there is one thread for each
*  terminal and one thread to process incoming connections.
*
*  When the client starts up it starts a listening thread.  That thread
*  calls the encina function cnm_ManageConnections and provides it a
*  port number.  The client spawns a thread (through cnm_ManageConnections)
*  for each terminal that connects to it.  That thread receives the input
*  from the terminal and translates it to a transaction data structure
*  (such as payment_data_t or newOrder_data_t).  The terminal thread
*  (in the client) then sends an RPC over to the server to process the request.
*
                 ____________________________
          |      Client          |
          |      Process         |              ----------
------  RPC |                      |   socket     | Terminal |
    |<------|-------------------Terminal-----~~~~~~-------|      |
    |     |                Thread  |              ----------
Server|     |                      |
    |     |                      |              ----------
    |  RPC |                      |   socket     | Terminal |
------ <------|-------------------Terminal-----~~~~~~-------|      |
          |                Thread  |              ----------
          |                    |
          |____________________________|

*
*
*=========================================================================
*/



#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <sys/stat.h>
#include <errno.h>
#include <math.h>
#include <time.h>
#include <fcntl.h>
#include <termio.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <tpm/mon/mon_client.h>
#include <tc/tc.h>
#include <dce/rpc.h>

#if defined (solaris)
#include <dce/pthread.h>
#else   /* solaris */
#include <pthread.h>
#endif

#include "tpcc_type.h"
#include "utilities.h"
#include "client_utils.h"
#include "do_tpcc.h"
#include "client_listen.h"
#include "client.h"

extern char *sys_errlist[];  /*  Translations of errno file errors */

/*
 *  ENTERING
 *              A macro that is called before processing a TPCC transaction.
 *              If debug mode is enabled it prints a message containing the name
 *              of the transaction being executed.
 */
#define ENTERING(msg) if (debug) err_printf("Entering %s\n", msg)

#define MAX_CONSECUTIVE_ERRS 3000
/*
 *  RETURNED
 *              A macro that is called after a transaction has been processed.
 *              If the transaction failed it reports an error.
 *              In debug mode it also emits a message indicating the processing
 *              has been completed.
 */
#define RETURNED(msg, hdrP)                             \
{                                                       \
  if (((hdrP)->returncode == TPCC_SUCCESS) ||           \
     ((hdrP)->returncode == INVALID_NEWO)) {            \
    consecutiveErrors = 0;                              \
  } else {                                              \
    consecutiveErrors++;                                \
  }                                                     \
  if (debug ||                                          \
     (((hdrP)->returncode != SUCCESS_CODE) &&           \
      ((hdrP)->returncode != INVALID_NEWO))) {          \
    clientUtils_ReportReturn(msg, hdrP);                \
    if (consecutiveErrors > MAX_CONSECUTIVE_ERRS) {     \
      err_printf("Too many consecutive errors (%d)\n",   \
                                        consecutiveErrors);     \
      exit_program(1);                                  \
    }                                                   \
  }                                                     \
}


int                             useSecurity = FALSE;
int                             null_test = 0;
int                             client_lock_handles = 0;

/* The following are global to the client*/
char *LOG_FILE_DIR = "runs/threads";
int             user_id;
int             user_port = 4011;
char    *user_code = "C";    /*  Prefix for output to identify this
                                                                * process as a client or a terminal
                                                                */
int             consecutiveErrors = 0;
char   *result_dir;

int debug = 0;
char log_file_name[100];
int logtrans = 0;
FILE *logtpcc = NULL;

static void check_parms(int argc, char *argv[]);
static void print_header(int argc, char *argv[]);


/*=======================================================*/

main(argc, argv)

int argc;
char *argv[];
{
   check_parms(argc,argv);  /*  Read and parse the command line parameters */

   err_printf("Client %d starting.\n", user_id);

   enroll_client(user_id);  /* enroll as a client */

   /*
    *  Open log file
    */
   logtpcc = fopen(log_file_name, "w");
   print_header(argc, argv);   /** Print a test header to the logfile  **/

   /*
    *    Start the listening thread:
    *    This call will not return
    */
   make_connections((void *)user_port);

   exit_program( 0 );
   return (0);                  /* to statisfy lint */

}      /***************        end of main     *************/

/*=======================================================*/

/*
 * User must supply user_id as a parm and all other parameters
 * as environment variables.
 */

/*---------------------------------------------------------------
 *       Check Parameters
 *  Check the parameters passed in.
 *
 *  Not all the parameters are relevant for this executable.
 *  This code is shared between the regular Encina Monitor
 *  based TPC-C client and other test clients that do not
 *  use the Encina Monitor.  The type of this executable is
 *  in client_type and is set to mon_client for the TPCC
 *  Monitor based client (the audited client).
 *---------------------------------------------------------------
 */
```

```
static void check_parms (argc,argv)
  int argc;
  char *argv[];

{
   char *host_name = getenv("HOST");
   char *home_dir = getenv("HOME");
   int             next_arg = 1;
   int errors = 0;
   char          *progName;
   int print_help = 0;

   user_id = -1;
   result_dir = ".";

   while (next_arg < argc) {
                if (!strcasecmp("-debug", argv[next_arg])) {
                   /*  Enable debug mode (for testing)  */
                   debug = 1;
                } else if (!strcasecmp("-dir", argv[next_arg])) {
                   /*  The directory for the client output */
                   result_dir = argv[++next_arg];
                } else if (!strcasecmp("-log", argv[next_arg])) {
                   /*  A less intrusive form of debug mode  */
                   logtrans = 1;
                } else if (!strcasecmp("-id", argv[next_arg])) {
                   /*  The id of this client  */
                   user_id = atol(argv[++next_arg]);
                } else if (!strcasecmp("-port", argv[next_arg])) {
                   /*  The id of this client  */
                   user_port = atol(argv[++next_arg]);
                   if (user_id < 0) user_id = user_port;
                } else if (!strcasecmp("-security", argv[next_arg])) {
                   /*  Enable security between the client and the server.
                    *  This is enabled by default
                    */
                   useSecurity = TRUE;
                } else if (!strcasecmp("-noSecurity", argv[next_arg])) {
                   /*  Disable security between the client and the server.
                    *  This is enabled by default
                    */
                   useSecurity = FALSE;
                } else if (!strcasecmp("-null", argv[next_arg])) {
                   /*  For testing: do not access the data in the DB  */
                   logprintf("Performing NULL test\n");
                   null_test = 1;
                } else if (!strcasecmp("-lock", argv[next_arg])) {
                   logprintf("Locking longterm handles\n");
                   client_lock_handles = atol(argv[++next_arg]);
                } else {
                   printf("invalid parameter: %s\n", argv[next_arg]);
                   print_help = 1;
                   break;
                }
     next_arg++;
   }

   if (user_id < 0) {
     printf("  Missing User  Id\n");
     print_help = 1;
   }

   if (print_help) {
                progName = strrchr(argv[0], '/');
                progName = (progName ? progName + 1 : argv[0]);

                printf("\nusage:\n You can specify the following in anyorder\n");
                printf(" You must specify the Id\n");

                printf("   -id <num>       The user ID for this client\n");
                printf("   -dir <dir>      Directory for output (default \".\")\n");
                printf("   -debug          enable debugging\n");
                printf("   -log            log all activity to a file\n");
                printf("   -security       enable secure communications between the client andPA\n");
                printf("   -null           NULL test: the server immediately returns\n");

                exit(-1);
   }

   sprintf(log_file_name, "%s/%s/C.%s.%d",
                   home_dir ? home_dir : "/home/encina",
                   LOG_FILE_DIR,
                   host_name ? host_name : "host", user_id);

/*
 *  print_header:
 *                Print some feedback to the user on the client configuration
 */
static void print_header(int argc, char *argv[])
{
   int i;
   if (!logtpcc)
                return;

   logprintf("Client %d starting a %s test.\n",
                     user_id,
                     null_test ? "NULL" : "DB");

   logprintf("Log file name %s\n", log_file_name);
   logprintf("Params: ");

   for (i=0; i<argc; i++) {
                fprintf(logtpcc, "%s ", argv[i]);
   }
   fprintf(logtpcc, "\n");
   fflush(logtpcc);
}
```

do_tpcc.h

```
/*
 * do_tpcc.h
 *
 *  $Revision: 1.7 $
 *  $Date: 1998/01/23 15:07:49 $
 *  $Log:       do_tpcc.h,v $
 *
 *  $TALog: do_tpcc.h,v $
 * Revision 1.7  1998/01/23  15:07:49  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.6 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 *
 */

#ifndef _DO_TPCC_H_INCLUDED_
#define _DO_TPCC_H_INCLUDED_

#include <dce/rpc.h>
#include <trpc/trpc.h>
#include "databuf.h"

#define WRONG_INPUT    0
#define NEW_ORDER      1
#define PAYMENT                         2
#define ORDER_STATUS   3
#define DELIVERY       4
#define STOCK_LEVEL    5
#define QUIT           9
#define MIN_OL                          5
#define MAX_FLDS       200     /* Maximum fields in a TPC-C form */

#define THRESHOLD_LEN  2

#define ON             1
#define OFF            0

#define YES   1
#define NO    0

#define INSIZE  1024

#define DO_ROLLBACK     1
#define DONT_ROLLBACK   0

/**  The response time requirements for the transactions in seconds.
 **  90% of the transactions are required to have a response time less
 **  than or equal to the value below.
 **/
#define NEWORD_90RT     5
#define PAYMENT_90RT    5
#define ORDSTAT_90RT    5
#define DELIVERY_90RT   5  /* 5 for interactive or 80 for background*/
#define STOCKLEV_90RT   20

/*
 *  What type of client is this?
 */
typedef enum {
   tk_client,
   dce_client,
   mon_client,
   db_client
} client_type_t;

extern client_type_t client_type;

typedef enum {
   transparent, explicit, longTerm, noReservation
} binding_t;

/* Handle from client to PA is now described using both the paHandle
and the mondHandle. */

#define NUM_TRANS       5
#define NEWO_ERR        6
#define PAYMENT_ERR     7
#define ORD_STAT_ERR    8
#define DELIVERY_ERR    9
#define STOCK_ERR       10
#define NEWO_ROLLBACK   11
#define END_OF_WINDOW 0xff
#define BEGIN_WINDOW  0xaa
#ifndef SHORT_WAITS
```

```
#define NEWO_MEAN_THINK_TIME            122
#define PAYMENT_MEAN_THINK_TIME                 122
#define ORDER_STAT_MEAN_THINK_TIME      102
#define DELIVERY_MEAN_THINK_TIME        51
#define STOCK_MEAN_THINK_TIME           51
#define NEWO_MIN_KEY_TIME               185
#define PAYMENT_MIN_KEY_TIME            31
#define ORDER_STAT_MIN_KEY_TIME                 21
#define DELIVERY_MIN_KEY_TIME           21
#define STOCK_MIN_KEY_TIME              21
#else
#define NEWO_MEAN_THINK_TIME            61
#define PAYMENT_MEAN_THINK_TIME                 61
#define ORDER_STAT_MEAN_THINK_TIME      51
#define DELIVERY_MEAN_THINK_TIME        26
#define STOCK_MEAN_THINK_TIME           26
#define NEWO_MIN_KEY_TIME               93
#define PAYMENT_MIN_KEY_TIME            16
#define ORDER_STAT_MIN_KEY_TIME                 11
#define DELIVERY_MIN_KEY_TIME           11
#define STOCK_MIN_KEY_TIME              11
#endif


#endif /* _DO_TPCC_H_INCLUDED_ */
```

encina.C

```
/* (C)1997 IBM Corporation */
/*****************************************************************/
/*                                                               */
/*                File: tuxclient.h                              */
/*                                                               */
/*****************************************************************/

#include <stdlib.h>
#include "inout.h"
#include "encina.h"

extern "C" {
}

extern "C" send_new_order(void *contextP, NewOrder_data *dataP);
extern "C" send_payment(void *contextP, Payment_data *dataP);
extern "C" send_stock_level(void *contextP, StockLevel_data *dataP);
extern "C" send_order_status(void *contextP, OrderStatus_data *dataP);
extern "C" send_delivery(void *contextP, Delivery_data *dataP);


void Encina::cleanup() {
}

Encina::Encina() {
   return;
}

Encina::~Encina() {
   return;
}

int Encina::tran (NewOrder_data *dataP, void *contextP, char *servname) {
   send_new_order(contextP, dataP);
   return 0;
}


int Encina::tran (Payment_data *dataP, void *contextP, char *servname) {
   send_payment(contextP, dataP);
   return 0;
}


int Encina::tran (OrderStatus_data *dataP, void *contextP, char *servname) {
   send_order_status(contextP, dataP);
   return 0;
}


int Encina::tran (StockLevel_data *dataP, void *contextP, char *servname) {
   send_stock_level(contextP, dataP);
   return 0;
}

int Encina::tran (Delivery_data *dataP, void *contextP, char *servname) {
   send_delivery(contextP, dataP);
   return 0;
}

int Encina::tran (char *servname) {
   return -1;
}

int Encina::atran (char *servname) {

   return 0;
}
```

encina.h

```
/* (C)1997 IBM Corporation */
/*****************************************************************/
/*                                                               */
/*     File: tuxclient.h                                         */
/*                                                               */
/*****************************************************************/

#ifndef ENCINA_H
#define ENCINA_H

const int TMINBUFSIZE = 1536;

class Encina {
public:
   static void cleanup();
   int tran(char *servname);
   int tran(NewOrder_data *dataP, void *contextP, char *servname);
   int tran(Payment_data *dataP, void *contextP, char *servname);
   int tran(StockLevel_data *dataP, void *contextP, char *servname);
   int tran(OrderStatus_data *dataP, void *contextP, char *servname);
   int tran(Delivery_data *dataP, void *contextP, char *servname);
   int atran(char *servname);
   Encina();
   ~Encina();
};

extern Encina encina;

#endif
```

encina_client.c

```
/*
 *                encina_client.c
 *
 * $Revision: 1.5 $
 * $Date: 1998/01/23 15:07:51 $
 * $Log:            $
 *
 *
 * $TALog: encina_client.c,v $
 * Revision 1.5  1998/01/23  15:07:51  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.4 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 *
 *
 */

/*
 *  encina_client.c
 *
 *                The Encina related code in the client that is common to both
 *                the monitor client and the toolkit client.
 *
 */

#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <trpc/trpc.h>
#include <encina/encina.h>
#include "utilities.h"
#include "client_utils.h"
#include "encina_client.h"


static trpc_handle_t bind_to_server(char *name);

/*
 *  encina_error_message
 *
 *  Report an encina error message by interpreting it and writing
 *  it to both the logfile (if any) and to standard error
 */
void encina_error_message(msg, n)
  char *msg;
  unsigned long n;
{
   char errorMsg[ENCINA_MAX_STATUS_STRING_SIZE];
   encina_StatusToString(n, ENCINA_MAX_STATUS_STRING_SIZE, errorMsg);
   err_printf("ERROR: %s.  Error code = %s (%d0x%x) \n", msg, errorMsg, n, n);
}

/*
 *  encina_error
 *
```

```
 *  This is called for FATAL errors.  It reports the error and exits.
 */
void encina_error(funcName, n)
  char *funcName;
  unsigned long n;
{
   char msg[128];
   sprintf("%s failed", funcName);
   encina_error_message(msg,  n);
   exit_program(1);
}

/*
 *  secure_handle
 *
 *              Secure a handle to an encina server.
 *              This can be called with either a PA handle or with
 *              a trpc handle to a toolkit server.
 */
void secure_handle(trpc_handle_t handle, int use_security)
{
   rpc_binding_handle_t       rpcHandle;
   unsigned long              status = 0;
   unsigned char              *serverPrincipal;

   ENCINA_CALL("trpc_GetRpcHandleFromBinding",
          trpc_GetRpcHandleFromBinding(handle,&rpcHandle));

   rpc_mgmt_inq_server_princ_name(rpcHandle, rpc_c_authn_default,
                        &serverPrincipal, &status);


   if (use_security) {
              DPRINT(("rpc_binding_set_auth_info -> principal %s, protect %d,authn %d authz
%d\n",
                        serverPrincipal, rpc_c_protect_level_connect,
                        rpc_c_authn_default, rpc_c_authz_dce));

              rpc_binding_set_auth_info(rpcHandle, serverPrincipal,
                                                  rpc_c_protect_level_connect,
                                                  rpc_c_authn_default,
                                                  NULL,
                                                  rpc_c_authz_dce,
                                                  &status);
   } else {
              DPRINT(("rpc_binding_set_auth_info -> principal %s, protect %d,authn %d authz
%d\n",
                        serverPrincipal, rpc_c_protect_level_none,
                        rpc_c_authn_default, rpc_c_authz_dce));

              rpc_binding_set_auth_info(rpcHandle, serverPrincipal,
                                                  rpc_c_protect_level_none,
                                                  rpc_c_authn_default,
                                                  NULL,
                                                  rpc_c_authz_dce,
                                                  &status);
   }


   if (status != rpc_s_ok) {
      switch (status) {
         case rpc_s_invalid_binding :
           printf("rpc binding invalid ****** \n");
           break;
         case rpc_s_wrong_kind_of_binding :
           printf("rpc binding is the wrong kind \n");
           break;
         case rpc_s_unknown_authn_service :
           printf("rpc authn service unknown \n");
           break;
      } /* switch */
      bde_Exit(1); }
```

encina_client.h

```
/*
*              encina_client.h
*
* $Revision: 1.5 $
* $Date: 1998/01/23 15:07:52 $
* $Log:          $
*
*
* $TALog: encina_client.h,v $
* Revision 1.5  1998/01/23  15:07:52  oz
* - Updated the SP TPCC directory to the latest files used
*   during the SP tpcc audit.
* [from r1.4 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*
*    Declarations common to monitor version and toolkit version
*
*/

#ifndef ENCINA_CLIENT_H
#define ENCINA_CLIENT_H

#include <trpc/trpc.h>

void encina_error_message(char *msg, unsigned long n);
void encina_error(char *funcName, unsigned long n);
void secure_handle(trpc_handle_t handle, int use_security);


#endif  /*  ENCINA_CLIENT_H  */
```

field.C

```
/* (C)1997 IBM Corporation*/
#include <stdio.h>
#include "field.h"
#include "inout.h"
#include "format.h"
#if 0
#if USE_ALLOCA
#include <alloca.h>
#endif
#endif

extern char const * const blanks;
extern char const * const underscores;
extern char const * const backspaces;
extern int position(InOut *ioP, int x, int y);

Field *genfield(InOut *ioP, int x, int y, int len, int *ptr) {
   return new IntField(ioP, x, y, len, ptr);
}
Field *genfield(InOut *ioP, int x, int y, int len, short *ptr) {
   return new ShortField(ioP, x, y, len, ptr);
}
Field *genfield(InOut *ioP, int x, int y, int len, long *ptr) {
   return new LongField(ioP, x, y, len, ptr);
}
Field *genfield(InOut *ioP, int x, int y, int len, char *ptr) {
   return new TextField(ioP, x, y, len, ptr);
}
Field *genfield(InOut *ioP, int x, int y, int len, double *ptr) {
   return new MoneyField(ioP, x, y, len, ptr);
}
Field *genfield(InOut *ioP, int x, int y, int len, unsigned char *ptr) {
   return new Int8Field(ioP, x, y, len, ptr);
}

/*************************************************************************
Field
*************************************************************************/
Field::Field(InOut *inoutP, int size, char *str)
   : ioP(inoutP), len(size), pos(0), changed(0), need_redisplay(0)
{
   need_free_string = need_free = 0;
   if (str == NULL) {
              string = new char[len+1];
              need_free_string = 1;
   } else {
              string = str;
   }
   ok_func = NULL;
   ok_data = NULL;
   string[0] = 0;
}

Field::Field(InOut *ioP, int inx, int iny, int size, char *str)
   : ioP(ioP), x(inx), y(iny), len(size), pos(0), changed(0), need_redisplay(0)
{
   need_free_string = need_free = 0;
   if (str == NULL) {
              string = new char[len+1];
              need_free_string = 1;
   } else {
              string = str;
   }
   ok_func = NULL;
   ok_data = NULL;
   string[0] = 0;
}

int Field::reset() {
   pos=0;
   changed=0;
   return 0;
}
Field::~Field() {
   if (need_free_string)
              delete [] string;
}
int Field::finalize_field() {
   changed = 0;
   string[pos] = 0;
   return 0;
}
int Field::display_field(int use_underscores) {
   position(ioP, x,y);
   ioP->write(string);
```

```
    if (use_underscores) {
                ioP->write(underscores, len-pos);
    } else {
                ioP->write(blanks, len-pos);
    }
    return 0;
}
int Field::get_key() {
    char key;
    int cc;
    cc = ioP->read(&key, 1);

    return (cc == 0)  ? EOF : key ;
}
int Field::add_char(int key) {
    if (pos >= len || (!isprint(key) && key != ' ')) {
                ioP->write("\a", 1);
                return 1;
    }
    changed = 1;
    string[pos] = key;
    ioP->write(&string[pos++], 1);
    return 0;
}
int Field::backspace() {
    ioP->write("\b_\b", 3);
    changed = 1;
    pos--;
    return 0;
}
int Field::start_position () {
    position(ioP, x, y);
    return 0;
}
int Field::get_field (int need_pos) {
    int key;

    if (need_pos)
                position(ioP, x, y);
    if (pos != 0) {
                need_redisplay = 1;
                ioP->write(string, pos);
                ioP->write(underscores, len-pos);
                if (len-pos < 6)
                    ioP->write(backspaces, len-pos);
                else
                    position(ioP, x+pos, y);
    }

    ioP->mark();
    while (1) {
                key = get_key();
                switch(key) {
                case EOF:
                    return EOF;

                case '\r':   /* Carriage Return */
                case '\n':   /* Newline */
                    ioP->hold();
                    if (changed) {
                                finalize_field();
                    }
                    ioP->pop();
                    display_field(1);
                    return ENTER;
                    break;

                case '\t':   /* Tab */
                case '\006': /* Ctrl-F */
                case '\016': /* Ctrl-N */
                    if (changed) {
                                finalize_field();
                    }
                    ioP->pop();
                    display_field(1);
                    return NEXT_FIELD;
                    break;

                case '\002': /* Ctrl-B */
                case '\020': /* Ctrl-P */
                    if (changed) {
                                finalize_field();
                    }
                    ioP->pop();
                    display_field(1);
                    return PREV_FIELD;

                case '\b':   /* Backspace */
                case '\177': /* Del */
                    if (pos > 0) {
                                backspace();
                    } else
                                ioP->write("\a", 1);
                    break;

                case '\014': /* Ctrl-L */
                    ioP->pop();
                    return REDISPLAY;
                case '\030': /* Ctrl-X */
                case '\003': /* Ctrl-C */
                    ioP->unmark();
                    return ABORT;

                default:
                    add_char(key);
                }
    }
}

/*****************************************************************************
IntField
*****************************************************************************/
IntField::IntField(InOut *ioP, int inx, int iny, int size, int *val) : Field(ioP, inx, iny, size), value(val) {
    if (value==NULL) {
                value = new int;
                need_free=1;
    }
}
IntField::IntField(InOut *ioP, int size, int *val) : Field(ioP, size), value(val) {
    if (value==NULL) {
                value = new int;
                need_free=1;
    }
}
IntField::~IntField() {
    if (need_free)
                delete value;
}

int IntField::add_char(int key) {
    if (pos < len && isdigit(key)) {
                changed = 1;
                string[pos] = key;
                ioP->write(&string[pos++], 1);
                return 0;
    }
    ioP->write("\a", 1);
    return 1;
}
int IntField::display_field(int use_underscores) {
    int firstchar;
#if USE_ALLOCA
    char *buf = (char *)alloca(len+1);
#else
    char *buf = new char[len+1];
#endif
    memset(buf, 'x', len);
    if (pos)
                firstchar = format_int(buf, len+1, *value);
    else
                firstchar = len;
    position(ioP, x, y);
    if (use_underscores) {
                ioP->write(underscores, firstchar);
                ioP->write(buf+firstchar, len-firstchar);
    } else {
                ioP->write(buf, len);
    }
    return 0;
}
int IntField::finalize_field() {
    changed = 0;
    string[pos] = 0;
    if (value != NULL)
                *value = atoi(string);
    return 0;
}

/*****************************************************************************
ShortField
*****************************************************************************/
ShortField::ShortField(InOut *ioP, int inx, int iny, int size, short *val) : Field(ioP, inx, iny, size),
value(val) {
    if (value==NULL) {
                value = new short;
                need_free=1;
    }
}
ShortField::ShortField(InOut *ioP, int size, short *val) : Field(ioP, size), value(val) {
    if (value==NULL) {
                value = new short;
                need_free=1;
    }
}
ShortField::~ShortField() {
    if (need_free)
                delete value;
}

int ShortField::add_char(int key) {
    if (pos < len && isdigit(key)) {
                changed = 1;
                string[pos] = key;
                ioP->write(&string[pos++], 1);
                return 0;
    }
    ioP->write("\a", 1);
```

```
    return 1;
}
int ShortField::display_field(int use_underscores) {
    int firstchar;
#if USE_ALLOCA
    char *buf = (char *)alloca(len+1);
#else
    char *buf = new char[len+1];
#endif
    if (pos)
                firstchar = format_short(buf, len+1, *value);
    else
                firstchar = len;
    position(ioP, x, y);
    if (use_underscores) {
                ioP->write(underscores, firstchar);
                ioP->write(buf+firstchar, len-firstchar);
    } else {
                ioP->write(buf, len);
    }
    return 0;
}
int ShortField::finalize_field() {
    changed = 0;
    string[pos] = 0;
    if (value != NULL)
                *value = atoi(string);
    return 0;
}

/*****************************************************************************
ShortField
*****************************************************************************/
Int8Field::Int8Field(InOut *ioP, int inx, int iny, int size, unsigned char *val) : Field(ioP, inx, iny,
size), value(val) {
    if (value==NULL) {
                value = new unsigned char;
                need_free=1;
    }
}
Int8Field::Int8Field(InOut *ioP, int size, unsigned char *val) : Field(ioP, size), value(val) {
    if (value==NULL) {
                value = new unsigned char;
                need_free=1;
    }
}
Int8Field::~Int8Field() {
    if (need_free)
                delete value;
}

int Int8Field::add_char(int key) {
    if (pos < len && isdigit(key)) {
                changed = 1;
                string[pos] = key;
                ioP->write(&string[pos++], 1);
                return 0;
    }
    ioP->write("\a", 1);
    return 1;
}
int Int8Field::display_field(int use_underscores) {
    int firstchar;
#if USE_ALLOCA
    char *buf = (char *)alloca(len+1);
#else
    char *buf = new char[len+1];
#endif
    if (pos)
                firstchar = format_char(buf, len+1, *value);
    else
                firstchar = len;
    position(ioP, x, y);
    if (use_underscores) {
                ioP->write(underscores, firstchar);
                ioP->write(buf+firstchar, len-firstchar);
    } else {
                ioP->write(buf, len);
    }
    return 0;
}
int Int8Field::finalize_field() {
    changed = 0;
    string[pos] = 0;
    if (value != NULL)
                *value = atoi(string);
    return 0;
}

/*****************************************************************************
LongField
*****************************************************************************/
LongField::LongField(InOut *ioP, int inx, int iny, int size, long *val) : Field(ioP, inx, iny, size),
value(val) {
    if (value==NULL) {
                value = new long;
                need_free=1;
    }
}
LongField::LongField(InOut *ioP, int size, long *val) : Field(ioP, size), value(val) {
    if (value==NULL) {
                value = new long;
                need_free=1;
    }
}
LongField::~LongField() {
    if (need_free)
                delete value;
}
int LongField::add_char(int key) {
    if (pos < len && isdigit(key)) {
                changed = 1;
                string[pos] = key;
                ioP->write(&string[pos++], 1);
                return 0;
    }
    ioP->write("\a", 1);
    return 1;
}
int LongField::display_field(int use_underscores) {
    int firstchar;
#if USE_ALLOCA
    char *buf = (char *)alloca(len+1);
#else
    char *buf = new char[len+1];
#endif
    if (pos)
                firstchar = format_long(buf, len+1, *value);
    else
                firstchar = len;
    position(ioP, x, y);
    if (use_underscores) {
                ioP->write(underscores, firstchar);
                ioP->write(buf+firstchar, len-firstchar);
    } else {
                ioP->write(buf, len);
    }
    return 0;
}
int LongField::finalize_field() {
    changed = 0;
    string[pos] = 0;
    if (value != NULL)
                *value = atoi(string);
    return 0;
}

/*****************************************************************************
MoneyField
*****************************************************************************/
MoneyField::MoneyField(InOut *ioP, int inx, int iny, int size, double *val) : Field(ioP, inx, iny, size),
value(val) {
    seen_dollar = seen_sign = seen_dot = seen_digit = 0;
    if (value==NULL) {
                value = new double;
                need_free=1;
    }
}
MoneyField::MoneyField(InOut *ioP, int size, double *val) : Field(ioP, size), value(val) {
    seen_dollar = seen_sign = seen_dot = seen_digit = 0;
    if (value==NULL) {
                value = new double;
                need_free=1;
    }
}
MoneyField::~MoneyField() {
    if (need_free)
                delete value;
}
int MoneyField::add_char(int key) {
    do {
                if (pos >= len)
                    break;
                if (key == '$') {
                    if (!(pos == 0 || (pos == 1 && seen_sign))) break;
                    seen_dollar = 1;
                } else if (key == '-') {
                    if (!(pos == 0 || (pos == 1 && seen_dollar))) break;
                    seen_sign = 1;
                } else if (key == '.') {
                    if (seen_dot) break;
                    seen_dot = 1;
                } else if (!isdigit(key))
                    break;
                if (seen_dot) {
                    if (seen_dot >= 4)
                                break;
                    seen_dot++;
                }
                changed = 1;
                string[pos] = key;
                ioP->write(&string[pos++], 1);
                return 0;
    } while (0);
    ioP->write("\a", 1);
    return 1;
}
int MoneyField::backspace() {
```

```
    ioP->write("\b_\b", 3);
    changed = 1;
    pos--;
    if (seen_dot)
                seen_dot--;
    if (string[pos] == '-')
                seen_sign = 0;
    if (string[pos] == '$')
                seen_dollar = 0;
    if (string[pos] == '.')
                seen_dot = 0;
    return 0;
}
int MoneyField::display_field(int use_underscores) {
    int firstchar;
#if USE_ALLOCA
    char *buf = (char *)alloca(len+1);
#else
    char *buf = new char[len+1];
#endif
    if (pos)
                firstchar = format_money(buf, len+1, *value);
    else
                firstchar = len;
    position(ioP, x, y);
    if (use_underscores) {
                ioP->write(underscores, firstchar);
                ioP->write(buf+firstchar, len-firstchar);
    } else {
                ioP->write(buf, len);
    }
    return 0;
}
int MoneyField::finalize_field() {
    changed = 0;
    string[pos] = 0;
    if (value != NULL) {
                *value = atof(string + seen_dollar + seen_sign);
                if (seen_sign)
                   *value = -*value;
    }
    return 0;
}
int MoneyField::reset() {
    Field::reset();
    seen_dollar = seen_sign = seen_dot = seen_digit = 0;
    return 0;
}

/*****************************************************************************
TextField
*****************************************************************************/
TextField::TextField(InOut *ioP, int inx, int iny, int size, char *str) : Field(ioP, inx, iny, size, str) {
    value=TextField::string;
}
TextField::TextField(InOut *ioP, int size, char *str) : Field(ioP, size, str) {
    value=TextField::string;
}
int TextField::add_char(int key) {
    if (pos >= len || (!isalnum(key) && key != ' ' && key != '.')) {
                ioP->write("\a", 1);
                return 1;
    }
    changed = 1;
    string[pos] = key;
    ioP->write(&string[pos++], 1);
    return 0;
}
```

field.h

```
/* (C)1997 IBM Corporation */
#if !defined(INCLUDE_FIELD_H)
#define INCLUDE_FIELD_H

#include "inout.h"

class Field {
public:
    enum return_codes { INVALID, ENTER, NEXT_FIELD, PREV_FIELD, ABORT, REDISPLAY };
    InOut *ioP;
    int x, y;
    const int len;
    int pos;
    int changed;
    int need_redisplay;
    char *string;
    int (*ok_func)(void *data);
    int need_free;
    int need_free_string;
    void *ok_data;
    Field(InOut *ioP, int size, char *string=NULL);
    Field(InOut *ioP, int x, int y, int size, char *string=NULL);
    virtual ~Field();
    virtual int get_field (int need_pos=1);
    int get_key ();
    virtual int backspace();
    virtual int reset();
    virtual int start_position();
    virtual int add_char(int key);
    virtual int display_field(int use_underscores=0);
    virtual int finalize_field();

    class Error {
                enum { USER_ABORT };
    };
};

class Int8Field : public Field {
public:
    unsigned char *value;
    int add_char(int key);
    int display_field(int use_underscores=0);
    int finalize_field();

    Int8Field(InOut *ioP, int x, int y, int size, unsigned char *value=NULL);
    Int8Field(InOut *ioP, int size, unsigned char *value=NULL);
    virtual ~Int8Field();
};

class ShortField : public Field {
public:
    short *value;
    int add_char(int key);
    int display_field(int use_underscores=0);
    int finalize_field();

    ShortField(InOut *ioP, int x, int y, int size, short *value=NULL);
    ShortField(InOut *ioP, int size, short *value=NULL);
    virtual ~ShortField();
};


class IntField : public Field {
public:
    int *value;
    int add_char(int key);
    int display_field(int use_underscores=0);
    int finalize_field();

    IntField(InOut *ioP, int x, int y, int size, int *value=NULL);
    IntField(InOut *ioP, int size, int *value=NULL);
    virtual ~IntField();
};

class LongField : public Field {
public:
    long *value;
    int add_char(int key);
    int display_field(int use_underscores=0);
    int finalize_field();

    LongField(InOut *ioP, int x, int y, int size, long *value=NULL);
    LongField(InOut *ioP, int size, long *value=NULL);
    virtual ~LongField();
};

class MoneyField : public Field {
public:
    int seen_dollar, seen_sign, seen_dot, seen_digit;
    double *value;
    int add_char(int key);
    int reset();
    int backspace();
    int display_field(int use_underscores=0);
    int finalize_field();
    MoneyField(InOut *ioP, int x, int y, int size, double *value=NULL);
    MoneyField(InOut *ioP, int size, double *value=NULL);
    virtual ~MoneyField();
};

class TextField : public Field {
public:
    char *value;
    int add_char(int key);
    TextField(InOut *ioP, int x, int y, int size, char *value=NULL);
    TextField(InOut *ioP, int size, char *value=NULL);
};

Field *genfield(InOut *ioP, int x, int y, int len, int *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, short *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, long *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, char *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, unsigned char *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, double *ptr);

#endif /* INCLUDE_FIELD_H */
```

format.C

```
/* (C)1997 IBM Corporation */
#include <string.h>
#include <math.h>

int format_char(char *buf, int size, char val) {
   int neg, pos;
   pos = size;
   buf[--pos] = 0;
   if (val == 0 && pos > 0) {
                buf[--pos] = '0';
                neg = 0;
   } else {
                neg = (val < 0) ? 1 : 0;
                if (neg) val = -val;
                while (val && pos > 0) {
                   buf[--pos] = (val % 10) + '0';
                   val /= 10;
                }
   }
   /* Too long */
   if (!pos && (val || neg)) {
                memset (buf, '*', size);
                return -1;
   }
   if (neg)
                buf[--pos] = '-';
   if (pos)
                memset(buf, ' ', pos);
   return pos;
}
int format_short(char *buf, int size, short val) {
   int neg, pos;
   pos = size;
   buf[--pos] = 0;
   if (val == 0 && pos > 0) {
                buf[--pos] = '0';
                neg = 0;
   } else {
                neg = (val < 0) ? 1 : 0;
                if (neg) val = -val;
                while (val && pos > 0) {
                   buf[--pos] = (val % 10) + '0';
                   val /= 10;
                }
   }
   /* Too long */
   if (!pos && (val || neg)) {
                memset (buf, '*', size);
                return -1;
   }
   if (neg)
                buf[--pos] = '-';
   if (pos)
                memset(buf, ' ', pos);
   return pos;
}
int format_int(char *buf, int size, int val) {
   int neg, pos;
   pos = size;
   buf[--pos] = 0;
   if (val == 0 && pos > 0) {
                buf[--pos] = '0';
                neg = 0;
   } else {
                neg = (val < 0) ? 1 : 0;
                if (neg) val = -val;
                while (val && pos > 0) {
                   buf[--pos] = (val % 10) + '0';
                   val /= 10;
                }
   }
   /* Too long */
   if (!pos && (val || neg)) {
                memset (buf, '*', size);
                return -1;
   }
   if (neg)
                buf[--pos] = '-';
   if (pos)
                memset(buf, ' ', pos);
   return pos;
}
int format_long(char *buf, int size, long val) {
   int neg, pos;
   pos = size;
   buf[--pos] = 0;
   if (val == 0 && pos > 0) {
                buf[--pos] = '0';
                neg = 0;
   } else {
                neg = (val < 0) ? 1 : 0;
                if (neg) val = -val;
                while (val && pos > 0) {
                   buf[--pos] = (val % 10) + '0';
                   val /= 10;
                }
   }
   /* Too long */
   if (!pos && (val || neg)) {
                memset (buf, '*', size);
                return -1;
   }
   if (neg)
                buf[--pos] = '-';
   if (pos)
                memset(buf, ' ', pos);
   return pos;
}

int format_float(char *buf, int size, int dec, double val) {
   static double pow10[] = { 1, 10, 100, 1000, 10000, 100000, 1000000 };
   int neg, pos;
   pos = size;
   buf[--pos] = 0;
   val = rint(val * pow10[dec]);
   neg = (val < 0) ? 1 : 0;
   if (neg) val = -val;

   while (val >= 1 && pos > 0) {
                if (!dec--) {
                   buf[--pos] = '.';
                   continue;
                }
                buf[--pos] = (int)fmod(val, 10) + '0';
                val /= 10;
   }
   if (dec >= 0) {
                while (dec >= 0 && pos > 0) {
                   if (!dec--) {
                                buf[--pos] = '.';
                   } else {
                                buf[--pos] = '0';
                   }
                }
                if (pos > 0)
                   buf[--pos] = '0';
   }
   /* Too long */
   if (!pos && (val >= 1 || neg)) {
                memset (buf, '*', size);
                return -1;
   }
   if (neg)
                buf[--pos] = '-';
   if (pos)
                memset(buf, ' ', pos);
   return pos;
}

int format_money(char *buf, int size, double val) {
   int pos;
   pos = format_float(buf, size, 2, val);
   if (pos > 0)
                buf[--pos] = '$';
   return pos;
}

int format_date(char *buf, int size, unsigned char* val) {
   memcpy(buf, val, size);
   buf[size]=0;
   return 0;
}

int format_phone(char *buf, int size, unsigned char *phone) {
   buf[0] = phone[0];
   buf[1] = phone[1];
   buf[2] = phone[2];
   buf[3] = phone[3];
   buf[4] = phone[4];
   buf[5] = phone[5];
   buf[6] = '-';
   buf[7] = phone[6];
   buf[8] = phone[7];
   buf[9] = phone[8];
   buf[10] = '-';
   buf[11] = phone[9];
   buf[12] = phone[10];
   buf[13] = phone[11];
   buf[14] = '-';
   buf[15] = phone[12];
   buf[16] = phone[13];
   buf[17] = phone[14];
   buf[18] = phone[15];
   buf[19] = '\0';
   return size;
}

int format_zip(char *buf, int size, unsigned char *zip) {
   buf[0] = zip[0];
   buf[1] = zip[1];
   buf[2] = zip[2];
   buf[3] = zip[3];
   buf[4] = zip[4];
   buf[5] = '-';
   buf[6] = zip[5];
   buf[7] = zip[6];
   buf[8] = zip[7];
   buf[9] = zip[8];
```

```
    buf[10] = '\0';
    return size;
}
```

format.h

```
/* (C)1997 IBM Corporation */
#if !defined(INCLUDE_FORMAT_H)
#define INCLUDE_FORMAT_H

int format_char (char *buf, int size, char   val);
int format_int  (char *buf, int size, int    val);
int format_long (char *buf, int size, long   val);
int format_short(char *buf, int size, short  val);
int format_float(char *buf, int size, int    dec, double val);
int format_money(char *buf, int size, double val);
int format_date (char *buf, int size, unsigned char  *val);
int format_phone(char *buf, int size, unsigned char  *phone);
int format_zip  (char *buf, int size, unsigned char  *zip);

#endif /* INCLUDE_FORMAT_H */
```

format_test.C

```
/* (C)1997 IBM Corporation */
#include "format.h"
#include <stdio.h>

void int_test() {
    char buf[256];
    int i;
    for (i = -100; i < -10; i+=10) {
                format_int(buf, 10, i);
                printf("%-10s %10d\n", buf, i);
    }
    for (i = -10; i < 10; i+=1) {
                format_int(buf, 10, i);
                printf("%-10s %10d\n", buf, i);
    }
    for (i = 10; i < 100; i+=10) {
                format_int(buf, 10, i);
                printf("%-10s %10d\n", buf, i);
    }
    for (i = 100; i < 1000; i+=100) {
                format_int(buf, 10, i);
                printf("%-10s %10d\n", buf, i);
    }
    for (i = 1000; i < 10000; i+=1000) {
                format_int(buf, 10, i);
                printf("%-10s %10d\n", buf, i);
    }
}

void double_test() {
    char buf[256];
    double i;
    for (i = -100; i < -10; i+=10) {
                format_float(buf, 10, 2, i);
                printf("%-10s %10.2f\n", buf, i);
    }
    for (i = -10; i < 10; i+=0.01) {
                format_float(buf, 10, 2, i);
                printf("%-10s %10.2f\n", buf, i);
    }
    for (i = 10; i < 100; i+=10) {
                format_float(buf, 10, 2, i);
                printf("%-10s %10.2f\n", buf, i);
    }
    for (i = 100; i < 1000; i+=100) {
                format_float(buf, 10, 2, i);
                printf("%-10s %10.2f\n", buf, i);
    }
    for (i = 1000; i < 10000; i+=1000) {
                format_float(buf, 10, 2, i);
                printf("%-10s %10.2f\n", buf, i);
    }
}

int main () {
    int_test();
    double_test();
}
```

inout.C

```
/* (C)1997 IBM Corporation */
#include <string.h>
#include <strings.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <errno.h>

#include "screen.h"

extern char *sys_errlist[];

#if 1
void InOut::write(const void *buf, size_t size) {
    if (IOError) return;
    debug("write('%*.*s', %d);\n", size, size, buf, size);
    output.queue(buf, size);
    if (!Hold && input.len() == 0) { /* Don't write anything until there is no input */
                flush();
    }
}

ssize_t InOut::read(void *buf, size_t size) {
    int rc;
    if (IOError) return(0);
    while (input.len() < size) {
                rc = ::read(in_fd, input.ptr(), input.free());
        debug("::read('%*.*s', %d) = %d;\n", rc, rc, input.ptr(), input.free(), rc);
        if (inlog) {
                fwrite(input.ptr(), rc, 1, inlog);
                fflush(inlog);
        }
                if (rc > 0) {
                    input.queue(rc);
                } else if (rc <= 0) {
                    IOError = 1;
                    return(0);
                }
    }
    memcpy(buf, input.ptr(), size);
    input.dequeue(size);
    debug("read('%*.*s', %d) = %d;\n", size, size, buf, size, size);
    return size;
}
#else
void InOut::write(const void *buf, size_t size) {
    debug("write('%s', %d);\n", buf, size);
    ::write(out_fd, buf, size);
}

ssize_t InOut::read(void *buf, size_t size) {
    int rc;
    rc = ::read(in_fd, buf, size);
    debug("read('%s', %d) = %d;\n", buf, size, rc);
    return rc;
}
#endif

void InOut::flush() {
    debug("flush();\n");
    Hold = 0;
    if (IOError) return;
    while (output.len()) {
        debug("::write('%*.*s', %d);\n", output.len(), output.len(), output.ptr(), output.len());
                int rc = ::write(out_fd, output.ptr(), output.len());
        if (outlog) {
                fwrite(output.ptr(), rc, 1, outlog);
                fflush(outlog);
        }
                if (rc > 0) {
                    output.dequeue(rc);
                } else if (rc < 0) {
                    err_printf("Error writing data!\n");
                    IOError = 1;
                    return;
                }
    }
}

void InOut::write(const void *buf) {
    write(buf, strlen((const char *)buf));
}

InOut::InOut(int in, int out) : input(256), output(2048) {
    struct termios buf;

#ifdef DEBUG
    {
                char buf[256];
                sprintf(buf, "logs/debug.%d", getpid());
                debugfile = fopen(buf, "w");
                sprintf(buf, "logs/in.%d", getpid());
                inlog     = fopen(buf, "w");
                sprintf(buf, "logs/out.%d", getpid());
                outlog    = fopen(buf, "w");
    }
#endif

    int rc;
    Hold = 0;
    debugfile = inlog = outlog = (FILE *)0;
    IOError = 0;

    in_fd = in;
```

```
  if (out < 0)
                out_fd = in;
  else
    out_fd = out;
  if ((rc = tcgetattr(in_fd, &save_term)) < 0) {
                return;
  }

  buf = save_term;

  buf.c_lflag &= ~(ECHO | ICANON); /* echo off, canonical mode off*/

  buf.c_cc[VMIN] = 1;    /* Case B: 1 byte at a time, no timer*/
  buf.c_cc[VTIME] = 0;

  err_printf("echo off - tcsetattr on %d\n", in_fd);
  if (tcsetattr(in_fd, TCSAFLUSH, &buf) < 0)
                return;
}

InOut::~InOut() {
  if (tcsetattr(in_fd, TCSAFLUSH, &save_term) < 0)
                return;
}
```

inout.h

```
/* (C)1997 IBM Corporation */

#ifndef INOUT_H
#define INOUT_H
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <termios.h>
#include <stdarg.h>
#include <string.h>

#include "tpcc.h"

/* This is for a VT100 */
#if 1
#define ESC "\033"
#define ESCc '\033'
#else
#define ESCc '^'
#define ESC "^"
#endif

#define TRIGGER "\021"
#define TRIGGERc '\021'

extern "C" err_printf(...);

#define POS(x,y) ESC "[" #y ";" #x "H"
#define CLEAR_EOS ESC "[J"

class InOut {
private:
  class Buffer {
  private:
                int BufSize;
                enum { NUMMARKS=8 };
                char *buffer;
                int marks[NUMMARKS];
  public:
                int Pos;
                int Start;
    int num_marks;
                Buffer(int size) {
                  BufSize = size;
                  buffer = new char[BufSize];
                  Pos = Start = 0;
                  num_marks = 0;
                }
                int  pos()      { return Pos; };
                void pos(int P)  { Pos = P; };
                int  start()    { return Start; };
                void start(int S) { Start = S; };
                int  len()      { return Pos-Start; };
                int  free()     { return BufSize-Pos-1; };
                void *ptr()     { return &buffer[Start]; };
                int  lastmark()  { if (num_marks) return marks[num_marks-1]; return 999; };

                void  mark()    {
                  if (num_marks < NUMMARKS)
                                marks[num_marks++] = Pos;
                  else {
                                fprintf(stderr, "Buffer mark overflow\n");
                                exit (1);
                  }
                }
                void unmark() {
                  if (num_marks <= 0)
                                return;
                  num_marks--;
                }
                void pop() {
                  if (num_marks <= 0)
                                return;
                  if (marks[num_marks-1] >= Start) {
                                Pos=marks[--num_marks];
                  } else {
                                num_marks=0;
                  }
                }
                void queue(int size) {
                  Pos += size;
                }
                void queue(const void *buf, int size) {
                  /* If this is too big see if we can move what we have over */
                  if (size+Pos >= BufSize) {
                                if (size + len() >= BufSize) {
                                  fprintf(stderr, "Buffer overflow\n");
                                  exit (1);
                                }
                                /* This requires memcpy to be "safe" */
                                if (Start + len() >= BufSize) {
                                  fprintf(stderr, "Strange Error: Start %d +len %d >= size %d\n",
                                                Start ,len(), BufSize);
                                  exit(1);
                                }
                                memcpy(buffer, &buffer[Start], len());
                                Pos -= Start;

                                /* Fix up our marks*/
                                int count = 0;
                                for (int i = 0; i < num_marks; i++) {
                                  if (marks[i] - Start >= 0)
                                                marks[count++] = marks[i] - Start;
                                }
                                num_marks = count;
                                Start = 0;
                  }
                  memcpy(&buffer[Pos], buf, size);
                  Pos += size;
                }
                void dequeue(int size) {
                  Start += size;
                  if (Start >= Pos) {
                                /* Fix up our marks*/
                                int count = 0;
                                for (int i = 0; i < num_marks; i++) {
                                  if (marks[i] - Start >= 0)
                                                marks[count++] = marks[i] - Start;
                                }
                                num_marks = count;

                                Start = Pos = 0;
                  }
                }
  };
  int in_fd, out_fd;
  int Hold;
  struct termios save_term;
  Buffer input;
  Buffer output;
  FILE *debugfile;
  FILE *inlog, *outlog;
public:
  int IOError;
  ssize_t read(void *buf, size_t size);
  void write(const void *buf, size_t size);
  void write(const void *buf);
  void flush();
  void mark() { debug("mark()\n"); output.mark(); };
  void unmark() { debug("unmark()\n"); output.unmark(); };
  void pop()  { debug("pop()\n\n"); output.pop(); };
  void hold() { debug("hold()\n");  Hold = 1; };
#ifdef DEBUG
  void debug(char *fmt, ...) {
                va_list args;

    fprintf(debugfile, "Start=%2d, Pos=%2d, Marks=%2d(%03d): ", output.Start, output.Pos,
output.num_marks, output.lastmark());
                va_start(args,fmt);
                vfprintf (debugfile, fmt, args);
                va_end (args);
                ::fflush(debugfile);
  }
#else
  void debug(char *fmt, ...) {};
#endif
  InOut(int in=0, int out=1);
  ~InOut();
};

extern char const * const blanks;
extern char const * const underscores;
extern char const * const backspaces;

int format_int(char *buf, int size, int val);
int format_float(char *buf, int size, int dec, double val);
int format_money(char *buf, int size, double val);

#endif /*  INOUT_H */
```

Jointraces

```
#define _LARGE_FILES
#include <stdio.h>
#include "analyze.h"

#define DATAFILE_VERSION 0x00000004


/*
struct data_header {
   char magic[8];
   int version;
   struct timeval absolute_time;
   int begin_time;
   int ramp_up;
   int ramp_down;
   int run_time;
   char user_header[MAX_USER_HEADER];
};
class Log_Event {
public:
   int timestamp;   // time since everything started (in ms)
   int flags;       // extra flags
   int slavenum;    // slave that generated event
   int rtenum;      // rte on slave that generated event (if applicable)
   int  type;       // type of event
};
class Variable_Entry {
public:
   char name[VARIABLENAME_LEN];
   int  flags;
   int  type;
   union {
                  char text[VARIABLETEXT_LEN];
                  int  number;
   } value;
};
*/

int output_pos = 0;
int debug = 0;

int atos (char *string);

#define FP_BUFSIZE (1024*1024*50)
char input_buffer[FP_BUFSIZE];
char output_buffer[FP_BUFSIZE];

void usage (char *name) {
   printf ("Usage: %s [options] file...\n", name);
   printf ("       -o name       outputfilename\n");
   printf ("       -v           verbose\n");
   printf ("       -t time       truncate attime\n");
   exit (1);
}

class file_info {
public:
   char *name;
   FILE *fp;
   struct timeval absolute_time;
   int offset;
   int begin_time;
   int run_time;
   int ramp_up;
   int ramp_down;
};

int read_header(file_info *info) {
   struct data_header header;
   info->fp = fopen(info->name, "r");
   if (info->fp == NULL) {
                  fprintf (stderr, "\nCan't open file '%s': %s(%d)\n",
                        info->name, strerror(errno), errno);
                  return 1;
   }
   setvbuf(info->fp, input_buffer, _IOFBF, sizeof(input_buffer));
   if (fread(&header, sizeof(header), 1, info->fp) != 1) {
                  fprintf (stderr, "\nCan't read header from file '%s': %s(%d)\n",
                        info->name, strerror(errno), errno);
                  return 1;
   }
   if (header.version != DATAFILE_VERSION) {
                  fprintf (stderr, "\nFile '%s' has incorrect version 0x%08x, should be 0x%08x\n",
                              info->name, header.version, DATAFILE_VERSION);
                  return 1;
   }
   info->ramp_up       = header.ramp_up;
   info->run_time      = header.run_time;
   info->ramp_down     = header.ramp_down;
   info->begin_time    = header.begin_time;
   info->absolute_time = header.absolute_time;
   return 0;
}

int write_header(file_info& info, FILE *output_fp) {
   struct data_header header;
   int rc;
   memset(&header, 0, sizeof(header));

   strncpy ((char *)&header.magic, "rtedata", 8);
   header.version       = DATAFILE_VERSION;
   header.absolute_time = info.absolute_time;
   header.ramp_up       = info.ramp_up;
   header.run_time      = info.run_time;
   header.ramp_down     = info.ramp_down;
   header.begin_time    = info.begin_time;

   rc = fwrite (&header, sizeof(header), 1, output_fp);
   if (rc != 1) {
                  fprintf (stderr, "\nCan't write file header information: %s(%d)\n",
                        strerror(errno), errno);
                  return 1;
   }
   output_pos += rc;

   return 0;
}

int dump_file(file_info &info, FILE* output_fp, int *max_slave, int *max_num, int truncate) {
   const int MAX_TIMESTAMPS=10;
   struct header_s header;
   int times[MAX_TIMESTAMPS];
   char buffer[8192];
   fpos_t pos;
   int count = 0;
   int rc, i;
   int last_slave = *max_slave;
   int last_num   = *max_num;

   if (debug)
                  printf ("\noffset=%d\n", info.offset);
   while (1) {
                  if (debug)
                     printf ("position=%d\n", output_pos);
                  if (debug==0 &&  count++ % 10000 == 0) {
                     printf ("Processing '%s' record %d\r", info.name, count-1);
                     fflush(stdout);
                  }

                  if (fgetpos(info.fp, &pos) < 0) {
                     fprintf (stderr, "\nCan't find current position: %s(%d)\n",
                                 strerror(errno), errno);
                     return 1;
                  }
                  // Read and Output header block for this transaction
                  rc = fread(&header, sizeof(header), 1, info.fp);
                  if (rc == EOF || rc == 0)
                     break;
                  if (rc != 1) {
                     fprintf (stderr, "\nCan't read header from file '%s': %s(%d)\n",
                                 info.name, strerror(errno), errno);
                     return 1;
                  }
                  header.slave += last_slave;
                  header.num   += last_num;
                  if (header.slave > *max_slave)
                     *max_slave = header.slave;
                  if (header.num > *max_num)
                     *max_num = header.num;
                  if (debug)
                     printf ("type=%d, timestamps=%d, user_data_length=%d, slave=%d, num=%d\n",
                                 header.type, header.num_timestamps,
                                 header.user_data_length, header.slave, header.num);

                  // Read and Output timestamps for this transaction
                  if (header.num_timestamps < 0 ||
                     header.num_timestamps >= MAX_TIMESTAMPS) {
                     fsetpos(info.fp, &pos);
                     fprintf(stderr, "\nIllegal number of timestamps %d\n",
                                 header.num_timestamps);
                     return 1;
                  }
                  rc = fread(times, sizeof(int), header.num_timestamps, info.fp);
                  if (rc != header.num_timestamps) {
                     fsetpos(info.fp, &pos);
                     fprintf (stderr, "\nCan't read times from file '%s': %s(%d)\n",
                                 info.name, strerror(errno), errno);
                     return 1;
                  }
                  // Fix up times for this transaction
                  for (i = 0; i < header.num_timestamps; i++) {
                     times[i] += info.offset;
                  }
                  if (truncate && times[0]/1000 >= truncate) {
                     fsetpos(info.fp, &pos);
                     fprintf (stderr, "\nTruncating file at timestamp\n");
                     return 1;
                  }
                  rc = fwrite (&header, sizeof(header), 1, output_fp);
                  if (rc != 1) {
                     fsetpos(info.fp, &pos);
                     fprintf (stderr, "\nCan't write header from file '%s': %s(%d)\n",
```

```
                        info.name, strerror(errno), errno);
                return 1;
            }
            output_pos += rc;

            rc = fwrite (times, sizeof(int), header.num_timestamps, output_fp);
            if (rc != header.num_timestamps) {
                fsetpos(info.fp, &pos);
                fprintf (stderr, "\nCan't write times from file '%s': %s(%d)\n",
                        info.name, strerror(errno), errno);
                return 1;
            }
            output_pos += rc;

            // Read and Output user data for this transaction
            if (header.user_data_length > sizeof(buffer)) {
                fsetpos(info.fp, &pos);
                fprintf(stderr, "\nuser_data too large: %d\n", header.user_data_length);
                return 1;
            }
            rc = fread(&buffer, header.user_data_length, 1, info.fp);
            if (rc == EOF)
                return 0;
            if (rc != 1) {
                fsetpos(info.fp, &pos);
                fprintf (stderr, "\nCan't read user_data from file '%s': %s(%d)\n",
                        info.name, strerror(errno), errno);
                return 1;
            }
            rc = fwrite (&buffer, header.user_data_length, 1, output_fp);
            if (rc != 1) {
                fsetpos(info.fp, &pos);
                fprintf (stderr, "\nCan't write user_data from file '%s': %s(%d)
(rc=%d,len=%d)\n",
                        info.name, strerror(errno), errno, rc, header.user_data_length);
                return 1;
            }
            output_pos += rc;
            // Done with record
        }
        printf ("Processing '%s' record %d\n", info.name, count);
        *max_slave+=1;
        *max_num+=1;
        return 0;
}

int main (int argc, char *argv[]) {
        char *output_name = "combined.output";
        int rc, c, i, num_files, truncate = 0;
        FILE* output_fp;
        int idx;
        file_info *files;

        setbuf (stdout, NULL);

        while ((c=getopt(argc,argv,"t:vo:"))!=EOF){
            switch (c) {
                case 'v':  debug++; break;
                case 'o':  output_name = optarg; break;
                case 't':  truncate    =atos(optarg); break;
                case 'h':
                case '?':
                default: usage(argv[0]); break;
            }
        }

        num_files = argc-optind;
        files = new file_info[num_files];

        rc = 0;
        for (i = 0; i < num_files; i++) {
                files[i].name = argv[optind+i];
                rc += read_header(&files[i]);
        }
        // We had an error
        if (rc) {
                exit (1);
        }

        // Open output file
        output_fp = fopen(output_name, "w");
        if (output_fp == NULL) {
                fprintf (stderr, "\nCan't open output file '%s': %s(%d)\n",
                        output_name, strerror(errno), errno);
                exit (1);
        }
        setvbuf(output_fp, output_buffer, _IOFBF, sizeof(output_buffer));

        // Find the first file
        idx = 0;
        for (i = 1; i < num_files; i++) {
                if (files[i].absolute_time.tv_sec < files[idx].absolute_time.tv_sec ||
                    (files[i].absolute_time.tv_sec == files[idx].absolute_time.tv_sec &&
                    files[i].absolute_time.tv_usec < files[idx].absolute_time.tv_usec)){
                    idx = i;
                }
        }
        // Calculate offsets for files
        for (i = 0; i < num_files; i++) {
                files[i].offset = (files[i].absolute_time.tv_sec -
                                        files[idx].absolute_time.tv_sec) * 1000 +
                                       (files[i].absolute_time.tv_usec -
                                        files[idx].absolute_time.tv_usec) / 1000;
        }
        // Dump new header
        write_header(files[idx], output_fp);
        // Dump each file in turn
        int max_slave = 0, max_num = 0;
        for (i = 0; i < num_files; i++) {
                if (dump_file(files[i], output_fp, &max_slave, &max_num, truncate)) {
                    if (truncate) {
                            fprintf (stderr, "File Truncated\n");
                    } else {
                            fprintf (stderr, "Merge terminated\n");
                            exit (1);
                    }
                }
        }

        return 0;
}

int atos (char *string) {
        int val=0, i;
        char *ptr;

        ptr = strtok(string, ":");
        if (ptr == NULL) {
                return atoi(string);
        }
        for (i = 0; i < 3 && ptr != NULL; i++) {
                val = val*60+atoi(ptr);
                ptr = strtok(NULL, ":");
        }
        printf ("val=%d\n", val);
        return val;
}
```

Makefile

```
#!/bin/ksh
#
#
# $Revision: 1.19 $
# $Date: 1998/01/26 20:37:33 $
# $Log:         Makefile,v $
#
# HISTORY
# $TALog: Makefile,v $
# Revision 1.19  1998/01/26  20:37:33  oz
# - Remove all the code associated with explicit binding
#
# - Removed mon_client_utils.c
# [from r1.18 by delta oz-21697-TPCC-remove-explicit-binding-code, r1.1]
#
# Revision 1.18  1998/01/26  16:43:30  oz
# - Removed the code for collecting stats in the client
#   and dumping them before exit.
# [from r1.17 by delta oz-21691-TPCC-remove-client-stats-code, r1.1]
#
# Revision 1.17  1998/01/26  16:19:20  oz
# - moved all the code pertaining to the background
#   thread to its own file and all the data structures
#   to client_utils.h
# [from r1.16 by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file, r1.1]
#
# Revision 1.16  1998/01/26  15:33:31  oz
# - Updated makefile: combined all online interfaces
# [from r1.15 by delta oz-21671-TPCC-merge-online-transaction-interfaces, r1.2]
#
# Revision 1.15  1998/01/23  15:07:39  oz
# - Updated the SP TPCC directory to the latest files used
#   during the SP tpcc audit.
# [from r1.14 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
#
#
#

# This must be defined if you wish to collecttimestamps for each transaction
# for creating data to use with "analyze".
COLLECT_TIMESTAMPS = -DCOLLECT_TIMESTAMPS

###  The following definitions are used to compile
###  Using the Transarc Standard (internal) environment
###  At Transarc

#ORACLE_HOME=/oracle/app/oracle/product/8.0.3/home
ORACLE_HOME=/home/oracle803/app/oracle/product/8.0.3


LDFLAGS=-H512 -T512
```

```
## ENC_DIR = /afs/tr/kansas/latest/internal
  ENC_DIR = /usr/lpp/encina


 ENC_LIB = -L${ENC_DIR}/lib -lEncMonCli -lEncClient -lEncina
 ENC_SLIB = -L${ENC_DIR}/lib -lEncMonServ -lEncServer -lEncClient -lEncina
 IDL_CMD = idl -no_cpp -no_mepv -keep c_source -cpp_opt "-P"
 TIDL_CMD = ${ENC_DIR}/bin/tidl -no_cpp
 DCE_LIB = -L/usr/lib -ldce -lm

INC =  -I. -I${ENC_DIR}/include -I./ora8MT_encMT

ORA_MT_OBJS_DIR = ./ora8MT_encMT
ORA_MT_OBJS = ${ORA_MT_OBJS_DIR}/c_trans.o ${ORA_MT_OBJS_DIR}/tpccpl.o  \
              callora.o ${ORACLE_HOME}/bench/gen/source/gettime.o


TWO_TASK_FLAGS= -bI:$(ORACLE_HOME)/lib/mili.exp
TWO_TASK_LIB= \
    -lnetwork -lserver -lcommon -lgeneric \
    -lnetwork -lserver -lcommon -lgeneric \
    -lnlsrtl3 -lcore4 -lnlsrtl3 -lcore4 \
    -lclntsh -lsqlplus -lodm \
    -lmm -lm -lld -lm-lepc -lncr

ORA_FLAGS =


  ORA_LIB = ${TWO_TASK_LIB}

##  To compile without ORACLE uncomment the following 3 lines
##  To compile with Oracle, make sure they are commented out
##   ORA_LIB =
##   ORA_MT_OBJS = callora.o
##   ORA_FLAGS = -DCOMPILE_WITHOUT_ORA


TIDL_INC = -I${ENC_DIR}/include -I.

DEBUG_ALL   = -DDEBUG_NEWO -DDEBUG_PMT -DDEBUG_ORDS -DDEBUG_DVRY
-DDEBUG_STKL \
              -DDEBUG_SQL

C_FLAGS=-Dunix -D_AIX41 -O -qmaxmem=10000 -bloadmap:load.map-D_AIX
C_D_FLAGS=

CFLAGS          = ${C_FLAGS} ${INC} ${ORA_FLAGS}
CLIENT_C_FLAGS = ${C_D_FLAGS} ${INC}


SERVER_OBJ  =           _tpcc_trans_sstub.o tpcc_trans_manager.o  \
                        _delivery_sstub.o delivery_manager.o

SERVER_SRC  =           _tpcc_trans_sstub.c tpcc_trans_manager.c  \
                        _delivery_sstub.c delivery_manager.c

CC = xlc_r4 -DTRACE_WITHOUT_TPP

CLIENT_OBJ = _tpcc_trans_cstub.o tpcc_trans_client.o tpcc_trans_cswtch.o  \
             _delivery_cstub.o delivery_client.o delivery_cswtch.o

CLIENT_SRC = _tpcc_trans_cstub.c tpcc_trans_client.c tpcc_trans_cswtch.c  \
             _delivery_cstub.c delivery_client.c delivery_cswtch.c

TIDL_HEADERS = tpcc_trans.h delivery.h _tpcc_trans.h _delivery.h


all:
        make tpcc_type.h
        cd screen; make
        make servers;
        make clients;
        make programs;

clients:    terminal tpcc_client

servers:    serverMT_stats

programs:     analyze tpcc_monitor

tpcc_client:   $(TIDL_HEADERS) tpcc_type.h                  \
        do_tpcc.o mon_client.o                                      \
                encina_client.o client_utils.o client_listen.o  \
                client_bg_thread.o                                  \
                $(CLIENT_OBJ)
           $(CC) $(CLIENT_C_FLAGS) -o tpcc_client                   \
        do_tpcc.o mon_client.o                                      \
                encina_client.o client_utils.o client_listen.o  \
                client_bg_thread.o                                  \
           $(CLIENT_OBJ) -L./screen -lClient                        \
           -L. ${ENC_LIB} ${DCE_LIB} -lC_r

terminal:   terminal.o emulator.o tran_stat.o client_utils.o
                $(CC) $(CLIENT_C_FLAGS) -o terminal \
                        terminal.o \
                        emulator.o \
                        tran_stat.o \
                        client_utils.o -L.${DCE_LIB} -lc_r


curses:             terminal.o client_utils.o curses.o
                    $(CC) $(CLIENT_C_FLAGS) -o curses                   \
                    terminal.o curses.o client_utils.o          \
                    -lcurses

emulator:   terminal.o emulator.o client_utils.o tran_stat.o
                    $(CC) $(CLIENT_C_FLAGS) -o emulator                 \
                    terminal.o emulator.o client_utils.o tran_stat.o \
                    -lm

tpcc_monitor:       tpcc_monitor.c tran_stat.c client_utils.c
                    $(CC) $(CLIENT_C_FLAGS) -o tpcc_monitor     \
                    tpcc_monitor.c client_utils.c tran_stat.c

analyze:    analyze_times.c
                    $(CC) -o analyze analyze_times.c


terminal.o:         terminal.c \
                        terminal.h

client_listen.o:    client_listen.c  \
                        tpcc_type.h client_utils.h

client_utils.o:             client_utils.c  \
                        databuf.h client_utils.h do_tpcc.h tpcc_type.h

do_tpcc.o:  do_tpcc.c  \
                        utilities.h client_utils.h do_tpcc.h \
                        client_listen.h
                        $(CC) $(CLIENT_C_FLAGS) -c do_tpcc.c

databuf.h:  $(TIDL_HEADERS) tpcc_type.h

client_utils.h:     tpcc_type.h

do_tpcc.h:  databuf.h

##  TIDL and IDL rules:

## tpcc_type.idl: common declarations
tpcc_type.h: tpcc_type.idl
        ${IDL_CMD} tpcc_type.idl

##  neworder.tidl:  The TIDL interface for the monitor client and server
tpcc_trans.h tpcc_trans_client.c tpcc_trans_manager.c \
        tpcc_trans_cswtch.c _tpcc_trans.idl: tpcc_trans.tidl tpcc_type.h
        ${TIDL_CMD} tpcc_trans.tidl ${TIDL_INC}

_tpcc_trans.h _tpcc_trans_cstub.c _tpcc_trans_sstub.c: _tpcc_trans.idl
        ${IDL_CMD} ${TIDL_INC} _tpcc_trans.idl

_tpcc_trans_cstub.o: _tpcc_trans_cstub.c
        $(CC) $(CLIENT_C_FLAGS) -c _tpcc_trans_cstub.c

tpcc_trans_cswtch.o: tpcc_trans_cswtch.c
        $(CC) $(CLIENT_C_FLAGS) -c tpcc_trans_cswtch.c

tpcc_trans_client.o: tpcc_trans_client.c
        $(CC) $(CLIENT_C_FLAGS) -c tpcc_trans_client.c

##  delivery.tidl:  The TIDL interface for the monitor client and server
delivery.h delivery_client.c delivery_manager.c \
        delivery_cswtch.c _delivery.idl: delivery.tidl tpcc_type.h
        ${TIDL_CMD} delivery.tidl ${TIDL_INC}

_delivery.h _delivery_cstub.c _delivery_sstub.c: _delivery.idl
        ${IDL_CMD} ${TIDL_INC} _delivery.idl

_delivery_cstub.o: _delivery_cstub.c
        $(CC) $(CLIENT_C_FLAGS) -c _delivery_cstub.c

delivery_cswtch.o: delivery_cswtch.c
        $(CC) $(CLIENT_C_FLAGS) -c delivery_cswtch.c

delivery_client.o: delivery_client.c
        $(CC) $(CLIENT_C_FLAGS) -c delivery_client.c

serverMT_stats:   $(TIDL_HEADERS) tpcc_type.h \
        serverMon.stats.o server.o  \
        $(SERVER_OBJ) ${ORA_MT_OBJS}
        $(CC) -o serverMT_stats \
        -L$(ORACLE_HOME)/lib -L$(ORACLE_HOME)/rdbms/lib \
        serverMon.stats.o server.o  \
        $(SERVER_OBJ) ${ORA_MT_OBJS) \
        ${ORA_LIB} ${ENC_SLIB} ${DCE_LIB}

tpccMT_server:    $(TIDL_HEADERS) tpcc_type.h \
        serverMon.o server.o  \
        $(SERVER_OBJ) ${ORA_MT_OBJS}
```

```
	$(CC) -o tpccMT_server \
	-L$(ORACLE_HOME)/lib -L$(ORACLE_HOME)/rdbms/lib \
	serverMon.o server.o  \
	$(SERVER_OBJ) $(ORA_MT_OBJS) \
	${ORA_LIB} ${ENC_SLIB} ${DCE_LIB}

server.o:	server.c do_tpcc.h $(SERVER_OBJ) $(TIDL_HEADERS)
		$(CC) $(CFLAGS) -c server.c -o server.o

serverMonLocal.stats.c:  serverMonLocal.c
		ln -sf serverMonLocal.c serverMonLocal.stats.c

serverMonLocal.stats.o:  serverMonLocal.stats.c do_tpcc.h \
		$(SERVER_XA_OBJ) $(TIDL_HEADERS)
		$(CC) $(CFLAGS) -DCOLLECT_TIMESTAMPS -c
serverMonLocal.stats.c

serverMon.stats.c:	serverMon.c
		ln -sf serverMon.c serverMon.stats.c

serverMon.stats.o:	serverMon.stats.c do_tpcc.h \
				$(SERVER_OBJ) $(TIDL_HEADERS)
		$(CC) $(CFLAGS) -DCOLLECT_TIMESTAMPS -c
serverMon.stats.c

serverEncina.o:		serverEncina.c utilities.h
		$(CC) $(CFLAGS) -c serverEncina.c

trans_generator:	trans_generator.c bmd.h
				$(CC) -O trans_generator.c -o trans_generator -lm

clean:
	cd screen; make clean
	rm -f *.o
	rm -f $(CLIENT_SRC) $(SERVER_SRC) $(SERVER_XA_SRC)
$(TIDL_HEADERS)
	rm -f $(CLIENT_REMOTE_SRC) $(SERVER_REMOTE_SRC)
	rm -f tpcc_tk.h tpcc_tk_client.c tpcc_tk_cswtch.c tpcc_tk_manager.c
	rm -f _tpcc_trans* _delivery*
	rm -f tpcc_type.h
	rm -f delivery.c tpcc_trans.c
	rm -f server tpcc_client
	rm -f serverMT_stats tpccMT_server
	rm -f liberty curses emulator
	rm -f analyze terminal
```

mon_client.c

```
/*
 *		mon_client.c
 *
 * $Revision: 1.22 $
 * $Date: 1998/02/17 22:13:49 $
 * $Log:	$
 *
 *
 * $TALog: mon_client.c,v $
 * Revision 1.22  1998/02/17  22:13:49   wenjian
 * [merge of changes from 1.19 to 1.20 into 1.21]
 *
 * Revision 1.20  1998/02/17  16:04:42  oz
 * - Split the login into two parts to allow for speciallogins
 * - If the warehouse ID is 0, this is a special login to
 *   query the client for status
 *
 * - Keep track of threads that have been initialized and also
 *   threads that are done.
 * [from r1.19 by delta oz-21864-TPCC-split-client-login-screen, r1.1]
 *
 * Revision 1.21  1998/02/17  22:07:03   wenjian
 * Minor changes for NT
 * [from r1.19 by delta wenjian-21750-TPCC-changes-for-porting-on-NT,r1.1]
 *
 * Revision 1.19  1998/01/26  20:37:35  oz
 * - Remove all the code associated with explicit binding
 *
 * - Removed GET_SERVER_INDEX
 * - Removed bindingType
 * - Removed explicit binding from CALLTPCC
 * - Removed calls to cancel_all_reservations and to init_handles
 * [from r1.18 by delta oz-21697-TPCC-remove-explicit-binding-code, r1.1]
 *
 * Revision 1.18  1998/01/26  16:43:32  oz
 * - Removed the code for collectingstats in the client
 *   and dumping them before exit.
 *
 * - Removed pre_rpc_stats and post_rpc_stats
 * - Removed code to write thestats out
 * [from r1.17 by delta oz-21691-TPCC-remove-client-stats-code, r1.1]
 *
 * Revision 1.17  1998/01/26  16:19:23  oz
 * - moved all the code pertaining to the background
 *   thread to its own file and all the data structures
 *   to client_utils.h
 * [from r1.16 by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file, r1.1]
 *
 * Revision 1.16  1998/01/26  15:33:32  oz
 * - call impTPCCNOInfo to make sure there is a server out there
 * [from r1.15 by delta oz-21671-TPCC-merge-online-transaction-interfaces, r1.2]
 *
 * Revision 1.15  1998/01/23  21:58:51  oz
 * - In order to simplify the Encina TPCC code: Merge the four
 *   online transactions into 1 interface
 * - Moved all the scripts to a scripts subdirectory
 * - Removed unused files
 * [from r1.14 by delta oz-21671-TPCC-merge-online-transaction-interfaces, r1.1]
 *
 * Revision 1.14  1998/01/23  15:07:53  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.13 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 *
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <time.h>
#include <sys/time.h>
#if defined (solaris)
#include <dce/pthread.h>
#else  /* solaris */
#include <pthread.h>
#endif  /* solaris */
#include <tpm/adl/adl.h>
#include <tpm/mon/mon.h>
#include <utils/trace.h>
#ifdef MULTIPLE_INTERFACE
#include "neworder.h"
#include "payment.h"
#include "stocklevel.h"
#include "orderstatus.h"
#else
#include "tpcc_trans.h"
#endif
#include "delivery.h"
#include "utilities.h"
#include "client_utils.h"
#include "do_tpcc.h"
#include "client.h"
#include "encina_client.h"

#if 0
#define SKIP_RPC
#endif

extern void start_bg_debug_thread(void);

#define MAX_CONSECUTIVE_ERRORS 20

static void read_mon_environment(void);
static void client_trace(char *comp, int value, int add);
static void dump_pa_ring_buffer(trpc_handle_t pa_handle);

extern int warehouse_offset;

adl_authnLevel_t		client_authnLevel;
adl_authzLevel_t		client_authzSvc;

char *cellName;
int envRetrieval = 0;

static total_tran_count_t total_counts;  /* counts of transactions over
                                                                        * the entire test
                                                                        */

MUTEX_T init_lock;
int info_list_len = 0;
thread_info_t **info_list = NULL;  /*  List of all the thread info
                                                                * structures.  This can be used
                                                                *  upon exit to cancel all the
                                                                *  reservations
                                                                */
static num_active_threads = 0;

#define NewOrder_code    NEWO_TRANS
#define Payment_code     PAYMENT_TRANS
#define OrderStatus_code ORDER_STAT_TRANS
#define Delivery_code    DELIVERY_TRANS
#define StockLevel_code  STOCK_TRANS

extern int useSecurity;

#define PRE_RPC_WORK(contextP, dataP, tran, sub_tran) \
                        pre_rpc(contextP, &(dataP)->header, tran, sub_tran)
#define POST_RPC_WORK(contextP, dataP, tran) \
                        post_rpc(contextP, &(dataP)->header, tran)
#define TIME_STR_P(infoP)  (&((infoP)->last_tran))
```

```
/*                  CALTPCC
 *  Macro to sends 1 RPC and then handles any errors.
 *
 *  The macro takes the name of the RPC (e.g., NewOrder)
 *  and makes the RPC by calling the appropriate function
 *  (e.g., impTPCCNewOrder).
 */
#ifdef SKIP_RPC
#define CALLTPCC(name,infoP,data,trpcStatusP)                          \
{                                                                      \
 struct timezone tz;                                                   \
 struct timespec timeP;                                                \
 char   tran_type[30];                                                 \
 strcpy(tran_type,UTIL_STRING(name));                                  \
 timeP.tv_sec = 0;                                                     \
 timeP.tv_nsec = 190000000;                                            \
 if ( strcmp(tran_type,"NewOrder")==0 )                                \
    timeP.tv_nsec = 450000000;                                         \
 if ( strcmp(tran_type,"Payment")==0 )                                 \
    timeP.tv_nsec = 90000000;                                          \
 pthread_delay_np(&timeP);                                             \
 gettimeofday(&TIME_STR_P(infoP)->send, &tz);                          \
}
#else
#define CALLTPCC(name,infoP,data,trpcStatusP)                          \
{                                                                      \
 struct timezone tz;
\
 gettimeofday(&TIME_STR_P(infoP)->send, &tz);                          \
 UTIL_CONCAT(impTPCC,name)(data,trpcStatusP);                          \
 if (*(trpcStatusP)) {                                                 \
   char msg[100];                                                      \
   sprintf(msg, "TRPC error during impTPCC%s", UTIL_STRING(name));     \
   (data)->header.returncode= TRPC_ERROR;                              \
   encina_error_message(msg, *(trpcStatusP));                          \
 } else if (((data)->header.returncode!= TPCC_SUCCESS) &&              \
          ((data)->header.returncode!= INVALID_NEWO)) {                \
   char msg[100];                                                      \
   sprintf(msg, "App error during impTPCC%s: ", UTIL_STRING(name));    \
   encina_error_message(msg, (data)->header.returncode);               \
 }
   \
}
#endif

/*
 *  pre_rpc --   For debug purposes
 *
 *  Called before an RPC is made.
 *  Set the state of the thread and keep track of the time the RPC is sent.
 *  This is used by the Background thread to report the state of the client.
 */
static void pre_rpc(thread_info_t *thread_infoP,
                    data_header *headerP,
                    int tran_type,
                    int sub_tran_type)
{
   tran_timing_t *curP;
   struct timezone tz;

   curP = &thread_infoP->last_tran;
   curP->terminal = thread_infoP->thread_index;
   curP->tran = tran_type;
   curP->sub_tran = sub_tran_type;

   gettimeofday(&curP->start, &tz);
   headerP->start_time.sec = 0;
   headerP->start_time.usec = 0;
   headerP->end_time.sec = 0;
   headerP->end_time.usec = 0;

   set_client_debug_state((void *)thread_infoP, thread_state_sent, tran_type);
}

/*
 *  post_rpc
 *
 *  Called when the RPC returns from the server
 *
 *  Keeps track of the client response time and the server response time
 *  as well as the state of the thread.  This is used by the background
 *  debug thread to report the state of the client
 */
static void post_rpc(thread_info_t *thread_infoP,
                                    data_header *headerP,
                                    int tran_type)
{
   double time_diff_s, time_diff_c;
   tran_timing_t *curP;
   struct timezone tz;

   curP = &thread_infoP->last_tran;

   curP->server = headerP->dtype;  /*  The server sets this by convention */
   curP->srvr_start.tv_sec = headerP->start_time.sec;
   curP->srvr_start.tv_usec = headerP->start_time.usec;
   curP->srvr_done.tv_sec = headerP->end_time.sec;
   curP->srvr_done.tv_usec = headerP->end_time.usec;
   gettimeofday(&curP->end, &tz);

   thread_infoP->num_trans++;
   if ((headerP->returncode == TPCC_SUCCESS)||
      (headerP->returncode == INVALID_NEWO)) {
    thread_infoP->consecutive_errors = 0;
    thread_infoP->tran[tran_type].num++;
    curP->tran_failed = 0;
    if (headerP->returncode == INVALID_NEWO) {
      curP->sub_tran |= 0x100;
    }
   } else {
    thread_infoP->tran[tran_type].errs++;
    thread_infoP->consecutive_errors++;
    curP->tran_failed = 1;
   }

   set_client_debug_state((void *)thread_infoP, thread_state_received, 0);

   if (tran_type <= MAX_TRAN_TYPE && tran_type > 0) {
     /* update total server round trip response time */
     time_diff_s = time_diff_ms(&(curP->srvr_done), &(curP->srvr_start));
     thread_infoP->tran[tran_type].RT[1]+= time_diff_s;

     /* update total client round trip response time*/
     time_diff_c = time_diff_ms(&(curP->end), &(curP->start));
     thread_infoP->tran[tran_type].RT[0]+= time_diff_c;
   } else {
                err_printf("Wrong tran_type %d\n", tran_type);
   }
}

/*
 * exit_program - restores original terminal attributes before leaving the
 *              program.
 */

void exit_program( err )
  short int err;
{
   if ( err )
                fprintf( stderr, "exit_program: Error Code = %d\n", err );

   MUTEX_LOCK(&init_lock);
   /**  Cancel all the longterm reservations (if any)
    **              and write out the time-stamps
    **/
   if (info_list && (info_list_len > 0)) {
    int i;

    for (i=0; i<info_list_len; i++) {

                if (info_list[i] && info_list[i]->initialized) {
                  info_list[i]->initialized = 0;
                }
    }
   }

   MUTEX_UNLOCK(&init_lock);

   if (logtpcc) {
    fclose(logtpcc);
   } else {
    if (logtpcc = fopen(log_file_name, "w")) {
                fprintf(logtpcc, "ERROR: Client exiting before SYNC with error %d\n",
                                err);
                fclose(logtpcc);
    }
   }

   mon_ExitClient( err );

   exit( err );
}

/*
 *  clnt_thread_init
 *
 *  This function must be called by each work thread
 *  It returns a pointer to a context that must be passed
 *  on calls back to this module.
 *  There is 1 threadInfo entry in an array for each executor thread.
 *  When an executor thread is started the first thing it does is call
 *  this thread_init function.  This function creates a context for the
 *  thread and if longterm reservations are used this function
 *  initializes the pa handle.
 */
void *clnt_thread_init(void)
{
   int thread_index;
   struct timezone tz;
   thread_info_t *thread_infoP;

   thread_infoP = (thread_info_t *)calloc(1, sizeof(thread_info_t));

   thread_infoP->descr.state = thread_state_init;
   gettimeofday(&thread_infoP->descr.init, &tz);
   thread_infoP->initialized = 1;
```

```
    MUTEX_LOCK(&init_lock);
    thread_index = info_list_len++;
    thread_infoP->thread_index = thread_index;
    thread_infoP->thread_id = get_thread_id();

    num_active_threads++;
    info_list =
                    (thread_info_t **)realloc((void *)info_list,
                                                    sizeof(thread_info_t *) * info_list_len);
    info_list[thread_index] = thread_infoP;

    MUTEX_UNLOCK(&init_lock);

    if (num_active_threads % 25 == 0)
       err_printf("Thread %d Initialized (currently %d areactive).\n",
                           thread_index, num_active_threads);


    return(thread_infoP);
}

/*
 *  thread_done
 *
 *  Called before a thread exits.
 *  Perform some cleanup.
 *
 */
void thread_done(contextP)
  void *contextP;
{
    int all_done = 0;
    int j;
    thread_info_t *infoP = (thread_info_t *)contextP;



    MUTEX_LOCK(&init_lock);

    num_active_threads--;

    err_printf("> thread_done, %d active\n", num_active_threads);

    set_client_debug_state((void *)infoP, thread_state_done, 0);
    infoP->done = 1;

    if (num_active_threads == 0) {
      all_done = 1;
    }

    if (info_list[infoP->thread_index] != infoP) {
      fprintf(stderr, "Strange error: expected to find %d ininfo_list[%d] and found %d instead\n",
                        infoP, infoP->thread_index,
                        info_list[infoP->thread_index]);
    }

    MUTEX_UNLOCK(&init_lock);
    if (all_done) {
                    int i;
                    thread_info_t **curP;
                    fprintf(stderr, "All Done - exiting\n");
                    MUTEX_LOCK(&init_lock);
                    for (i=0, curP=info_list; i<info_list_len; i++, curP++) {
                       free(*curP);
                    }
                    free(info_list);
                    info_list = NULL;
                    info_list_len = 0;
                    MUTEX_UNLOCK(&init_lock);
#if 1
                    exit(0);
#endif
    }
}

/*
 *  The following send_*** functions are called from the screen
 *  module after the transaction data is received in order to
 *  send the data to the server for processing.
 *
 */

/*
 *  send_new_order
 *                  Send a new order request to the server
 */
void send_new_order(contextP, dataP)
  void *contextP;
  newOrder_data_t *dataP;
{
    thread_info_t *thread_context = (thread_info_t *)contextP;
    trpc_status_t trpcStatus;

    DPRINT(("New Order, w_id %d, %d orders\n", dataP->w_id, dataP->o_ol_cnt));
    PRE_RPC_WORK(thread_context, dataP, NEWO_TRANS, dataP->o_all_local == 0);
    CALLTPCC(NewOrder,thread_context,dataP,&trpcStatus)
    POST_RPC_WORK(thread_context, dataP, NEWO_TRANS);
}
```

```
/*
 *  send_payment
 *                  Send a payment request to the server
 */
void send_payment(contextP, dataP)
  void *contextP;
  payment_data_t *dataP;
{
    trpc_status_t trpcStatus;
    thread_info_t *thread_context = (thread_info_t *)contextP;
    int w_id = dataP->w_id;
    int c_w_id = dataP->c_w_id;

    PRE_RPC_WORK(thread_context, dataP, PAYMENT_TRANS,
                                    dataP->w_id != dataP->c_w_id);
    CALLTPCC(Payment,thread_context,dataP,&trpcStatus);
    POST_RPC_WORK(thread_context, dataP, PAYMENT_TRANS);
}

/*
 *  send_order_status
 *                  Send a order status request to the server
 */
void send_order_status(contextP, dataP)
  void *contextP;
  orderStatus_data_t *dataP;
{
    trpc_status_t trpcStatus;
    thread_info_t *thread_context = (thread_info_t *)contextP;

    PRE_RPC_WORK(thread_context, dataP, ORDER_STAT_TRANS, 0);
    CALLTPCC(OrderStatus,thread_context,dataP,&trpcStatus);
    POST_RPC_WORK(thread_context, dataP, ORDER_STAT_TRANS);
}

/*
 *  send_delivery
 *                  Send a delivery request to the server
 */
void send_delivery(contextP, dataP)
  void *contextP;
  delivery_data_t *dataP;
{
    trpc_status_t trpcStatus;
    thread_info_t *thread_context = (thread_info_t *)contextP;

    PRE_RPC_WORK(thread_context, dataP, DELIVERY_TRANS, 0);
    CALLTPCC(Delivery,thread_context,dataP,&trpcStatus);
    POST_RPC_WORK(thread_context, dataP, DELIVERY_TRANS);
}

/*
 *  send_stock_level
 *                  Send a stock level request to the server
 */
void send_stock_level(contextP, dataP)
  void *contextP;
  stockLevel_data_t *dataP;
{
    trpc_status_t trpcStatus;
    thread_info_t *thread_context = (thread_info_t *)contextP;

    PRE_RPC_WORK(thread_context, dataP, STOCK_TRANS, 0);
    CALLTPCC(StockLevel,thread_context,dataP,&trpcStatus);
    POST_RPC_WORK(thread_context, dataP, STOCK_TRANS);
}

int too_many_errors(contextP)
  void *contextP;
{
  thread_info_t *thread_context = (thread_info_t *)contextP;

  return (thread_context->consecutive_errors > MAX_CONSECUTIVE_ERRORS);




 *  Enroll the client:
 *                  Perform the needed initialization and get the necessary
 *                  handles.
 */
void enroll_client(user_id)
  int user_id;
{
    int i, server_id;
    mon_status_t                monStatus;
    char *env_str;
    char serverName[48];
    static  char  *clientName="tpcc_client";
    struct timezone tz;
    struct timeval a_time;

    read_mon_environment();

    MUTEX_INIT(&init_lock);
    info_list = NULL;
    info_list_len = 0;

    gettimeofday(&a_time, &tz);
```

```
#ifdef WIN32
   srand(a_time.tv_sec ^ a_time.tv_usec);
#else
   srand48(a_time.tv_sec ^ a_time.tv_usec);
#endif
   if (useSecurity) {
                client_authnLevel = ADL_AUTHN_CONNECT;
                client_authzSvc = ADL_AUTHZ_DCE;
   } else {
                client_authnLevel = ADL_AUTHN_NONE;
                client_authzSvc = ADL_AUTHZ_NONE;
   }

   if (envRetrieval == 0) mon_RetrieveEnable(FALSE);

   ENCINA_CALL("mon_InitClient",mon_InitClient(clientName,cellName));

   DPRINT(("mon_SecuritySetDefaults-> authn %d, authz %d\n",
                client_authnLevel, client_authzSvc));
   ENCINA_CALL("mon_SecuritySetDefaults",
                mon_SecuritySetDefaults(client_authnLevel,client_authzSvc));
   ENCINA_CALL("mon_SetHandleCacheRefreshInterval",
                mon_SetHandleCacheRefreshInterval(300));

   {
      dbInfo_data_t data;
      trpc_status_t trpcStatus;
      /* Get DB Info -- currently id does not do anything
         but it will tell us if there is a server out there.
         Better to know instead of when all the terminals
         are up and ready
       */
      impTPCCNOInfo(&data,&trpcStatus);
      if (trpcStatus) {
         char msg[100];
         sprintf(msg, "TRPC error during db info at init.");
         encina_error_message(msg, trpcStatus);
         exit(33);
      }
   }

   start_bg_debug_thread();
}

/*-----------------------------------------------------------------*/
/*      Read environment paramaters                                */
/*-----------------------------------------------------------------*/
static void read_mon_environment()
{
   char *env_str;

   cellName = getenv("ENCINA_TPM_CELL");
   CHECK_ENVIRON(cellName, "ENCINA_TPM_CELL");

   if (env_str = getenv("TPCC_ENV_RETRIEVE")) {
                envRetrieval = atoi(env_str);
   }
}

/*
 *  dump_pa_ring_buffer() -- For Debugging --
 *      Dump the ring buffer in the PA we are talking to
 *      Only works if we are using long term reservation
 */
static void dump_pa_ring_buffer(pa_handle)
  trpc_handle_t pa_handle;
{
  err_printf("Dumping Ring Buffer of server\n");
  admin_trace_DumpRingBuffer((handle_t)pa_handle, "stderr");
}
```

Neworder.h

```
#ifndef TRANSARC_neworder_h
#define TRANSARC_neworder_h

#include <trpc/trpc.h>
#include "_neworder.h"

#include <encina/c_prologue.h>

#define neworder_v1_0_c_ifspec          _neworder_v1_0_c_ifspec
#define neworder_v1_0_s_ifspec          _neworder_v1_0_s_ifspec

typedef struct neworder_v1_0_epv {
void (ENCINA_DCE_FUNCTION_PTR_CALLING *impTPCCNewOrder)(
#ifdef IDL_PROTOTYPES

                newOrder_data_t *dataP,
                trpc_status_t *trpcStatus
#endif
);

void (ENCINA_DCE_FUNCTION_PTR_CALLING *impTPCCNOInfo)(
#ifdef IDL_PROTOTYPES

                dbInfo_data_t *dataP,
                trpc_status_t *trpcStatus
#endif
);

} neworder_v1_0_epv_t;

extern void impTPCCNewOrder(
#ifdef IDL_PROTOTYPES

                newOrder_data_t *dataP,
                trpc_status_t *trpcStatus
#endif
);

extern void impTPCCNOInfo(
#ifdef IDL_PROTOTYPES

                dbInfo_data_t *dataP,
                trpc_status_t *trpcStatus
#endif
);

trpc_handle_tmon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t            handle,
                trpc_tranInfo_t         *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

void            mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t            handle,
                trpc_handle_ttrpcHandle,
                trpc_tranInfo_t         *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

trpc_handle_tmon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t            handle,
                trpc_tranInfo_t         *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

void            mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t            handle,
                trpc_handle_ttrpcHandle,
                trpc_tranInfo_t         *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

extern neworder_v1_0_epv_t              neworder_v1_0_client_epv;
extern _neworder_v1_0_epv_t             neworder_v1_0_manager_epv;
extern rpc_mgr_epv_t                    neworder_v1_0_mgr_epv;

#include <encina/c_epilogue.h>
#endif /* TRANSARC_neworder_h */
```

orderstatus.h

```
#ifndef TRANSARC_orderstatus_h
#define TRANSARC_orderstatus_h

#include <trpc/trpc.h>
#include "_orderstatus.h"

#include <encina/c_prologue.h>

#define orderstatus_v1_0_c_ifspec       _orderstatus_v1_0_c_ifspec
#define orderstatus_v1_0_s_ifspec       _orderstatus_v1_0_s_ifspec

typedef struct orderstatus_v1_0_epv {
void (ENCINA_DCE_FUNCTION_PTR_CALLING *impTPCCOrderStatus)(
#ifdef IDL_PROTOTYPES

                orderStatus_data_t *dataP,
                trpc_status_t *trpcStatus
#endif
);

} orderstatus_v1_0_epv_t;

extern void impTPCCOrderStatus(
#ifdef IDL_PROTOTYPES

                orderStatus_data_t *dataP,
                trpc_status_t *trpcStatus
#endif
);
```

```
trpc_handle_tmon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

void            mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_handle_ttrpcHandle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

trpc_handle_tmon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

void            mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_handle_ttrpcHandle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

extern orderstatus_v1_0_epv_t               orderstatus_v1_0_client_epv;
extern _orderstatus_v1_0_epv_t              orderstatus_v1_0_manager_epv;
extern rpc_mgr_epv_t                        orderstatus_v1_0_mgr_epv;

#include <encina/c_epilogue.h>
#endif /* TRANSARC_orderstatus_h */
```

Payment.h

```
#ifndef TRANSARC_payment_h
#define TRANSARC_payment_h

#include <trpc/trpc.h>
#include "_payment.h"

#include <encina/c_prologue.h>

#define payment_v1_0_c_ifspec               _payment_v1_0_c_ifspec
#define payment_v1_0_s_ifspec               _payment_v1_0_s_ifspec

typedef struct payment_v1_0_epv {
void (ENCINA_DCE_FUNCTION_PTR_CALLING *impTPCCPayment) (
#ifdef IDL_PROTOTYPES

                                payment_data_t *dataP,
                                trpc_status_t *trpcStatus
#endif
);

} payment_v1_0_epv_t;

extern void impTPCCPayment (
#ifdef IDL_PROTOTYPES

                                payment_data_t *dataP,
                                trpc_status_t *trpcStatus
#endif
);

trpc_handle_tmon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

void            mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_handle_ttrpcHandle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

trpc_handle_tmon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

void            mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_handle_ttrpcHandle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

extern payment_v1_0_epv_t payment_v1_0_client_epv;
extern _payment_v1_0_epv_t                  payment_v1_0_manager_epv;
extern rpc_mgr_epv_t                        payment_v1_0_mgr_epv;

#include <encina/c_epilogue.h>
#endif /* TRANSARC_payment_h */
```

Screen.C

```
/* (C)1997 IBM Corporation */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

#include "screen.h"
#include "format.h"
#include "encina.h"

#define USE_INSULTS
#define LOCAL_SESSION_DATA

extern "C" err_printf(...);

extern char const * const blanks;
extern char const * const underscores;
extern char const * const backspaces;

static int clear_eos(InOut *ioP);
static int clear_eos(char *buf);
static int string_empty(char const *text);
static int pos_zero(int const *val);
static int pos_nonzeros(int const **val);

/*****************************************************************************
Screen
*****************************************************************************/
int Screen::reset() {
   has_data=0;
   pos=0;
   if (dataptr) memset(dataptr, 0, data_len);
   for (int i = 0; fields && fields[i] != NULL; i++) {
                fields[i]->reset();
   }
   return 0;
};
int Screen::present_empty_fields() {
   if (empty_fields)
                threadP->write(empty_fields, empty_fields_len);
//   threadP->write(end_str, end_str_len);
   return 0;
}
int Screen::present() {
   threadP->write(screen, screen_len);
   threadP->write(session_data, session_data_len);
   if (has_data) {
                for (int i = 0; fields[i] != NULL; i++) {
                   fields[i]->display_field(1);
                }
//              threadP->write(end_str, end_str_len);
   } else {
                present_empty_fields();
   }
   return 0;
};
int Screen::user_input() {
   int key;
   has_data = 1;
   fields[pos]->start_position();
   threadP->flush();
//   threadP->mark();
   key = fields[pos]->get_field(0);
   do {
                switch (key) {
                case EOF:
```

```
                return 0;
                break;
            case Field::NEXT_FIELD:
                if (fields[++pos] == NULL) {
                        pos = 0;
                }
                break;
            case Field::PREV_FIELD:
                if (--pos < 0) {
                        while (fields[++pos] != NULL);
                        pos--;
                }
                break;
            case Field::REDISPLAY:
                present();
                break;
            case Field::ABORT:
//              position(1, 2);
                threadP->write(end_str, end_str_len);
                return 0;
            case Field::ENTER:
                if (validate()) {
//                      threadP->pop();
                        return 1;
                }
                break;
            }
            key = fields[pos]->get_field();
    } while (1);
    return 0;
}
Screen::~Screen() {
    if (fields != NULL) {
            for (int lpos = 0; fields[lpos] != NULL; lpos++) {
                delete fields[lpos];
            }
            delete [] fields;
    }
    fields=NULL;
}

int Screen::display_status(int status) {
    position(threadP, status_x, status_y);
    threadP->write("Execution Status: ");
    if (status == TRAN_OK) {
            threadP->write("Transaction Committed");
    } else if (status == INVALID_ITEM) {
            threadP->write("Item number is not valid");
    } else {
            threadP->write("ERROR: Rollback -- ");
//          threadP->write("Rollback -- ");
            char buf[6];
            format_int(buf, 5, status);
            threadP->write(buf, 5);
    }
    return 0;
}

int Screen::handle() {
    threadP->debug("%s - reset\n", tran_type);
    reset();

    threadP->debug("%s - present\n", tran_type);
    threadP->hold();
    present();
    threadP->write(TRIGGER, 1);
    threadP->debug("%s - user_input\n", tran_type);
    if (!user_input()) {
            threadP->write(end_str, end_str_len);
            threadP->write(TRIGGER, 1);
            return -1;
    }
    threadP->flush();
    threadP->hold();
    threadP->debug("%s - process\n", tran_type);
    if (process()) {
            threadP->write(end_str, end_str_len);
            threadP->write(TRIGGER, 1);
            return -1;
    }
    threadP->debug("%s - respond\n", tran_type);
    respond();
    //  position(threadP, 1, 2);
    threadP->write(end_str, end_str_len);
    threadP->write(TRIGGER, 1);
    threadP->flush();
    return 0;
}

/*****************************************************************************
NewOrder
*****************************************************************************/
int NewOrder::reset() {
    Screen::reset();
    pos=start_field;
    memset(dataptr, 0, sizeof(*data));
    return 0;
};
NewOrder::NewOrder(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
    tran_type = NEWORDER_SERVICE;
    dataptr = data = new NewOrder_data;
    data_len = sizeof(NewOrder_data);


    status_x = 1;
    status_y = 24;

    screen          = static_screen;
    empty_fields    = static_empty_fields;
#ifdef LOCAL_SESSION_DATA
    session_data = new char[static_session_data_len+1];
    sprintf(session_data, "%s%5d", POS(12,4), user_dataP->warehouse);
#else
    session_data    = static_session_data;
    sprintf(session_data, "%s%5d", POS(12,4), user_dataP->warehouse);
#endif
    screen_len      = static_screen_len;
    empty_fields_len = static_empty_fields_len;
    session_data_len = static_session_data_len;

    int lpos = 0;
    fields = new Field *[2+MAX_ITEMS*3+1];
    for (int i = 0; i < MAX_ITEMS; i++) {
                fields[lpos++] = genfield(threadP,  3, 9+i, 5, &data->item[i].s_OL_SUPPLY_W_ID);
                fields[lpos++] = genfield(threadP, 10, 9+i, 6, &data->item[i].s_OL_I_ID);
                fields[lpos++] = genfield(threadP, 45, 9+i, 2, &data->item[i].s_OL_QUANTITY);
#if defined(USE_SMART_FIELDS)
                if (i > 0) {
                    int **tmp = new int *[4];
                    tmp[0] = &fields[lpos-6]->pos;
                    tmp[1] = &fields[lpos-5]->pos;
                    tmp[2] = &fields[lpos-4]->pos;
                    tmp[3] = NULL;
                    fields[pos-3]->ok_func = (int(*)(void*))pos_nonzeros;
                    fields[pos-3]->ok_data = tmp;
                    fields[pos-2]->ok_func = (int(*)(void*))pos_nonzeros;
                    fields[pos-2]->ok_data = tmp;
                    fields[pos-1]->ok_func = (int(*)(void*))pos_nonzeros;
                    fields[pos-1]->ok_data = tmp;
                }
#endif
    }
    start_field = lpos;
    fields[lpos++] = genfield(threadP, 29, 4, 4, &data->s_D_ID); /* District */
    fields[lpos++] = genfield(threadP, 12, 5, 4, &data->s_C_ID); /* Customer */
    fields[lpos++] = NULL;
    reset();
};

int NewOrder::validate() {
    if (!fields[start_field]->pos) {
                pos=start_field;
                message(threadP, "District ID is a required field");
                return 0;
    }
    if (!fields[start_field+1]->pos) {
                pos=start_field+1;
                message(threadP, "Customer ID is a required field");
                return 0;
    }

    int last=-1;
    data->s_O_OL_CNT = 0;
    data->s_all_local = 1;
    data->s_W_ID = user_dataP->warehouse;
    for (int i = 0; i < MAX_ITEMS*3; i+=3 ) {
        if (fields[i]->pos || fields[i+1]->pos || fields[i+2]->pos){
            if (!fields[i]->pos) {
                pos=i;
#if defined(USE_INSULTS)
                message(threadP, "Yeah, I think this is a bogus field too.");
#else
                message(threadP, "Warehouse ID is a required field");
#endif
                return 0;
            }
            if (!fields[i+1]->pos) {
                pos=i+1;
#if defined(USE_INSULTS)
                message(threadP, "Umm, WHAT did you want?");
#else
                message(threadP, "Item ID is a required field");
#endif
                return 0;
            }
            if (data->item[i/3].s_OL_QUANTITY <= 0) {
                pos=i+2;
#if defined(USE_INSULTS)
                message(threadP, "So something plus nothing is...");
#else
                message(threadP, "Please enter a quantity greater than 0");
#endif
                return 0;
            }
            if (data->item[i/3].s_OL_SUPPLY_W_ID != data->s_W_ID) {
                data->s_all_local=0;
            }
            data->s_O_OL_CNT++;
```

```
    } else if (last < 0) {
      last = i;
    }
  }
  if (data->s_O_OL_CNT <= 0) {
    pos=0;
#if defined(USE_INSULTS)
    message(threadP, "It's kind of pointless without ordering something isn't it?");
#else
    message(threadP, "Please enter an item to order");
#endif
    return 0;

  }
  // Compress the order lines:  some of them may be empty
  int ind;
  for (i=0, ind=0; ind<data->s_O_OL_CNT ; i++ ) {
    if (fields[i*3]->pos) {
      if (i > ind) {
        data->item[ind] = data->item[i];
      }
      ind ++;
    }
  }
  if (i > ind) {
    int j;
    for (j=ind; j<i; j++) {
      /*  At least one empty line was skipped  */
      data->item[j].s_OL_SUPPLY_W_ID = 0;
      data->item[j].s_OL_I_ID = 0;
      data->item[j].s_OL_QUANTITY = 0;
    }
  }

  return 1;
}

int NewOrder::respond() {
  int i;
  double amount, total_amount, cost;
  char buf[32];
  position(threadP,  1, 9); clear_eos(threadP);
  position(threadP, 25, 5); threadP->write(data->s_C_LAST);
  position(threadP, 52, 5); threadP->write(data->s_C_CREDIT);
  position(threadP, 15, 6); format_int(buf, 9, data->s_O_ID); threadP->write(buf, 8);

  display_status(data->s_transtatus);
  if (data->s_transtatus != TRAN_OK) {
            return -1;
  }

  position(threadP, 25, 5); threadP->write( data->s_C_LAST);
  position(threadP, 52, 5); threadP->write( data->s_C_CREDIT);
  position(threadP, 15, 6); format_int( buf, 9, data->s_O_ID); threadP->write(buf, 8);
  position(threadP, 48, 6); format_int( buf, 3, data->s_O_OL_CNT); threadP->write(buf, 2);
  position(threadP, 61, 4); format_date(buf, 20, data->s_O_ENTRY_D); threadP->write(buf, 19);
  position(threadP, 64, 5); format_float(buf, 6, 2, data->s_C_DISCOUNT * 100);
threadP->write(buf, 5);
  position(threadP, 59, 6); format_float(buf, 6, 2, data->s_W_TAX*100); threadP->write(buf, 5);
  position(threadP, 74, 6); format_float(buf, 6, 2, data->s_D_TAX*100); threadP->write(buf, 5);
  total_amount = 0;
  for (i=0; i < data->s_O_OL_CNT; i++) {
            position(threadP,  3, 9+i); format_int(buf, 6, data->item[i].s_OL_SUPPLY_W_ID);
threadP->write( buf, 5 );
            position(threadP, 10, 9+i); format_int(buf, 7, data->item[i].s_OL_I_ID);
threadP->write( buf, 6 );
            position(threadP, 19, 9+i); threadP->write( data->item[i].s_I_NAME);
            position(threadP, 45, 9+i); format_int(buf, 3, data->item[i].s_OL_QUANTITY);
threadP->write(buf, 2);
            position(threadP, 51, 9+i); format_int(buf, 4, data->item[i].s_S_QUANTITY);
threadP->write(buf, 3);
            position(threadP, 58, 9+i); threadP->write(&data->item[i].s_brand_generic, 1);
            position(threadP, 62, 9+i); format_money(buf, 8, data->item[i].s_I_PRICE);
threadP->write(buf, 7);
            position(threadP, 71, 9+i); format_money(buf, 10, data->item[i].s_OL_AMOUNT);
threadP->write(buf, 9);
  }
  /* Clear the screen of any empty input fields*/
  position(threadP, 63, 24); threadP->write( "Total:");
  position(threadP, 70, 24); format_money( buf, 10, data->s_total_amount ); threadP->write( buf, 9
);

  return 0;
}

/*****************************************************************************
Payment
*****************************************************************************/
Payment::Payment(User_data *udP, Thread_data *threadP)  : Screen(udP, threadP) {
  tran_type = PAYMENT_SERVICE;
  dataptr = data = new Payment_data;
  data_len = sizeof(Payment_data);

  int lpos = 0;
  screen          = static_screen;
  empty_fields    = static_empty_fields;
#ifdef LOCAL_SESSION_DATA
  session_data = new char[static_session_data_len+1];
  sprintf(session_data, "%s%5d", POS(12,6), user_dataP->warehouse);
#else
  session_data    = static_session_data;
  sprintf(session_data, "%s%5d", POS(12,6), user_dataP->warehouse);
#endif
  screen_len      = static_screen_len;
  empty_fields_len = static_empty_fields_len;
  session_data_len = static_session_data_len;

  fields = new Field *[7];
  fields[lpos++] = genfield(threadP,  52,  6,  2, &data->s_D_ID); /* District */
  fields[lpos++] = genfield(threadP,  11, 11,  4, &data->s_C_ID); /* Customer # */
  fields[lpos++] = genfield(threadP,  29, 12, 16, (char *)data->s_C_LAST); /* Name */
  fields[lpos++] = genfield(threadP,  33, 11,  4, &data->s_C_W_ID); /* Cust-Warehouse */
  fields[lpos++] = genfield(threadP,  54, 11,  2, &data->s_C_D_ID); /* Cust-District */
  fields[lpos++] = genfield(threadP,  23, 17,  8, &data->s_H_AMOUNT); /* Amount Paid */
  fields[lpos++] = NULL;
#if defined(USE_SMART_FIELDS)
  fields[1]->ok_func = (int(*)(void*))pos_zero;
  fields[1]->ok_data = &fields[2]->pos;
  fields[2]->ok_func = (int(*)(void*))pos_zero;
  fields[2]->ok_data = &fields[1]->pos;
#endif
};

int Payment::validate() {
  if (!fields[0]->pos) {
            pos=0;
            message(threadP, "District ID is a required field");
            return 0;
  }
  if (fields[1]->pos) {
#if defined(USE_BYNAME)
            data->s_byname = 0;
#endif
  } else if (fields[2]->pos) {
#if defined(USE_BYNAME)
            data->s_byname = 1;
#endif
  } else {
            pos=1;
            message(threadP, "Customer ID or Name is required");
            return 0;
  }

  if (!fields[3]->pos) {
            pos=3;
            message(threadP, "Customer Warehouse is a required field");
            return 0;
  }

  if (!fields[4]->pos) {
            pos=4;
            message(threadP, "Customer District is a required field");
            return 0;
  }

  if (data->s_H_AMOUNT <= 0) {
            pos=5;
            message(threadP, "Enter a positive amount");
            return 0;
  }

  data->s_W_ID = user_dataP->warehouse;

  return 1;
}

int Payment::respond() {
  if (data->s_transtatus != TRAN_OK) {
            display_status(data->s_transtatus);
            return -1;
  }

  char buf[32];
  position(threadP,  52, 6); format_int(buf, 3, data->s_D_ID);   threadP->write(buf, 2);
  position(threadP,  33,11); format_int(buf, 5, data->s_C_W_ID); threadP->write(buf, 4);
  position(threadP,  54,11); format_int(buf, 3, data->s_C_D_ID); threadP->write(buf, 2);
  position(threadP,  7, 4); threadP->write( data->s_H_DATE );
  position(threadP,  1, 7); threadP->write( data->s_W_STREET_1);
  position(threadP,  42, 7); threadP->write( data->s_D_STREET_1);
  position(threadP,  1, 8); threadP->write( data->s_W_STREET_2);
  position(threadP,  42, 8); threadP->write( data->s_D_STREET_2);
  position(threadP,  1, 9); threadP->write( data->s_W_CITY);
  position(threadP,  22, 9); threadP->write( data->s_W_STATE);
  position(threadP,  25, 9); format_zip(buf, 10, data->s_W_ZIP); threadP->write(buf, 10);
  position(threadP,  42, 9); threadP->write( data->s_D_CITY);
  position(threadP,  63, 9); threadP->write( data->s_D_STATE);
  position(threadP,  66, 9); format_zip(buf, 10, data->s_D_ZIP); threadP->write(buf, 10);
  position(threadP,  11,11); format_int( buf, 5, data->s_C_ID); threadP->write(buf, 4);
  position(threadP,  9,12); threadP->write( data->s_C_FIRST);
  position(threadP,  26,12); threadP->write( data->s_C_MIDDLE);
  position(threadP,  29,12); threadP->write( data->s_C_LAST);
  position(threadP,  58,12); format_date(buf, 10, data->s_C_SINCE); threadP->write( buf, 10);
  position(threadP,  9,13); threadP->write( data->s_C_STREET_1);
  position(threadP,  58,13); threadP->write( data->s_C_CREDIT);
  position(threadP,  9,14); threadP->write( data->s_C_STREET_2);
  position(threadP,  58,14); format_float(buf, 6, 2, data->s_C_DISCOUNT*100);
threadP->write(buf, 6);
  position(threadP,  9,15); threadP->write( data->s_C_CITY);
```

```
   position(threadP,  30,15); threadP->write( data->s_C_STATE);
   position(threadP,  33,15); format_zip(buf, 10, data->s_C_ZIP); threadP->write(buf, 10);
   position(threadP,  58,15); format_phone(buf, 18, data->s_C_PHONE ); threadP->write(buf, 18);
   position(threadP,  17,17); format_money( buf, 15, data->s_H_AMOUNT); threadP->write(buf,
14);
   position(threadP,  55,17); format_money( buf, 16, data->s_C_BALANCE); threadP->write(buf,
15);
   position(threadP,  17,18); format_money( buf, 15, data->s_C_CREDIT_LIM); threadP->write(buf,
14);

   if (data->s_C_CREDIT[0] == 'B' && data->s_C_CREDIT[1] == 'C') {
                int i, size = strlen((char *)data->s_C_DATA);
                for (i = 0; i < 4; i++) {
                   position(threadP, 12,20+i);
                   threadP->write(data->s_C_DATA, (size > 50)?50:size);
                   size -= 50;
                   if (size <= 0) break;

                }
   }

   return 0;
}

/************************************************************************
OrderStatus
************************************************************************/
OrderStatus::OrderStatus(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
   tran_type = ORDERSTATUS_SERVICE;
   dataptr = data = new OrderStatus_data;
   data_len = sizeof(OrderStatus_data);

   status_x=1;
   status_y=25;

   int pos = 0;
   screen          = static_screen;
   empty_fields    = static_empty_fields;
#ifdef LOCAL_SESSION_DATA
   session_data = new char[static_session_data_len+1];
   sprintf(session_data, "%s%5d", POS(12,4), user_dataP->warehouse);
#else
   session_data    = static_session_data;
   sprintf(session_data, "%s%5d", POS(12,4), user_dataP->warehouse);
#endif
   screen_len      = static_screen_len;
   empty_fields_len = static_empty_fields_len;
   session_data_len = static_session_data_len;

   fields = new Field *[4];
   fields[pos++] = genfield(threadP,  29,  4,  2, &data->s_D_ID ); /* District */
   fields[pos++] = genfield(threadP,  11,  5,  4, &data->s_C_ID ); /* Customer ID */
   fields[pos++] = genfield(threadP,  44,  5, 16, (char *)data->s_C_LAST ); /* Customer Name */
   fields[pos++] = NULL;
#if defined(USE_SMART_FIELDS)
   fields[1]->ok_func = (int(*)(void*))pos_zero;
   fields[1]->ok_data = &fields[2]->pos;
   fields[2]->ok_func = (int(*)(void*))pos_zero;
   fields[2]->ok_data = &fields[1]->pos;
#endif
};

int OrderStatus::validate() {
   if (!fields[0]->pos) {
                pos=0;
                message(threadP, "District ID is a required field");
                return 0;
   }
   if (fields[1]->pos) {
#if defined(USE_BYNAME)
                data->s_byname = 0;
#endif
   } else if (fields[2]->pos) {
#if defined(USE_BYNAME)
                data->s_byname = 1;
#endif
   } else {
                pos=1;
                message(threadP, "Customer ID or Name is required");
                return 0;
   }

   data->s_W_ID = user_dataP->warehouse;

   return 1;
}

int OrderStatus::respond() {
   display_status(data->s_transtatus);
   if (data->s_transtatus != TRAN_OK)
                return -1;

   char buf[32];

   position(threadP, 11, 5); format_int(buf, 5, data->s_C_ID); threadP->write(buf, 4);
   position(threadP, 24, 5); threadP->write(data->s_C_FIRST);
   position(threadP, 41, 5); threadP->write(data->s_C_MIDDLE);
   position(threadP, 44, 5); threadP->write(data->s_C_LAST);
   position(threadP, 15, 6); format_money(buf, 11, data->s_C_BALANCE); threadP->write(buf, 10);
   position(threadP, 15, 8); format_int(buf, 9, data->s_O_ID); threadP->write(buf, 8);
   position(threadP, 38, 8); format_date(buf, 19, data->s_O_ENTRY_D); threadP->write(buf);
   if (data->s_O_CARRIER_ID > 0) {
                position(threadP, 76, 8);
                format_int(buf, 3, data->s_O_CARRIER_ID);
                threadP->write(buf, 2);
   }

   for (int i=0; i < data->s_ol_cnt; i++) {
                position(threadP,  3, i+10);
                format_int(buf, 6, data->item[i].s_OL_SUPPLY_W_ID);
                threadP->write(buf, 5);

                position(threadP, 14, i+10);
                format_int(buf, 7, data->item[i].s_OL_I_ID);
                threadP->write(buf, 6);

                position(threadP, 25, i+10);
                format_int(buf, 3, data->item[i].s_OL_QUANTITY);
                threadP->write(buf, 2);

                position (threadP, 32, i+10);
                format_money(buf, 10, data->item[i].s_OL_AMOUNT);
                threadP->write(buf, 9);

                position (threadP, 47, i+10);
                format_date(buf, 20, data->item[i].s_OL_DELIVERY_D);
                threadP->write(buf, 19);
   }

   return 0;
}

/************************************************************************
Delivery
************************************************************************/
Delivery::Delivery(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
   tran_type = DELIVERY_SERVICE;
   dataptr = data = new Delivery_data;
   data_len = sizeof(Delivery_data);

   status_x = 1;
   status_y = 8;

   int pos = 0;
   screen          = static_screen;
   empty_fields    = static_empty_fields;
#ifdef LOCAL_SESSION_DATA
   session_data = new char[static_session_data_len+1];
   sprintf(session_data, "%s%5d", POS(12,4), user_dataP->warehouse);
#else
   session_data    = static_session_data;
   sprintf(session_data, "%s%5d", POS(12,4), user_dataP->warehouse);
#endif
   screen_len      = static_screen_len;
   empty_fields_len = static_empty_fields_len;
   session_data_len = static_session_data_len;

   fields = new Field *[2];
   fields[pos++] = genfield(threadP,  17,  6,  2, &data->s_O_CARRIER_ID ); /* Carrier Number */

   fields[pos++] = NULL;
};

int Delivery::validate() {
   if (!fields[0]->pos) {
                pos=0;
                message(threadP, "Carrier ID is a required field");
                return 0;
   }
   time((time_t *)&(data->s_queued_time));

   data->s_W_ID = user_dataP->warehouse;

   return 1;
}

int Delivery::respond() {
   if (data->s_transtatus == TRAN_OK) {
                position(threadP, status_x, status_y);
                threadP->write("Execution Status: Delivery has been queued");
   } else {
                display_status(data->s_transtatus);
                return -1;
   }
   return 0;
}

/************************************************************************
StockLevel
************************************************************************/
StockLevel::StockLevel(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
   tran_type = STOCKLEVEL_SERVICE;
   dataptr = data = new StockLevel_data;
   data_len = sizeof(StockLevel_data);

   status_x = 1;
   status_y = 10;
```

```
    int pos = 0;
    screen          = static_screen;
    empty_fields    = static_empty_fields;
    session_data    = static_session_data;
#ifdef LOCAL_SESSION_DATA
    session_data = new char[static_session_data_len+1];
    sprintf(session_data, "%s%5d%s%2d", POS(12,4), user_dataP->warehouse,
                                                        POS(29,4),
user_dataP->district);
#else
    session_data    = static_session_data;
    sprintf(session_data, "%s%5d%s%2d", POS(12,4), user_dataP->warehouse,
                                                        POS(29,4),
user_dataP->district);
#endif
    screen_len      = static_screen_len;
    empty_fields_len = static_empty_fields_len;
    session_data_len = static_session_data_len;

    fields = new Field *[2];
    fields[pos++] = genfield(threadP,  24,  6,  2, &data->s_threshold ); /* Threshold */
    fields[pos++] = NULL;
};

int StockLevel::validate() {
    if (data->s_threshold <= 0) {
                pos=0;
                message(threadP, "A positive non-zero threshold is required");
                return 0;
    }
    data->s_W_ID = user_dataP->warehouse;
    data->s_D_ID = user_dataP->district;

    return 1;
}

int StockLevel::respond() {
    display_status(data->s_transtatus);
    if (data->s_transtatus != TRAN_OK)
                return -1;

    position(threadP, 12, 8);
    char buf[5];
    format_int(buf, 4, data->s_low_stock);
    threadP->write(buf, 4);

    return 0;
}

/*****************************************************************************
 perform
*****************************************************************************/
int NewOrder::process() {
    if (tran_type == NULL)
                return 0;

    if (encina.tran(data,threadP->contextP, tran_type) < 0) {
                return -1;
    }
    return 0;
}

int Payment::process() {
    if (tran_type == NULL)
                return 0;

    if (encina.tran(data,threadP->contextP, tran_type) < 0) {
                return -1;
    }
    return 0;
}

int StockLevel::process() {
    if (tran_type == NULL)
                return 0;

    if (encina.tran(data,threadP->contextP, tran_type) < 0) {
                return -1;
    }
    return 0;
}

int OrderStatus::process() {
    if (tran_type == NULL)
                return 0;

    if (encina.tran(data,threadP->contextP, tran_type) < 0) {
                return -1;
    }
    return 0;
}

int Delivery::process() {
    if (tran_type == NULL)
       return 0;

    if (encina.tran(data,threadP->contextP, tran_type) < 0) {
       return -1;
    }
    return 0;
}


int Screen::process() {
    if (tran_type == NULL)
                return 0;
    return 0;
}


/*****************************************************************************
Login
*****************************************************************************/
Login::Login(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
    tran_type = NULL;
    status_x=1;
    status_y=24;

    dataptr = NULL;
    data_len = 0;

    int pos = 0;
    screen          = static_screen;
    screen_len      = static_screen_len;
    empty_fields    = static_empty_fields;
    empty_fields_len = static_empty_fields_len;

    fields = new Field *[3];
    fields[pos++] = genfield(threadP,  16,  5,  5, &(udP->warehouse) ); //Warehouse
    fields[pos++] = genfield(threadP,  34,  5,  2, &(udP->district)  ); //District
    fields[pos++] = NULL;
};
int Login::validate() {
    if (!fields[0]->pos) {
                pos=0;
                message(threadP, "Warehouse ID is a required field");
                return 0;
    }
    if (!fields[1]->pos) {
                pos=1;
                message(threadP, "District ID is a required field");
                return 0;
    }

    return 1;
}


/*****************************************************************************
Menu
*****************************************************************************/
Menu::Menu(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
    tran_type = NULL;
    status_x=1;
    status_y=24;

    int pos = 0;
    screen          = static_screen;
    screen_len      = static_screen_len;
    empty_fields    = NULL;
    empty_fields_len = 0;


    fields = NULL;
};

/*****************************************************************************
Static data
*****************************************************************************/
char const * const blanks      = "                                ";
char const * const underscores = "________________________________________________";
char const * const backspaces  = "\b\b\b\b\b\b\b\b\b\b";

/*****************************************************************************
Utility Functions
*****************************************************************************/
static int string_empty(char const *data) {
    return data[0] == 0;
}
static int pos_zero(int const *val) {
    return *val == 0;
}
static int pos_nonzeros(int const **val) {
    int const **ptr;
    for (ptr = val; *ptr; ptr++) {
                if (**ptr == 0)
                    return 0;
    }
    return 1;
}

int position(int x, int y, char *buf) {
    int pos = 0;
    buf[pos++] = ESCc;
    buf[pos++] = '[';
    if (y >= 10) buf[pos++] = (y / 10) + '0';
    buf[pos++] = (y % 10) + '0';
    buf[pos++] = ';';
```

```
    if (x >= 10) buf[pos++] = (x / 10) + '0';
    buf[pos++] = (x % 10) + '0';
    buf[pos++] = 'H';
    buf[pos++] = 0;
    return 0;
}

int position(InOut *threadP, int x, int y) {
    char buf[16];
    position(x, y, buf);
    threadP->write(buf);
    return 0;
}

static int clear_eos(InOut *threadP) {
    threadP->write (ESC "[J");
    return 0;
}


int message(InOut *threadP, char const *text, int need_flush) {
    position(threadP, 1,25);
    threadP->write(text);
    clear_eos(threadP);
    if (need_flush)
      threadP->flush();
    return 0;
}

static int clear_eos(char *buf) {
    buf[0] = ESCc;
    buf[1] = '[';
    buf[2] = 'J';
    return 0;
}
```

Screen.h

```
/* (C)1997 IBM Corporation */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <termios.h>
#include <time.h>

#include "field.h"
#include "inout.h"
#include "tpcc.h"

extern int position(int x, int y, char *buf);
extern int position(InOut *ioP, int x, int y);
extern int message(InOut *ioP, char const *text, int need_flush=1);


class User_data {
public:
    int warehouse;
    int district;
};

class Thread_data : public InOut {
  public:
    void            *contextP;
    Thread_data(int infd, int outfd, void *conP) : InOut(infd, outfd), contextP(conP) {};
};

class Screen {
protected:
    static char const end_str[];
    static int        end_str_len;
          int         has_data;
    void      *dataptr;
    char      *tran_type;
    char const *screen;
    char const *empty_fields;
    char      *session_data;
    int        screen_len;
    int        session_data_len;
    int        empty_fields_len;
    int        pos;
    int        status_x, status_y;
    int        data_len;
    Thread_data                 *threadP;

public:
    User_data  *user_dataP;
    Field **fields;
    virtual char const *isa() { return "Screen"; };
    virtual int reset();
    virtual int present();
    virtual int present_empty_fields();
    virtual int process();
    virtual int user_input();
    virtual int validate()          { return 1; };
    virtual int respond()           { return 0; };
    int handle();
    int display_status(int status);
    Screen(User_data *udP, Thread_data *thrP) {
                user_dataP = udP;
                threadP = thrP;
                has_data = 0;
                pos = 0;
                fields = NULL;
                screen = empty_fields = session_data = NULL;
                screen_len = session_data_len = empty_fields_len = 0;
    };
    virtual~Screen();
};

class Login : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    int validate();
    Login::Login(User_data*udP, Thread_data *thrP);
};

class NewOrder : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
    int start_field;
public:
    NewOrder_data *data;

    int reset();
    NewOrder::NewOrder(User_data*udP, Thread_data *thrP);
    int validate();
    int process();
    int respond();
};

class Payment : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    Payment_data *data;
    int validate();
    int process();
    int respond();
    Payment(User_data *udP, Thread_data *thrP);
};

class OrderStatus : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    OrderStatus_data *data;
    int validate();
    int process();
    int respond();

    OrderStatus(User_data *udP, Thread_data *thrP);
};

class Delivery : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    Delivery_data *data;
    int validate();
    int process();
    int respond();

    Delivery(User_data*udP, Thread_data *thrP);
};

class StockLevel : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
```

```
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    StockLevel_data *data;
    int validate();
    int process();
    int respond();

    StockLevel(User_data *udP, Thread_data *thrP);
};

class Menu : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    Menu(User_data *udP, Thread_data *thrP);
};
```

screen_data.C

```
/* (C)1997 IBM Corporation */
#include "screen.h"


char const NewOrder::static_screen[] =
    POS( 1, 3) CLEAR_EOS
    POS(36, 3) "New Order"
    POS( 1, 4) "Warehouse"
    POS(19, 4) "District:"
    POS(55, 4) "Date:"
    POS( 1, 5) "Customer:"

    POS(19, 5) "Name:"
    POS(44, 5) "Credit:"
    POS(57, 5) "Disc.:"
    POS( 1, 6) "Order Number:"
    POS(25, 6) "Number of Lines:"
    POS(52, 6) "W_Tax:"
    POS(67, 6) "D_Tax:"
    POS( 2, 8) "Supp_W Item_Num Item_Name"
    POS(44, 8) "Qty  Stock  B/G   Price   Amount"
;

char const NewOrder::static_empty_fields[] =
    POS(29, 4) "____" /* District */
    POS(12, 5) "____" /* Customer */

    POS( 3, 9) "_____"
    POS(10, 9) "______"
    POS(45, 9) "__"
    POS( 3,10) "_____"
    POS(10,10) "______"
    POS(45,10) "__"
    POS( 3,11) "_____"
    POS(10,11) "______"
    POS(45,11) "__"
    POS( 3,12) "_____"
    POS(10,12) "______"
    POS(45,12) "__"
    POS( 3,13) "_____"
    POS(10,13) "______"
    POS(45,13) "__"
    POS( 3,14) "_____"
    POS(10,14) "______"
    POS(45,14) "__"
    POS( 3,15) "_____"
    POS(10,15) "______"
    POS(45,15) "__"
    POS( 3,16) "_____"
    POS(10,16) "______"
    POS(45,16) "__"
    POS( 3,17) "_____"
    POS(10,17) "______"
    POS(45,17) "__"
    POS( 3,18) "_____"
    POS(10,18) "______"
    POS(45,18) "__"
    POS( 3,19) "_____"
    POS(10,19) "______"
    POS(45,19) "__"
    POS( 3,20) "_____"
    POS(10,20) "______"
    POS(45,20) "__"
    POS( 3,21) "_____"
    POS(10,21) "______"
    POS(45,21) "__"
    POS( 3,22) "_____"
    POS(10,22) "______"
    POS(45,22) "__"
    POS( 3,23) "_____"
    POS(10,23) "______"
    POS(45,23) "__"
;

char NewOrder::static_session_data[] =
    POS(12,4) "#####" /* Warehouse Id */
;

int NewOrder::static_screen_len = sizeof(NewOrder::static_screen) - 1;
int NewOrder::static_empty_fields_len = sizeof(NewOrder::static_empty_fields) - 1;
int NewOrder::static_session_data_len = sizeof(NewOrder::static_session_data) - 1;

/* Payment */
char const Payment::static_screen[] =
    POS( 1, 3) CLEAR_EOS
    POS( 38,3) "Payment"
    POS( 1,4) "Date:"
    POS( 1,6) "Warehouse:"
    POS( 42,6) "District:"
    POS( 1,11) "Customer:"
    POS( 17,11) "Cust-Warehouse:"
    POS( 39,11) "Cust-District:"
    POS( 1,12) "Name:"
    POS( 50,12) "Since:"
    POS( 50,13) "Credit:"
    POS( 50,14) "%Disc:"
    POS( 50,15) "Phone:"
    POS( 1,17) "Amount Paid:"
    POS( 37,17) "New Cust-Balance:"
    POS( 1,18) "Credit Limit:"
    POS( 1,20) "Cust-Data:"
;
char const Payment::static_empty_fields[] =
    POS( 52, 6) "__" /* District */
    POS( 11,11) "____" /* Customer # */
    POS( 33,11) "_____" /* Cust-Warehouse */
    POS( 54,11) "__" /* Cust-District */
    POS( 29,12) "________________" /* Name */
    POS( 23,17) "________" /* Amount Paid */
;

char Payment::static_session_data[] =
    POS( 12,6) "#####" /* Warehouse */
;

int Payment::static_screen_len = sizeof(Payment::static_screen) - 1;
int Payment::static_empty_fields_len = sizeof(Payment::static_empty_fields) - 1;
int Payment::static_session_data_len = sizeof(Payment::static_session_data) - 1;

/* Order Status */
char const OrderStatus::static_screen[] =
    POS( 1, 3) CLEAR_EOS
    POS(35, 3) "Order-Status"
    POS( 1, 4) "Warehouse:"
    POS(19, 4) "District:"
    POS( 1, 5) "Customer:"
    POS(18, 5) "Name:"
    POS( 1, 6) "Cust-Balance:"
    POS( 1, 8) "Order-Number"
    POS(26, 8) "Entry-Date:"
    POS(60, 8) "Carrier-Number:"
    POS( 1, 9) "Supply-W"
    POS(14, 9) "Item-Num"
    POS(25, 9) "Qty"
    POS(33, 9)"Amount"
    POS(45, 9) "Delivery-Date"
;
char const OrderStatus::static_empty_fields[] =
    POS(29, 4) "__" /* District */
    POS(11, 5) "____" /* Customer ID */
    POS(44, 5) "________________" /* Customer Name */
;

char OrderStatus::static_session_data[] =
    POS(12, 4) "#####" /* Warehouse */
;
int OrderStatus::static_screen_len = sizeof(OrderStatus::static_screen) - 1;
int OrderStatus::static_empty_fields_len = sizeof(OrderStatus::static_empty_fields) - 1;
int OrderStatus::static_session_data_len = sizeof(OrderStatus::static_session_data) - 1;

/* Delivery */
char const Delivery::static_screen[]=
    POS(1,3) CLEAR_EOS
    POS( 38,3) "Delivery"
    POS( 1,4) "Warehouse:"
    POS( 1,6) "Carrier Number:"
;
char const Delivery::static_empty_fields[]=
    POS( 17,6) "__" /* Carrier Number */
;
char Delivery::static_session_data[] =
    POS( 12, 4) "#####" /* Warehouse */
;
int Delivery::static_screen_len= sizeof(Delivery::static_screen) - 1;
int Delivery::static_empty_fields_len= sizeof(Delivery::static_empty_fields) - 1;
int Delivery::static_session_data_len= sizeof(Delivery::static_session_data) - 1;
```

```
/* Stock level */
char const StockLevel::static_screen[] =
   POS( 1, 3) CLEAR_EOS
   POS(35, 3) "Stock-Level"
   POS( 1, 4) "Warehouse:"
   POS(19, 4) "District:"
   POS( 1, 6) "Stock Level Threshold:"
   POS( 1, 8) "Low Stock:"
;

char const StockLevel::static_empty_fields[] =
   POS( 24,6) "__" /* Threshold */
;

char StockLevel::static_session_data[] =
   POS( 12,4) "#####" /* Warehouse */
   POS( 29,4) "##"   /* District */
;
int StockLevel::static_screen_len = sizeof(StockLevel::static_screen) - 1;
int StockLevel::static_empty_fields_len = sizeof(StockLevel::static_empty_fields) - 1;
int StockLevel::static_session_data_len = sizeof(StockLevel::static_session_data) - 1;

/* Login */
char const Login::static_screen[] =
   POS( 1, 1) CLEAR_EOS
   POS(30, 3) "Please login."
   POS( 5, 5) "Warehouse:"
   POS(24, 5) "District:"
;
char const Login::static_empty_fields[] =
   POS( 16,5) "_____" /* Warehouse */
   POS( 34,5) "__" /* District */
;
int Login::static_screen_len = sizeof(Login::static_screen) - 1;
int Login::static_empty_fields_len = sizeof(Login::static_empty_fields) - 1;

/* Menu */
char const Menu::static_screen[] =
   POS(1, 1) CLEAR_EOS
   "(1)New-Order (2)Payment (3)Order-Status (4)Delivery (5)StockLevel (9)Exit"
;

int Menu::static_screen_len = sizeof(Menu::static_screen) - 1;

/* end string */
char const Screen::end_str[] = "\033[H\n";
int Screen::end_str_len = sizeof(Screen::end_str) - 1;
```

screens.h

```
/* (C)1997 IBM Corporation */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <termios.h>
#include <time.h>

#include "field.h"
#include "inout.h"
#include "tpcc.h"

extern int position(int x, int y, char *buf);
extern int position(InOut *ioP, int x, int y);
extern int message(InOut *ioP, char const *text, int need_flush=1);


class User_data {
public:
   int warehouse;
   int district;
};

class Thread_data : public InOut {
 public:
   void          *contextP;
   Thread_data(int infd, int outfd, void *conP) : InOut(infd, outfd), contextP(conP) {};
};

class Screen {
protected:
   static char const end_str[];
   static int       end_str_len;
        int       has_data;
   void     *dataptr;
   char     *tran_type;
   char const *screen;
   char const *empty_fields;
   char      *session_data;
   int       screen_len;
   int       session_data_len;
   int       empty_fields_len;
   int       pos;
   int       status_x, status_y;
   int       data_len;
   Thread_data                 *threadP;
public:
   User_data  *user_dataP;
   Field **fields;
   virtual char const *isa() { return "Screen"; };
   virtual int reset();
   virtual int present();
   virtual int present_empty_fields();
   virtual int process();
   virtual int user_input();
   virtual int validate()          { return 1; };
   virtual int respond()           { return 0; };
   int handle();
   int display_status(int status);
   Screen(User_data *udP, Thread_data *thrP) {
                user_dataP = udP;
                threadP = thrP;
                has_data = 0;
                pos = 0;
                fields = NULL;
                screen = empty_fields = session_data = NULL;
                screen_len = session_data_len = empty_fields_len = 0;
   };
   virtual ~Screen();
};

class Login : public Screen {
protected:
   static char const static_screen[];
   static char const static_empty_fields[];
   static char static_session_data[];
   static int  static_screen_len;
   static int  static_empty_fields_len;
   static int  static_session_data_len;
public:
   int validate();
   Login::Login(User_data *udP, Thread_data *thrP);
};

class NewOrder : public Screen {
protected:
   static char const static_screen[];
   static char const static_empty_fields[];
   static char static_session_data[];
   static int  static_screen_len;
   static int  static_empty_fields_len;
   static int  static_session_data_len;
   int start_field;
public:
   NewOrder_data *data;

   int reset();
   NewOrder::NewOrder(User_data *udP, Thread_data *thrP);
   int validate();
   int process();
   int respond();
};

class Payment : public Screen {
protected:
   static char const static_screen[];
   static char const static_empty_fields[];
   static char static_session_data[];
   static int  static_screen_len;
   static int  static_empty_fields_len;
   static int  static_session_data_len;
public:
   Payment_data *data;
   int validate();
   int process();
   int respond();
   Payment(User_data *udP, Thread_data *thrP);
};

class OrderStatus : public Screen {
protected:
   static char const static_screen[];
   static char const static_empty_fields[];
   static char static_session_data[];
   static int  static_screen_len;
   static int  static_empty_fields_len;
   static int  static_session_data_len;
public:
   OrderStatus_data *data;
   int validate();
   int process();
   int respond();

   OrderStatus(User_data *udP, Thread_data *thrP);
};

class Delivery : public Screen {
protected:
   static char const static_screen[];
   static char const static_empty_fields[];
   static char static_session_data[];
   static int  static_screen_len;
   static int  static_empty_fields_len;
   static int  static_session_data_len;
```

```
public:
   Delivery_data *data;
   int validate();
   int process();
   int respond();

   Delivery(User_data *udP, Thread_data *thrP);
};

class StockLevel : public Screen {
protected:
   static char const static_screen[];
   static char const static_empty_fields[];
   static char static_session_data[];
   static int  static_screen_len;
   static int  static_empty_fields_len;
   static int  static_session_data_len;
public:
   StockLevel_data *data;
   int validate();
   int process();
   int respond();

   StockLevel(User_data *udP, Thread_data *thrP);
};

class Menu : public Screen {
protected:
   static char const static_screen[];
   static char const static_empty_fields[];
   static char static_session_data[];
   static int  static_screen_len;
   static int  static_empty_fields_len;
   static int  static_session_data_len;
public:
   Menu(User_data *udP, Thread_data *thrP);
};
```

serverDebug.h

```
/*
 * serverDebug.h
 *
 * $Revision: 1.7 $
 * $Date: 1998/01/23 15:08:51 $
 * $Log:         serverDebug.h,v $
 * Revision 4.4  95/05/16  10:55:40  10:55:40    tpcc (TPCC Benchmark)
 * Added necessary RCS ident strings
 *
 *
 */
#ifndef SERVER_DEBUG
#define SERVER_DEBUG

#include <utils/trace.h>

#ifdef DEFINE_SERVER_DEBUG
long serverDebug = 0;
#else
extern long serverDebug;
#endif

#ifdef TRACE_TRANS
#define TRACETRAN(list) logprintf list
#else
#define TRACETRAN(list)
#endif

#ifdef DEBUG_SERVER
#define AUDITLOG(list) if (serverDebug & AUDIT_TRANS) UNCOND_EVENT list
#define NEWOLOG(list) if (serverDebug & DBG_NEWO) err_printf list
#define PAYLOG(list) if (serverDebug & DBG_PAY) err_printf list
#define OSLOG(list) if (serverDebug & DBG_OS) err_printf list
#define STKLOG(list) if (serverDebug & DBG_STK) err_printf list
#define DEBUGP(list) if (serverDebug) err_printf list
#else
#define AUDITLOG(list)
#define NEWOLOG(list)
#define PAYLOG(list)
#define OSLOG(list)
#define STKLOG(list)
#define DELLOG(list)
#define DEBUGP(list)
#endif

#define ERRLOG(list) err_printf list
#define SQL_RET_CODE(var, code) var = (code)

/* Fix DPRINT to write on a debugging unit that can get set differently
for delivery */

#ifdef UNIT_TEST
#define DPRINT(list) dprint list
#define DELPRINT(list) delprint list
#else
#define DPRINT(list)
#define DELPRINT(list)
#endif

#define DBG_NEWO        0x0001
#define DBG_PAY                 0x0002
#define DBG_OS                  0x0004
#define DBG_STK                 0x0008
#define DBG_DEL                 0x0010
#define DBG_ERR                 0x0020
#define AUDIT_TRANS     0x0100

#endif                  /* SERVER_DEBUG */
```

server.c

```
/*
 * server.c
 *
 * $Revision: 1.9 $
 * $Date: 1998/01/23 15:08:48 $
 * $Log:         server.c,v $
 *
 * $TALog: server.c,v $
 * Revision 1.9  1998/01/23  15:08:48  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.8 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 *
 */
/*
 *
 *      TPCC Server
 *
 *  There are currently three versions of the TPCC benchmark
 *  implemented here: An Encina monitor based benchmark,
 *  an Encina Toolkit based benchmark and a DCE only benchmark.
 *
 *  This file, server.c, contains all the code that is common to
 *  all the versions.  Each server has its own main file:
 *  serverMon.c for the monitor server, serverTK.c for the toolkit
 *  server and serverDce.c for the dce server.
 *
 *  Each server is comprised of three main modules: the server specifc
 *  one (mentioned above), the common one, in this file, and the
 *  server part, which is in the SQL files DBInfo.ec, dbInit.ec,
 *  delivery.ec, newOrder.ec, orderStatus.ec, payment.ec, stockLevel.ec.
 */
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/errno.h>
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <dce/pthread.h>

#include <utils/trace.h>
#include <tpm/mon.h>

#include "utilities.h"
#include "server.h"
#include "tpcc_type.h"
#include "do_tpcc.h"
#include <time.h>

#define DEFINE_SERVER_DEBUG
#include "serverDebug.h"

#if defined(solaris)
extern int errno;
#endif

#define TPCC_HOME   "/tmp"
#define TIME_PREFIX_LEN 50
extern char sys_errlist[];

void dprint(char *format, ...);
/*
 *  Global variables common to all types of servers
 */
FILE *server_logtrans = NULL;
int logtrans = -1;
FILE *dvry_log = NULL;   /*  FILE structure for delivery log */
int dvry_log_fd = -1;    /*  File descriptor for delivery log */
int status_log  = -1;    /*  File descriptor for status log   */
FILE *deliveryLog = NULL;
FILE *deliveryOut = NULL;
int serverIdNumber = 0;  /*  The ID of the server
                             *  This is used to identify output
                             */
int serverPid = 0;
int num_mults = 15;      /*  The number of times the matrices are
                             *  multiplied (in order to spend some time)
                             */
```

```
int server_null_test = 0;
int server_init = 0;            /*  The time (in seconds) the test started
                                *  This is used by the deferred delivery
                                *  which reports its times as elapsed time
                                *  since start time
                                */

void err_printf(char *format, ...);
void logprintf(char *format, ...);

void open_log_files()
{
   /* open DVRY_LOG to keep delivery transactions logs*/
   char logname[MAX_STR_LEN], fname[MAX_STR_LEN];
   char buffer[MAX_STR_LEN];
   char *tpcc_home;
   char *log_dir;
   int  bytes;
   int  current_fp;
   int  current;

   log_dir = getenv("DELIVERY_LOGS");
   if (log_dir == NULL) {
                fprintf(stderr, "DELIVERY_LOGS not specified, using %s\n",
                                TPCC_HOME);
                log_dir = TPCC_HOME;
   }

/*
   sprintf(buffer, "%s/status.%d", log_dir, getpid());
   status_log = creat(buffer, 0666 );
 */

   tpcc_home = getenv("TPCC_HOME");
   if (tpcc_home == NULL) {
                fprintf(stderr, "TPCC_HOME not specified, using /tmp\n");
                tpcc_home = "/tmp";
   }

   sprintf(fname, "%s/CURRENT", tpcc_home );
   current_fp = open(fname, O_RDONLY);
   bytes = read(current_fp, buffer, MAX_STR_LEN);
   if (bytes == -1) {
                fprintf(stderr, "Could not read CURRENTfile.\n");
                exit(1);
   }
   buffer[bytes] = '\0';
   current = atoi(buffer);
   close(current_fp);

   dvry_log = NULL;
}

/*
 * logprintf() -- variable argument function used to print error
 *                      and debug statements. Function is called when
 *                      any of the debug macros (defined in serverDebug.h)
 *                      are used.
 */


/*
 *  get_thread_id
 *     A function that returns the thread ID of the current thread
 */
int get_thread_id()
{
   pthread_t thread = pthread_self();
   int thread_id = pthread_getunique_np(&thread);
   return(thread_id);
}

void print_time_prefix(FILE *file)
{
   time_t cur_timet;
   char time_str[30];

   cur_timet = time(&cur_timet);
   strftime(time_str, 29, "%X", localtime(&cur_timet));

   fprintf(file, "%4d %5d %4d %s - ",
                   serverIdNumber, serverPid, get_thread_id(), time_str);
}


char *get_time_prefix(char *buffer)
{
   time_t cur_timet;
   char time_str[30];
   int len;

   cur_timet = time(&cur_timet);
   strftime(time_str, 29, "%X", localtime(&cur_timet));

   len = sprintf(buffer, "%4d %5d %4d %s - ",
                           serverIdNumber, serverPid, get_thread_id(), time_str);
   if (len >= TIME_PREFIX_LEN) {
     fprintf(stderr, "TIME_PREFIX_LEN (%d) too small: %d\n",
                           TIME_PREFIX_LEN, len);
     exit(12);
   }
   return(buffer);
}


void logprintf(char *format, ...)
{
   char formatBuffer[200];
   char *fmt = formatBuffer;
   int fmtLen;
   va_list ap;
   va_start(ap, format);

   fmtLen = TIME_PREFIX_LEN + strlen(format) + 2;
   if (fmtLen > sizeof(formatBuffer)) {
      fmt = (char *)malloc(fmtLen);
   }
   get_time_prefix(fmt);
   strcat(fmt, format);
   if (server_logtrans) {
                vfprintf(server_logtrans, fmt, ap);
                fflush(server_logtrans);
   } else {
                vfprintf(stderr, fmt, ap);
   }
   if (fmt != formatBuffer) free(fmt);
   va_end(ap);
}

void err_printf(char *format, ...)
{
   char formatBuffer[200];
   char *fmt = formatBuffer;
   int fmtLen;
   char timeBuffer[128];
   va_list ap;
   va_start(ap, format);

   fmtLen = TIME_PREFIX_LEN + strlen(format) + 2;
   if (fmtLen > sizeof(formatBuffer)) {
      fmt = (char *)malloc(fmtLen);
   }
   get_time_prefix(fmt);
   strcat(fmt, format);
   if (server_logtrans) {
      vfprintf(server_logtrans, fmt, ap);
      fflush(server_logtrans);
   }
   vfprintf(stderr, fmt, ap);

   if (fmt != formatBuffer) free(fmt);
   va_end(ap);
}


/*
 * dprint() -- variable argument function used to print debug
 *                      statements; for use with DPRINT macro.
 */

void dprint(char *format, ...)
{
   va_list ap;
   va_start(ap, format);

   print_time_prefix(stderr);
   vfprintf(stderr, format, ap);

   va_end(ap);
}
```

server.h

```
/*
 *                  server.h
 *
 * $Revision: 1.7 $
 * $Date: 1998/01/23 15:08:50 $
 * $Log:            $
 *
 *
 * $TALog: server.h,v $
 * Revision 1.7  1998/01/23  15:08:50  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.6 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 *
 */

/**  server.h  **/

/**  Declarations common to all the server modules  **/
```

```
#ifndef TPCC_SERVER_H
#define TPCC_SERVER_H

#define get_dbname_from_id(i) rmList[i].dbName

typedef enum {
   mon_server = 11
} server_type_t;

extern int server_no_db;
extern int serverIdNumber;
extern int server_init;
extern server_type_t server_type;
extern int get_db_for_wh(int);

#endif   /*  TPCC_SERVER_H */
```

ServerMon.c

tpcc_monitor.c

```
/*
 *                tpcc_monitor.c
 *
 * $Revision: 1.5 $
 * $Date: 1998/07/08 18:15:42 $
 * $Log:            $
 *
 *
 * $TALog: tpcc_monitor.c,v $
 * Revision 1.5  1998/07/08  18:15:42  wenjian
 * Minor change for print_header().
 * [from r1.4 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.6]
 *
 * Revision 1.4  1998/07/02  18:28:52  wenjian
 * - Change the display to table format.
 * - Display more information of client process.
 * - Rewrite code for better modularity.
 * [from r1.3 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.5]
 *
 * Revision 1.1  1998/04/29  19:47:43  wenjian
 * - Copy open_socket from terminal.c
 * - Communicate with tpcc_client, calculate the TPM and response time,
 *   and display the information
 * [added by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.1]
 *
 * Revision 1.18  1998/02/17  22:07:07  wenjian
 * - Add head files
 * - Define macros to dead with the different function names on NT
 * [from r1.17 by delta wenjian-21750-TPCC-changes-for-porting-on-NT,r1.1]
 *
 */

/*
 *   This is the monitor program for checking status of each client machine
 */

#include <sys/types.h>
#include <sys/stat.h>
#ifndef WIN32
#include <sys/select.h>
#include <termios.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/un.h>
#else /* WIN32 */
#include <io.h>
#include <process.h>
#include <winsock.h>
#include "tran_stat.h"
#endif /* WIN32 */
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#if defined (solaris)
#include <dce/pthread.h>
#else   /* solaris */
#include <pthread.h>
#endif  /* solaris */
#include "terminal.h"
#include "client_utils.h"
#include "tpcc_type.h"
#include "tpcc_monitor.h"

#define BIND_LOCAL

#ifdef WIN32
#define EADDRINUSE      WSAEADDRINUSE
#define SLEEP(A)        Sleep(A*1000)
#define RANDOM                  rand
#define SRANDOM                 srand
#define stat                    _stat
#define close                   _close
#define read(A,B,C) recv(A,B,C,0)
#define write(A,B,C) send(A,B,C,0)
#else
#define SLEEP(A)        sleep(A)
#define RANDOM                  random
#define SRANDOM                 srandom
#endif

#define MAX_PORTS 20
#define MAX_CLIENTS 30
#define MAX_FILE_NAME 30

#define AUTOMATIC    0
#define MANUAL               1
#define ALL      2
#define INDIVIDUAL   3

#define QUIT                 10001
#define MODE_CHANGE  10002
#define WRONG_INPUT  10003
#define REFRESH              10004
#define HELP                 10005
#define LEADING_SPACE "          "

#define EXIT(code) exit_program(code)
#define TRIGGERc '\021'

static void write_to_all_clients(int clnt_id, int *out, char *buffer, int len);
static void write_to_client(int out, char *buffer, int len);
static int read_from_all_clients(int clnt_id, int *sock, int initialized);
static int read_from_client(int sock, char *buffer, int buf_len);
void   print_header(char *prefix, int rt_mode, int display, double interval);
void   print_info(char *, client_status_t client_st);
static void show_info();
static int open_socket(int id, char *address);
static void exit_program(int err);
void   one_print_rt_avg(client_status_t, int, int);
int    get_next_choice();
void   get_client_info(client_status_t *, char *, int, int);
void   clients_status_summary(client_status_t**client_stat);
void   display_client(int clnt, client_status_t **client_stat);
void   display_all_clients(client_status_t *client_stat[]);
void   print_errs(client_status_t **client_stat);
void   set_client_st_rt(client_status_t *, total_tran_count_t *, total_tran_count_t *);
void   set_client_st(client_status_t *, double, int, char *, double, int, int, double);
void   client_stat_init(client_status_t **);

/* global variable */
static char *sys_errlist[256];
int    user_id;          /*  this terminal's ID (what file to write)  */
char   *user_code = "T";  /*  Prefix for output to identify this
                          *  process as a client or a terminal
                          */
FILE   *logtpcc = NULL;   /*  The log file for debug output          */
int    debug = 0;                /*  Debug mode -- set to 0 for real runs              */
int    initialized=0;    /*  The variable for initialization        */
int    client_id;                /*  which client to talk to                           */
int    next_port_to_try;
char   client_address[MAX_CLIENTS+1][MAX_FILE_NAME];
int    num_client;
int    client_port;
int    num_port;
int    input_mode=AUTOMATIC;          /*  control auto or manual input mode*/
int    display_mode=ALL;        /*  control display of client(s)*/
int    rt_mode=0;               /*  0-client RT, 1-server RT */

/* the followings are global vars to summarize the TPM and RTs for all clnts */
client_status_t  *client_stat[MAX_CLIENTS+1];

/*
 *  Open a socket to the client whose id is 'id'
 */
static int open_socket(port, address)
  int port;
  char *address;
{
    int retries = 0;
    int sock;   /* socket */
    struct sockaddr_in sockAddrIn;
    struct sockaddr *sockAddrP;
    int sockAddrLen;
    int ip_address;

    if (getenv("TPCC_DEBUG") != NULL)
      debug = 1;

    /* now open a stream socket */
    ip_address = inet_addr(address);
    if (ip_address == -1) {
```

```
                struct hostent *hp;
                struct in_addr in;
                char *buf;
                hp = gethostbyname(address);
                if (hp == NULL) {
                   err_printf("Could not resolve %s\n", address);
                   exit(2);
                }
                buf = hp->h_addr_list[0];
                (void) memcpy(&in.s_addr, buf, sizeof (in.s_addr));
                ip_address = inet_addr(inet_ntoa(in));
                if (ip_address == -1) {
                   err_printf("host %s resolved to %s which failed. OOPS\n",
                                address, inet_ntoa(in));
                   exit(2);
                }
  }

#ifdef WIN32
  if (1) {
   WORD wVersionRequested;
   WSADATA wsaData;
   int err;

   wVersionRequested = MAKEWORD( 2, 2 );

   err = WSAStartup( wVersionRequested, &wsaData );
   if ( err != 0 ) {
                printf("could not find a usable WinSock DLL\n");
                exit(2);
   }

   /* Confirm that the WinSock DLL supports 2.0.*/
   /* Note that if the DLL supports versions greater    */
   /* than 2.2 in addition to 2.2, it will still return */
   /* 2.2 in wVersion since that is the version we      */
   /* requested.                                        */

   if ( LOBYTE( wsaData.wVersion ) != 2 ||
                 HIBYTE( wsaData.wVersion ) != 2 ) {
                printf("could not find a usable WinSock DLL\n");
                WSACleanup( );
      exit(2);
   }
}
#endif

   /* The WinSock DLL is acceptable. Proceed. */
   sock = socket(AF_INET, SOCK_STREAM, 0);
/*   printf("open_sock: socket is %d\n", sock); */
   if (sock == -1) {
                err_printf("open_socket: failed (errno=%s (%d)).\n",
                                sys_errlist[errno], errno);
                exit(2);
   }
#ifdef BIND_LOCAL
   /*  Bind the socket (There is a limit of less than
    *  4000 on ephemeral sockets
    */
   while (1) {
      int rc;
      int port = next_port_to_try ++;
      sockAddrIn.sin_family = AF_INET;
      sockAddrIn.sin_port = port;
      sockAddrIn.sin_addr.s_addr = INADDR_ANY;
      rc = bind(sock,
                                (struct sockaddr *)&sockAddrIn,
             sizeof(sockAddrIn));
      if (rc == 0) {
                   DPRINT(("Bound socket %d to port %d\n", sock, port));
                   break;
                }
                if (errno != EADDRINUSE) {
         err_printf("open_socket: bind to port %d failed errno=%s(%d),sock %d, retry %d\n",
                                port,
                 sys_errlist[errno], errno, sock, retries);
                   exit(13);
      }
                if (port > 65000) {
                   err_printf("open_socket: bind to port %d failed - too many ports tried\n",
                                port);
                   exit(13);
                }
   }
#endif

   sockAddrIn.sin_family = AF_INET;
   sockAddrIn.sin_port = port;
   sockAddrIn.sin_addr.s_addr = ip_address;
   sockAddrP = (struct sockaddr *)&sockAddrIn;
   sockAddrLen = sizeof(sockAddrIn);

   while(connect(sock, sockAddrP, sockAddrLen) == -1) {
                /*
                 *  Try up to 15 times to connect to the socket
                 *  before giving up
                 */
                if (retries++ < 15) {
                   SLEEP(1+ retries/3);
                   err_printf("open_socket: connect failed errno=%s(%d),sock %d, retry
%d\n",
                                sys_errlist[errno], errno, sock, retries);
                } else {
                   err_printf("open_socket: connect failed errno=%s(%d)\n",
                                sys_errlist[errno], errno);
                   close(sock);
                   EXIT(1);
                }
   }
   return(sock);
}

#ifndef WIN32
static void set_term_for_input(int in_fd, struct termios *save_term)
{
   struct termios tc_new;
   int rc;
   if ((rc = tcgetattr(in_fd, save_term)) < 0) {
                return;
   }

   tc_new = *save_term;

#if 1
   tc_new.c_lflag &= ~(ECHO | ICANON); /* echo off, canonical mode off */

   tc_new.c_cc[VMIN] = 1;     /* Case B: 1 byte at a time, no timer */
   tc_new.c_cc[VTIME] = 0;
#else
   /* New TTY stuff */
   tc_new.c_cflag &= ~(CBAUD | HUPCL);
   tc_new.c_cflag &= ~(CBAUD);
   tc_new.c_cflag |= CLOCAL | B38400;
   tc_new.c_iflag = IGNBRK;
   tc_new.c_iflag &= ~(IXON|IXOFF|IXANY);
   tc_new.c_oflag = tc_new.c_lflag = 0;
   tc_new.c_cc[VMIN] = 1;
   tc_new.c_cc[VTIME] = 0;
#endif

   if (tcsetattr(in_fd, TCSANOW, &tc_new) < 0)
                return;
}
#endif

static void get_term_env(argc, argv)
                int argc;
                char *argv[];

{
   int  i;
   char *filename;
   FILE *inFile;

   if (argc < 4) {
                   fprintf(stderr, "Usage: %s num_port start_port num_clients
filename\n",argv[0]);
                   exit(1);
   }
   num_port = atoi(argv[1]);
   if (num_port > MAX_PORTS) {
                fprintf(stderr, "Too many ports for each client.\n");
                exit(1);
   }
   client_port = atoi(argv[2]);
   num_client = atoi(argv[3]);
   if (num_client <=0) {
                fprintf(stderr, "Number of clients has to be greater than 0.\n");
                exit(1);
   }
   if (num_client > MAX_CLIENTS) {
                fprintf(stderr, "Too many clients.\n");
                exit(1);
   }
   filename = argv[4];
   inFile = fopen(filename,"r");
   if (inFile == NULL) {
                fprintf(stderr,"File '%s' does NOT exist.\n", filename);
                exit(1);
   }
   for (i=0; i<num_client; i++) {
      int length;
      length = fscanf(inFile,"%s",client_address[i]);
      if (length<=0) {  /* the end of file */
                                num_client = i;
                                break;
                }
   }

}
```

```
static void show_info()
{
    int sock[MAX_PORTS*MAX_CLIENTS];
    int clnt_id, i, j;
    char outbuf[128];
    int  len, w;
    static last_choice=0;

    for (clnt_id=0; clnt_id<num_client; clnt_id++) {
      for (i=0; i<num_port; i++) {
                int sock_id = clnt_id*num_port + i;
                sock[sock_id]=open_socket(client_port+i, client_address[clnt_id]);
      }
    }

    /* first read and write login as a special terminal */
    read_from_all_clients(0, sock, 0);
    sprintf(outbuf, "%d\t%d\n",0, 1);
    len = strlen(outbuf);
    write_to_all_clients(0, sock, outbuf, len);

    /* initial data are garbage from the login session, so we skip them */
    read_from_all_clients(0, sock, 0);

    while (1) {

        int       choice;

        choice = get_next_choice();

                if ( choice == QUIT ) {
                      /* exit the tpcc_monitor program */
                    write_to_all_clients(0, sock, "q", 1);
                        break;
                    }
                    else if ( choice == WRONG_INPUT ) {
                                fprintf(stderr,"Wrong input. Try again.\n");
        }
        else if ( choice == REFRESH ) {
              write_to_all_clients(choice, sock, "n", 1);
              read_from_all_clients(choice, sock, 1);
              if (last_choice==0)
                                display_all_clients(client_stat);
              else
                                display_client(last_choice, client_stat);
        }
        else if ( choice == HELP ) {
                                fprintf(stderr,"Help info is not available yet.\n");
        }
        else if ( choice != MODE_CHANGE ) {
              if (choice==0)
                                display_all_clients(client_stat);
              else
                                display_client(choice, client_stat);
                              last_choice = choice;
        }
        /* other inputs will not be responsed here */
    }
}

/*
 *    main program
 *
 */
int main(argc, argv)
  int argc;
  char *argv[];
{
    int sock;
    int next_arg = 1;
    int num_local_wares;
    int i;

    SRANDOM(getpid());
    next_port_to_try = 13000 + (RANDOM() % 120) * 150;

    get_term_env(argc, argv);
    client_stat_init(client_stat);
#ifdef WIN32
    input_mode = MANUAL;
#endif
    show_info();
    exit_program(0);
}



/* read_from_all_clients:
 *
 * num_print: 1 means the login data,
 *            0 means the first staus data to be skipped,
 *            2 means the normal data to be displayed
 */
static int read_from_all_clients(int clnt_id, int *sock, int initialized)
{
    char buffer[MAX_PORTS][4096];
    int num;
    int i,j, first_clnt, last_clnt, cur_clnt;
    int len[MAX_PORTS];

    for (cur_clnt=0; cur_clnt<num_client; cur_clnt++) {
      for (i=0; i<num_port; i++) {
        int s_id = cur_clnt * num_port + i;
                    len[i] = read_from_client(sock[s_id],buffer[i],sizeof(buffer[i]));
                    if ( initialized ) {
          /* Individual client and port start from 1.
           * Index 0 is reserved for summary
           */
                        get_client_info(&client_stat[cur_clnt+1][i+1], buffer[i], cur_clnt, i);
                    }
      }
    } /* for cur_clnt */

    if (initialized)
                    clients_status_summary(&client_stat[0]);
    return 1;
}


static void write_to_all_clients(int clnt_id, int *out, char *buffer, int len)
{
  int i;
  int first_clnt, last_clnt, cur_clnt;

  for (cur_clnt=0; cur_clnt<num_client; cur_clnt++) {
    for (i=0; i<num_port; i++) {
      int sock_id = cur_clnt*num_port + i;
/*                  fprintf(stderr, "write_to_all, cur_clnt is %d, num_port is %d, socket is %d\n",
cur_clnt, num_port, sock_id); */
                    write_to_client(out[sock_id], buffer, len);
    }
  }
}

/*
 * write_to_client
 *
 *  Input: out: the file descriptor to be used to send the output
 *                Buffer: The buffer to be sent
 *                len: The length of the above buffer
 */
static void write_to_client(int out, char *buffer, int len)
{
    int w, i;

    DPRINT((">> write_to_client. len:%d\n",len));
    w = write(out, buffer, len);
    if (w != len) {
                    fprintf(stderr," write_to_client write returned: %d instead of%d\n",
                                w, len);
                    EXIT(4);
    }
    DPRINT(("<< write_to_client\n"));
}

/*
 *  read_from_client
 *                Process data returning from the client.
 *                At this point we simply send the data to standard out
 *
 *  Input: sock: The file descriptor to be used to read the data
 *                Buffer: A buffer allocated by the caller to store the data read
 *                len: The length of the above buffer
 *  OUTPUT: Returns the size it actually read.
 *
 *  The input from the client is a null terminated string.
 */
static int read_from_client(int sock, char *buffer, int buf_len)
{
    int r;
    char *nextP = buffer;
    int len = 0;
    int done = 0;
    char  mark[10]=ENDMSG;

    do {
                int i;
                DPRINT(("before Reading from client, len %d\n",len));
                r = read(sock, nextP, buf_len - len);
                if (r < 1) {
                    if (r == 0) {
                                fprintf(stderr,"Connection closed.  Bye!\n");
                                EXIT(5);
                    }

                    fprintf(stderr,"process_client_data read returned: %d,err_code=%d\n",
r,errno);
                    EXIT(6);
                }
```

```
                for (i=0; i<r; i++) {
                    /* a mark to make sure it is the end of message */
                    if ( (i+3 <r) && (nextP[i] == mark[0]) && (nextP[i+1]==mark[1])
                                && (nextP[i+2]==mark[2]) && (nextP[i+3] == mark[3]) ) {
                            nextP[i] = '\0';
                            done = 2;
                            break;
                    }
                    if (nextP[i] == TRIGGERc) {
                            done = 1;
                            break;
                    }
                }
                nextP += r;
                len += r;
    } while ((len < buf_len) && !done);
    if (len < buf_len-1) {
      if (done==1)
         buffer[len] = '\0';
      else
                 buffer[len-strlen(mark)] = '\0';
    }

    DPRINT(("read_from_client returning %d\n",len));
    return(len);
}

static void exit_program(int err)
{
    if ( err )
                fprintf( stderr, "exit_program: Error Code = %d\n", err );
    exit( err );
}


/* skip_endline:
 *
 */

char *skip_endline(char *curP)
{
    int  i;

    for (i=0; i<strlen(curP); i++) {
                if (curP[i]=='\n')
                    return curP+i+1;
    }
    printf("ERROR in skip_endline\n");
}

/* get_client_info:
 * calculate TPM and response time
 * tag: is used to for future expandation
 */

void get_client_info(client_status_t *client_rec,
                                char *buf, int clnt, int port)
{
    int             i, j;
    int             type;
    long    time_sec, time_usec;
    double time_diff1, time_diff2;
    char    prefix1[50], prefix2[50], prefix3[50];
    char    *bufP = buf;
    total_tran_count_t      tran_ct;
    tran_info_t  *curP;
    int                     total_newo;
    double tpm1, tpm2;
    int    err_diff1, err_diff2;
    static init_state = 1;
    static total_tran_count_t               tran_reported[MAX_CLIENTS][MAX_PORTS][2];
    static struct timeval  oldUserTime[MAX_CLIENTS][MAX_PORTS];
    static struct timeval  oldSysTime[MAX_CLIENTS][MAX_PORTS];
    static struct timeval  newUserTime[MAX_CLIENTS][MAX_PORTS];
    static struct timeval  newSysTime[MAX_CLIENTS][MAX_PORTS];
    int    num_threads, num_threadsInit;
    double cpuUsagePercent;

    /* initialiation */
    if (init_state) {
       for (i=0; i<MAX_CLIENTS; i++)
         for (j=0; j<MAX_PORTS; j++) {
                memset(&tran_reported[i][j][0],0, sizeof(tran_ct));
                memset(&tran_reported[i][j][1],0, sizeof(tran_ct));
         }
                init_state = 0;
    }

    /* read information from buf line by line */
    /* this part has to be consistent with client_status_report() */
    memset(&tran_ct, 0, sizeof(tran_ct));
    sscanf(bufP,"%s%s%s",prefix1,prefix2,prefix3);
    bufP = skip_endline(bufP);
    sscanf(bufP, "%d %d %d %d %d %d%d",&time_sec, &time_usec,
                                            &total_newo, &tran_ct.errors,
                                                &num_threads, &num_threadsInit);
    bufP = skip_endline(bufP);
    sscanf(bufP, "%d %d %d %d",
                &newUserTime[clnt][port].tv_sec, &newUserTime[clnt][port].tv_usec,
                &newSysTime[clnt][port].tv_sec, &newSysTime[clnt][port].tv_usec);
    bufP = skip_endline(bufP);
    for (i=1, curP=tran_ct.tran+1; i<=MAX_TRAN_TYPE; i++, curP++) {
                float  clnt_id, srv;
                sscanf(bufP,"%d%d %d %f %f"&type, &(curP->num), &(curP->errs),
                                &clnt_id, &srv);
       curP->RT[0] = clnt_id;
                curP->RT[1] = srv;
       fflush(stdout);

       if (i<MAX_TRAN_TYPE)  bufP = skip_endline(bufP);
    }

    /* calculate the TPM */
    /* only the last interval is interested. */
    /* time_diff1, tpm1, and err_diff1 should be removed */
    tran_ct.time = time_sec + (double)time_usec / 1e6;
    tran_ct.tran[NEWO_TRANS].num= total_newo;
    time_diff1 = tran_ct.time - tran_reported[clnt][port][0].time;
    time_diff2 = tran_ct.time - tran_reported[clnt][port][1].time;
    tpm2 = (double)(total_newo - tran_reported[clnt][port][1].tran[NEWO_TRANS].num)/
time_diff2 * 60;

    /* calculate the errors */
    err_diff1 = tran_ct.errors-tran_reported[clnt][port][0].errors;
    err_diff2 = tran_ct.errors-tran_reported[clnt][port][1].errors;

    /* calculate the CPU usage */
    if ((int)tran_reported[clnt][port][1].time==0)
                cpuUsagePercent = 0.0;
    else {
                double user_time_diff, sys_time_diff;
       user_time_diff = (double)newUserTime[clnt][port].tv_usec/1e6
                                        + newUserTime[clnt][port].tv_sec
                                - (double)oldUserTime[clnt][port].tv_usec/ 1e6
                                        - oldUserTime[clnt][port].tv_sec;
       sys_time_diff = (double)newSysTime[clnt][port].tv_usec/1e6
                                        + newSysTime[clnt][port].tv_sec
                                - (double)oldSysTime[clnt][port].tv_usec/ 1e6
                                        - oldSysTime[clnt][port].tv_sec;
       cpuUsagePercent = (user_time_diff + sys_time_diff) / time_diff2 * 100;
    }

    /* keep all the information except RT in client_rec */
    set_client_st(client_rec, tpm2, err_diff2, prefix2,
                                time_diff2, num_threads-num_threadsInit,num_threadsInit,
                                cpuUsagePercent);
    set_client_st_rt(client_rec, &tran_ct,&tran_reported[clnt][port][1]);

    /* keep the old values */
    tran_reported[clnt][port][0]= tran_reported[clnt][port][1];
    tran_reported[clnt][port][1]= tran_ct;
    oldUserTime[clnt][port]= newUserTime[clnt][port];
    oldSysTime[clnt][port]= newSysTime[clnt][port];
}


void clients_status_summary(client_status_t*client_stat[])
{
    int  i, j, k;

    /* initialization before each summary*/
    /* client_stat[i][0] is the summary of client i */
    /* client_stat[0][0] is the summary of all clients */
    for (i=0; i<=num_client; i++) {
      client_stat[i][0].tpm= 0;
      client_stat[i][0].err_diff = 0;
      client_stat[i][0].num_thrds= 0;
      client_stat[i][0].num_thrdsInit= 0;
      client_stat[i][0].cpu_usage_percent= 0;
      for (k=1; k<=MAX_TRAN_TYPE; k++) {
                client_stat[i][0].num_trans[k]= 0;
                client_stat[i][0].rt_diff[k][0]= 0;
                client_stat[i][0].rt_diff[k][1]= 0;
      }
    }

    for (i=1; i<=num_client; i++) {
      for (j=1; j<=num_port; j++) {
        /* calculate the summary for client i */
                client_stat[i][0].tpm += client_stat[i][j].tpm;
                client_stat[i][0].err_diff += client_stat[i][j].err_diff;
                client_stat[i][0].total_errs += client_stat[i][j].err_diff;
                client_stat[i][0].cpu_usage_percent += client_stat[i][j].cpu_usage_percent;
        if (j==1) {
                strcpy(client_stat[i][0].cur_time,client_stat[i][1].cur_time);
                client_stat[i][0].interval = client_stat[i][j].interval;
        }
                client_stat[i][0].num_thrds += client_stat[i][j].num_thrds;
                client_stat[i][0].num_thrdsInit += client_stat[i][j].num_thrdsInit;
```

```
      /* response time */
      for (k=1; k<=MAX_TRAN_TYPE; k++) {
                  client_stat[i][0].num_trans[k]+=client_stat[i][j].num_trans[k];
                  client_stat[i][0].rt_diff[k][0]+=client_stat[i][j].rt_diff[k][0];
                  client_stat[i][0].rt_diff[k][1]+=client_stat[i][j].rt_diff[k][1];
      }
    }

    /* caclulate the summary for all clients */
    client_stat[0][0].tpm += client_stat[i][0].tpm;
    client_stat[0][0].err_diff += client_stat[i][0].err_diff;
    client_stat[0][0].total_errs += client_stat[i][0].err_diff;
    if (i==1) {
                  strcpy(client_stat[0][0].cur_time,client_stat[i][0].cur_time);
                  client_stat[0][0].interval = client_stat[i][0].interval;
    }
    client_stat[0][0].num_thrds += client_stat[i][0].num_thrds;
    client_stat[0][0].num_thrdsInit += client_stat[i][0].num_thrdsInit;
    /* response time */
    for (k=1; k<=MAX_TRAN_TYPE; k++) {
                client_stat[0][0].num_trans[k]+=client_stat[i][0].num_trans[k];
                client_stat[0][0].rt_diff[k][0]+=client_stat[i][0].rt_diff[k][0];
        client_stat[0][0].rt_diff[k][1]+=client_stat[i][0].rt_diff[k][1];
    }
  }
}


void display_client(int clnt, client_status_t *client_stat[])
{
   int i, j;
   char prefix[20];

   print_header(client_stat[clnt][0].cur_time, rt_mode, clnt,
            client_stat[clnt][0].interval);
   /* print out information for each client*/
   for (i=0; i<num_port; i++) {
      sprintf(prefix, "(port-%1d)",client_port+i);
              print_info(prefix,client_stat[clnt][i+1]);
      one_print_rt_avg(client_stat[clnt][i+1], rt_mode, 0);
   }
   /* print out information for all the clients */
   printf ("--------------------------------------------------------------------------------\n");
   print_info("Total",client_stat[clnt][0]);
   one_print_rt_avg(client_stat[clnt][0], rt_mode, 1);
   fflush(stdout);
}

void display_all_clients(client_status_t *client_stat[])
{
   int i, j;

   print_header(client_stat[0][0].cur_time, rt_mode, 0,
            client_stat[0][0].interval);
   /* print out information for each client*/
   for (i=0; i<num_client; i++) {
              print_info(client_address[i],client_stat[i+1][0]);
      one_print_rt_avg(client_stat[i+1][0], rt_mode, 0);
   }
   /* print out information for all the clients */
   printf ("--------------------------------------------------------------------------------\n");
   print_info("Total",client_stat[0][0]);
   one_print_rt_avg(client_stat[0][0], rt_mode, 1);
   fflush(stdout);
   fflush(stdout);
}


static void one_print_rt_avg(client_status_t client_st,
                                        int rt_mode, int both)
{
   int i;
   static char   *srvPrefix = "Server RT                          ";
   static char   *clnPrefix = "Client RT                          ";
   for (i=1; i<=MAX_TRAN_TYPE; i++) {
              if ( client_st.num_trans[i]>0 )
        printf(" %4.3f", (double)(client_st.rt_diff[i][rt_mode])/(client_st.num_trans[i]*1000));
      else
                 printf("   -- ");
   }
   if (client_st.cpu_usage_percent>0)
     printf("%5.1f",client_st.cpu_usage_percent);
   else printf("   --");
   printf("\n");
   if (both) {
        rt_mode = 1 - rt_mode;  /* print SRV or CLNT RT based on rt_mode */
        printf("%s",rt_mode ? srvPrefix : clnPrefix);
        for (i=1; i<=MAX_TRAN_TYPE; i++) {
                 if ( client_st.num_trans[i]>0 )
           printf(" %4.3f",
(double)(client_st.rt_diff[i][rt_mode])/(client_st.num_trans[i]*1000));
          else
                   printf("   -- ");
        }
        printf("\n");
    }
}

void set_client_st(client_status_t *client_rec, double tpm, int errs,
                                char *time, double interval, int num_thr, int num_thrInit,
                                double cpuUsagePercent)
{
   client_rec->tpm = tpm;
   client_rec->err_diff = errs;
   client_rec->total_errs += errs;
   strcpy(client_rec->cur_time,time);
   client_rec->interval = interval;
   client_rec->num_thrds = num_thr;
   client_rec->num_thrdsInit = num_thrInit;
   client_rec->cpu_usage_percent = cpuUsagePercent;
}


void set_client_st_rt(client_status_t *client_rec,
                total_tran_count_t *curP, total_tran_count_t *prevP)
{
   int i,j;

   for (i=1; i<=MAX_TRAN_TYPE; i++) {
     for (j=0; j<2; j++) { /* for client RT and server RT */
       client_rec->num_trans[i]=curP->tran[i].num - prevP->tran[i].num;
       client_rec->rt_diff[i][j]=curP->tran[i].RT[j] - prevP->tran[i].RT[j];
     }
   }
}

int get_next_choice()
{

   static  int choice=0;
   static  struct timeval timeout={60,0};
   static  int  new_start=1;
   static  int  interval = 60;
   static  struct  timeval time1, time2;
   char cmdbuf[256];
   fd_set  readfds;
   int     j;
   int     isNT=0;
   struct  timezone tz;


#ifdef WIN32
   isNT = 1;
#endif

   /* print out interactive input choices */
   if (input_mode==AUTOMATIC){
     fprintf(stderr, "\n(AUTO-%d)[Interval Manual ",interval);
   }
   else if (!isNT) { /* MANUAL_MODE */
     fprintf(stderr, "\n(MANUAL) [Auto ");
   }
   fprintf(stderr, "Refresh Client-rt Server-rt Quit 0...%1d] > ",num_client);
   FD_ZERO(&readfds);
   FD_SET(0, &readfds);

   if ( input_mode==AUTOMATIC) {
     if (new_start) {
                 gettimeofday(&time1,&tz);
       timeout.tv_sec = interval;
                 new_start = 0;
     }
     else {
       float time_diff;
                 gettimeofday(&time2,&tz);
       time_diff = time_diff_ms(&time2,&time1)/1000;
       if (time_diff < timeout.tv_sec)
                           timeout.tv_sec -= (int)time_diff;
       else
                           timeout.tv_sec = 0;
       time1 = time2;
     }
     if (select(1,&readfds,NULL,NULL,&timeout) > 0) {
       int cc;
       cc = read(0, cmdbuf, sizeof(cmdbuf));
       if (cc <= 0)
             fprintf(stderr, "\n Reading error from stdin\n");
       else {
                 if ( cmdbuf[0]=='m' ) {
                           input_mode = MANUAL;
                           return(MODE_CHANGE);
                 }
         else if ( cmdbuf[0] == 'i' || cmdbuf[0] == 'I' ) {
                           fprintf(stderr, "New interval in seconds: ");
                           scanf("%d",&interval);
                           if (interval < 5) {
                             interval = 5;
                           }
                           timeout.tv_sec = interval;
                           fprintf(stderr, "interval change successful.\n");
```

```
                        new_start = 1;
                        return(MODE_CHANGE);
                }
                else if ( cmdbuf[0] == '\n' )  { /* for a single return key */
                        new_start = 1;
                        return(REFRESH);
                }
            }
        } else { /* no input so far,keep the last input_mode and display_mode */
                new_start = 1;
                return(REFRESH);
        }
    }
    else { /* MANUAL mode */
        scanf("%s",cmdbuf);
        if ( (cmdbuf[0] == 'a') && (!isNT) ) {
                input_mode = AUTOMATIC;
            new_start = 1;
                return(MODE_CHANGE);
        }
        else if (cmdbuf[0] == '\n') {
                return(choice);
        }
    }

    /* take care of the common input */
    if ( cmdbuf[0]<='9' && cmdbuf[0]>='0' ) {
            choice = atoi(cmdbuf);
                if (choice==0) {
                        display_mode = ALL;
                        return(choice);
                }
                else if (choice <= num_client) {
                    display_mode = INDIVIDUAL;
                    return choice;
                }
                else {
                        choice = 0;
                        return WRONG_INPUT;
                }
    }
    else if (cmdbuf[0] == 'q' || cmdbuf[0] == 'Q') {
                return(QUIT);
    }
    else if (cmdbuf[0] == 'c' || cmdbuf[0] == 'C') {
                rt_mode = 0;
                return(choice);
    }
    else if (cmdbuf[0] == 's' || cmdbuf[0] == 'S') {
                rt_mode = 1;
                return(choice);
    }
    else if (cmdbuf[0] == 'r' || cmdbuf[0] == 'R') {
                return(REFRESH);
    }
    else if (cmdbuf[0] == 'h' || cmdbuf[0] == 'H') {
                return(HELP);
    }
    else   return(WRONG_INPUT);
}


/* type=1 means server RT; type=0 means client RT */
void print_header(char *prefix, int rt_mode, int clnt, double time_diff)
{
    int i;
    /* TPM ErrPm Act Init NewO Pay SL OS DVRY */
    printf("\n\n");
/*    printf("%s", input_mode==AUTOMATIC ? "[AUTO]" : "[MANUAL]");*/

    printf("%s %s - %sRT, Interval %.0f sec \n", prefix,
                                        clnt==0 ? "All clients" : client_address[clnt-1],
                                    rt_mode ? "Server" : "Client", time_diff);
    printf ("%s TPM ErrPm Errs  Init   Act    NO    PA    OS    DV    SL
%s\n",LEADING_SPACE,"cpu%");
    printf ("-------------------------------------------------------------------------------\n");
}


void print_info(char *prefix, client_status_t client_st)
{
      int i;

                printf("%s",prefix);
                for (i=0; i<strlen(LEADING_SPACE)-strlen(prefix);i++)
                        printf(" ");
      printf("%5.0f",client_st.tpm);
      /* print out err if any */
      if (client_st.err_diff != 0)
                 printf("%5.1f", (double)client_st.err_diff / client_st.interval * 60);
      else
                 printf("   -- ");
      if (client_st.total_errs !=0 )
                 printf("%4d", client_st.total_errs);
      else
                 printf("  -- ");
      printf("%5d%5d ",client_st.num_thrdsInit,client_st.num_thrds);
}

void print_errs(client_status_t **client_stat)
{
    int i, j;
    char *prefix="Total Errors     ";

    if (client_stat[0][0].total_errs>0) {
                printf("Total Errors     %4d\n",client_stat[0][0].total_errs);

        for (i=0; i<num_client; i++) {
                    if (client_stat[i+1][0].total_errs>0)
                            printf("%s",client_address[i]);
          for (j=0; j<strlen(prefix)-strlen(client_address[i]); j++)
                            printf(" ");
          printf("%4d\n",client_stat[i+1][0].total_errs);
        }
    }
}


void client_stat_init(client_status_t *client_stat[])
{
    int  i,j;

    /* allocate space first */
    for (i=0; i<=num_client; i++) {
        client_stat[i] = (client_status_t *)malloc((num_port+1)*sizeof(client_status_t));
                if (client_stat[i]==NULL) {
           fprintf(stderr,"client_stat_init: malloc failed\n");
           exit(1);
        }
        for (j=0; j<=num_port; j++) {
                    client_stat[i][j].total_errs = 0;
                    client_stat[i][j].cpu_usage_percent = 0;
                }
    }
}
```

ServerMon.c

```
/*
 * serverMon.c
 *
 * $Revision: 1.18 $
 * $Date: 1998/01/24 14:17:06 $
 * $Log:         serverEncina.c,v $
 *
 * $TALog: serverMon.c,v $
 * Revision 1.18  1998/01/24  14:17:06  oz
 * - User server name to identify server and name delivery file
 * - Use env variable HOME instead of /home/encina if HOME is set
 *
 * - Removed the machine list
 *   The server ID is computed from the first number found in the host name
 * [from r1.17 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]
 *
 * Revision 1.17  1998/01/23  21:59:00  oz
 * - In order to simplify the Encina TPCC code: Merge the four
 *   online transactions into 1 interface
 * - Moved all the scripts to a scripts subdirectory
 * - Removed unused files
 * [from r1.16 by delta oz-21671-TPCC-merge-online-transaction-interfaces, r1.1]
 *
 * Revision 1.16  1998/01/23  15:08:53  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.15 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 *
 *                 serverMon.c
 *
 *  Code that is monitor specific.
 *
 */

#define ORACLE

#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdarg.h>
#include <time.h>
#include <tmxa/tmxa.h>
#include <tc/tc.h>
#include <tpm/mon/mon_server.h>

#if defined (solaris)
#include <dce/pthread.h>
#else   /* solaris */
#include <pthread.h>
#endif  /* solaris */
#include <utils/trace.h>

#include "databuf.h"
```

```
#include "utilities.h"
#include "serverDebug.h"
#ifdef MULTIPLE_INTERFACE
#include "neworder.h"
#include "payment.h"
#include "stocklevel.h"
#include "orderstatus.h"
#else
#include "tpcc_trans.h"
#endif
#include "delivery.h"
#include "server.h"

#ifdef COLLECT_TIMESTAMPS
#  define FUNCTION_BEGIN(name, dataP) pre_oracle(name, &(dataP)->header)
#  define FUNCTION_END(name, dataP)   post_oracle(name, &(dataP)->header)
#else
#  define FUNCTION_BEGIN(name, dataP)
#  define FUNCTION_END(name, dataP)   ((dataP)->header.dtype = serverIdNumber)
#endif /* COLLECT_TIMESTAMPS */

#define CASECMP(x,y) strcasecmp(x,y) == 0

inModule("serverMon");

static void start_deferred_delivery_threads( void );
static void queue_delivery(delivery_data_t *dataP);
extern double gettime();

extern int server_null_test;

static void get_mon_server_env();

server_type_t server_type = mon_server;

char *tpcc_serverName = NULL;
int total_num_warehouses;
int num_deferred_dvry_threads = 2;
int num_worker_threads = 1;
int dvry_queue_size = 10000;
char oracle_home[256];

typedef struct {
   pthread_mutex_t          lock;
   pthread_cond_t           q_cond;
   pthread_cond_t           work_cond;

   int                                  num_waiters; /*  Number of new requests waiting   */

   int                                  head, tail;
   int                                  allocated;  /* Total size of the queue */
   int                                  size;   /* Num elements currently there  */
   delivery_data_t          *data;
} deferred_dvry_t;


static deferred_dvry_t deferred_dvry_data;
#define MAX_DVRY_QUEUE deferred_dvry_data.allocated

typedef struct {
   rpc_binding_handle_t mondHandle;
   int          handleValid;
} mondInfo_t;

mondInfo_t *mondInfo;

static void display_mon_env()
{
   char *env_str;
   char envMsg[64];

#define DISPLAY_ENV_VAR(var) \
    if ((env_str = getenv(var)) != NULL) { \
                UNCOND_EVENT("%s == '%s'\n", var, env_str); \
    } else { \
                UNCOND_EVENT("%s not set\n", var);  \
    }

   UNCOND_EVENT("TPCC Server display env.  ID: %d\n", serverIdNumber);

   /*
    *          For debugging purpose: have the first PA
    *  display the following information
    */


   if ((serverIdNumber & 0xff) == 0) {
    DISPLAY_ENV_VAR("RPC_SUPPORTED_PROTSEQS");
    DISPLAY_ENV_VAR("RPC_UNSUPPORTED_NETADDRS");
    DISPLAY_ENV_VAR("ENCINA_BINDING_TIMEOUT");
    DISPLAY_ENV_VAR("ENCINA_THREAD_POOL_QUEUE_LENGTH");
    DISPLAY_ENV_VAR("ENCINA_THREAD_POOL_QUEUE_LENGTH");
    DISPLAY_ENV_VAR("ENCINA_TPOOL_SIZE");
    DISPLAY_ENV_VAR("ENCINA_RPC_THREAD_STACK_SIZE");
   }
}


extern char *strchr(char *, int);

/*  get_server_index()  --      This is used for debug purposes only
 *
 *  Return the server index for this server.
 *  By convention, all the client machines hvae similar
 *  names with different numbers, as in client1 client2, ...
 *  If the convention is followed the server index is the first
 *  number found.  Otherwise, it is 0.
 *
 */
static int get_server_index()
{
  int i, ind;
  char host_name[128];
  if (0 == gethostname(host_name, sizeof(host_name))) {
    err_printf("Machine is on host %s\n", host_name);
    ind = strcspn(host_name, "0123456789");
    return(atol(host_name + ind));
  }

  return(0);
}

static parse_cmd_line(int argc, char *argv[], char **scheduling, int *interface_type)
{
    int nextInd = 1;
    char usageStr[128];
    int envRetrieval;
    if ((nextInd + 3) > argc) {
        sprintf(usageStr,
                "Not enough parameters.  Usage: %s[-no_db] interfaces schedulingPolicy envRetrievalFlag
[dvry=#] [debugFlag] [db:<rmName>] [cn=#].", argv[0]);

        fprintf(stderr, "%s\n", usageStr);
        mon_TerminateServer(usageStr);

    } else {
        if (strcmp(argv[nextInd], "-no_db") == 0) {
            server_null_test = 1;
            fprintf(stderr, " ---== NULL test ==---\n");
            nextInd++;
        }
        *interface_type = strtol(argv[nextInd++], NULL, 0);
        *scheduling = argv[nextInd++];
        envRetrieval = atoi(argv[nextInd++]);

        while (nextInd < argc) {
            if (strncmp(argv[nextInd], "db:", 3) == 0) {
                nextInd++;
            } else if (strncmp(argv[nextInd], "dvry=", 5) == 0) {
                num_deferred_dvry_threads = atol(argv[nextInd] + 5);
                if (num_deferred_dvry_threads < 0 || num_deferred_dvry_threads > 200)
                    num_deferred_dvry_threads = 1;
                nextInd++;
            } else if (strncmp(argv[nextInd], "dvryQ=", 6) == 0) {
                dvry_queue_size = atol(argv[nextInd] + 6);
                if (dvry_queue_size < 1 || dvry_queue_size > 200000)
                    dvry_queue_size = 10;
                nextInd++;
            } else {
                serverDebug = atol(argv[nextInd++]);
            }
        }
    }
}


static void set_scheduling(char  *scheduling)
{
    mon_paAccess_t paAccess;
    UNCOND_EVENT("Setting Scheduling Policy: %s\n", scheduling);

    if(CASECMP(scheduling, "MON_CONCURRENT_SHARED")){
        paAccess = MON_CONCURRENT_SHARED;
    } else if(CASECMP(scheduling, "MON_EXCLUSIVE")) {
        num_deferred_dvry_threads = 1;
        paAccess = MON_EXCLUSIVE;
    } else if(CASECMP(scheduling, "MON_SHARED")) {
                num_deferred_dvry_threads = 1;
        paAccess = MON_SHARED;
    } else {
                err_printf("Invalid Policy: '%s'\n", scheduling);
        mon_TerminateServer("Invalid scheduling policy specified.");
    }

    ENCINA_CALL("mon_SetSchedulingPolicy",
            mon_SetSchedulingPolicy(paAccess));
}

static void register_interfaces(int interface_type)
{
    char *env_str;
    int env_val;

    UNCOND_EVENT("Registering interfaces\n");

    num_worker_threads = 0;

#ifdef MULTIPLE_INTERFACE
```

```
    if (interface_type & NEWO_INTERFACE) {
                err_printf("Exporting neworder interface\n");
        ENCINA_CALL("mon_InitServerInterface",
                mon_InitServerInterface(MON_SERVER_INTERFACE(neworder,1,0)\
));
    }
    if (interface_type & PAYMENT_INTERFACE) {
                err_printf("Exporting payment interface\n");
        ENCINA_CALL("mon_InitServerInterface",
                mon_InitServerInterface(MON_SERVER_INTERFACE(payment,1,0))\
);
    }
    if (interface_type & ORDER_STAT_INTERFACE) {
        err_printf("Exporting order status interface\n");
        ENCINA_CALL("mon_InitServerInterface",
                mon_InitServerInterface(MON_SERVER_INTERFACE(orderstatus,1\
,0)));
    }
    if (interface_type & STOCK_INTERFACE) {
        err_printf("Exporting stock level interface\n");
        ENCINA_CALL("mon_InitServerInterface",
                mon_InitServerInterface(MON_SERVER_INTERFACE(stocklevel,1,\
0)));
    }
#else
    if (interface_type & ONLINE_INTERFACES) {
      err_printf("Exporting online interface\n");
        ENCINA_CALL("mon_InitServerInterface",
                mon_InitServerInterface(MON_SERVER_INTERFACE(tpccTrans,1,0\
)));
    }
#endif

    if (interface_type & ONLINE_INTERFACES) {
      if ((env_str = getenv("ENCINA_APPL_TPOOL_SIZE")) != NULL) {
        env_val = atol(env_str);
        if (env_val > 0 && env_val < 100)
          num_worker_threads += env_val;
        else
          num_worker_threads += 10;
      }
      if ((env_str = getenv("ENCINA_TPOOL_SIZE")) != NULL) {
        env_val = atol(env_str);
        if (env_val > 0 && env_val < 100)
          num_worker_threads += env_val;
        else
          num_worker_threads += 5;
      }
    }
    err_printf("Using %d worker threads\n", num_worker_threads);

    if (interface_type & DELIVERY_INTERFACE) {
      err_printf("Exporting delivery interface\n");
      if (num_deferred_dvry_threads > 0) {
        start_deferred_delivery_threads();
      }
      ENCINA_CALL("mon_InitServerInterface",
                                mon_InitServerInterface(MON_SERVER_INTERFACE(delivery,1,0)));
    } else {
      num_deferred_dvry_threads = 0;
    }
}

void main(argc,argv)
    int argc;
    char *argv[];
{
    int rc;
    int pa_num;
    char *scheduling = "";
    int rmId;
    char intermediary[256];
    extern int serverPid;
    int interface_type = ALL_INTERFACE;
    int num_cn;
    char dvryFileName[100];

    inFunction("server_Init");

/* hard code first for a quick test */
/* getenv didn't work, though we have ORACLE_HOME defined */
    strcpy(oracle_home,getenv("ORACLE_HOME"));
/*  strcpy(oracle_home, "/home/oracle815/app/oracle/product/8.1.5");  */
    err_printf("oracle_home is %s\n", oracle_home);

    serverPid = getpid();
    UNCOND_EVENT("TPCC Server Starting\n");

    /*  Use the top 8 bits of the serverIdNumber to store the server index */
    serverIdNumber = (get_server_index() & 0xff) * 1000;

    parse_cmd_line(argc, argv, &scheduling, &interface_type);

    display_mon_env();

    DEBUGP(("Debug level set at %d\n", serverDebug));
    mon_RetrieveEnable(FALSE);

    err_printf("Setting scheduling %s.\n", scheduling);
    set_scheduling(scheduling);
    err_printf(" Registering interfaces \n");
    register_interfaces(interface_type);

    err_printf("Calling mon_init\n");
    ENCINA_CALL("mon_InitServer", mon_InitServer());
    ENCINA_CALL("mon_SetHandleCacheRefreshInterval",
            mon_SetHandleCacheRefreshInterval(300));

    pa_num = mon_RetrievePaNum();
    tpcc_serverName = mon_RetrieveServerId();
    if (pa_num > 0)
                serverIdNumber += pa_num;
    err_printf("PA Number %d, serverId %d (%s)\n",
                        pa_num, serverIdNumber, tpcc_serverName);

    num_cn = num_deferred_dvry_threads + num_worker_threads;
    sprintf (dvryFileName, "/home/encina/runs/deliveries/tpccdel_%s",
        tpcc_serverName);
    if ((rc = get_db_ready("tpcc/tpcc", 0, num_cn, dvryFileName)) != 0) {
      WARNING("failed to open database %s: %d\n",
            "tpcc/tpcc", rc);
      err_printf("failed to open database %s: %d\n",
            "tpcc/tpcc", rc);
    }

    err_printf(">> Calling mon_BeginService()\n");

    ENCINA_CALL("mon_BeginService", mon_BeginService());

    fprintf(stderr, "mon_BeginService returned ... terminating\n");
    TPCexit();
}

/*
 *  The routine executed by the deferred delivery thread
 *
 *  Logic:
 *              Wait until there is a valid request in the deferred delivery data.
 *              After processing the request data_valid is set to FALSE
 *              (allowing new requests to be queued).
 *  This is a simple fixed size queue implemented in a cyclic array
 */
static void deferred_delivery()
{
    pthread_mutex_lock(&deferred_dvry_data.lock);

    while (1) {
                if (deferred_dvry_data.size > 0) {
                    /*
                     *  There is a request to be processed
                     */
                    int ind = deferred_dvry_data.head % MAX_DVRY_QUEUE;
                    delivery_data_t data = deferred_dvry_data.data[ind];

                    deferred_dvry_data.head ++;
                    deferred_dvry_data.size --;

                    if (deferred_dvry_data.num_waiters > 0)
                                pthread_cond_signal(&deferred_dvry_data.q_cond);

/*
                    if (deferred_dvry_data.head % 1000 == 0) {
                                err_printf("Processed %d deferred deliveries so far, queue size %d\n",
                                            deferred_dvry_data.head,
                                            deferred_dvry_data.size);
                    }
*/
                    if (deferred_dvry_data.head > deferred_dvry_data.tail) {
                                err_printf("Error: Deferred Queue: head %d > tail %d\n",
                                            deferred_dvry_data.head,
                                            deferred_dvry_data.tail);
                                continue;
                    }
                    pthread_mutex_unlock(&deferred_dvry_data.lock);

                    do_delivery(&data);

                    pthread_mutex_lock(&deferred_dvry_data.lock);

                } else {
                    /*
                     *  Wait for a request to be queued
                     */
                    DPRINT(("Deferred delivery waiting\n"));
                    pthread_cond_wait(&deferred_dvry_data.work_cond,
                                            &deferred_dvry_data.lock);
        }
    }
}

/*
 *  queue_delivery
 *
 *  Queue a delivery request to be processed in the background
 *  The queue is implemented as a simple queue of size 1.
 *  if data_valid is true: there is already a request waiting in the queue
 *  Sleep on a condition variable until the queue is empty.
 *  Once the queue is empty put the request in the queue, wake up the
```

```
 *  background thread and leave.
 */

static void queue_delivery(dataP)
  delivery_data_t *dataP;
{
   struct timezone tz;
   struct timeval now;
   int waited = 0;
   static int last_report_time = 0;

   pthread_mutex_lock(&deferred_dvry_data.lock);

   while (deferred_dvry_data.size >= MAX_DVRY_QUEUE) {
                /*  The request queue is full
                 *  Wait until a request is processed and removed from the queue.
                 */
                deferred_dvry_data.num_waiters ++;
                DPRINT((">> queue_delivery: %d waiters, size %d\n",
                            deferred_dvry_data.num_waiters, deferred_dvry_data.size));
                DPRINT(("Queue Delivery waiting, %d waiters\n",
                             deferred_dvry_data.num_waiters));
                pthread_cond_wait(&deferred_dvry_data.q_cond,
                                          &deferred_dvry_data.lock);
                deferred_dvry_data.num_waiters --;
                waited ++;
   }
   DPRINT(("Queuing delivery\n"));
   /*
    *  There is room in the queue.
    *  Enter the request and wake up the background thread
    */
   gettimeofday(&now, &tz);
#ifdef USE_ORACLE_GETTIME
   dataP->start_queue = gettime();
#else
   dataP->start_queue = (double)now.tv_sec + (now.tv_usec / 1000000.0);
#endif

   deferred_dvry_data.size++;
   deferred_dvry_data.data[deferred_dvry_data.tail % MAX_DVRY_QUEUE] = *dataP;
   deferred_dvry_data.tail++;
   pthread_cond_signal(&deferred_dvry_data.work_cond);
   if (now.tv_sec - last_report_time > 59) {
                err_printf("queue_delivery - %d waiters, size %d\n",
            deferred_dvry_data.num_waiters, deferred_dvry_data.size);
                last_report_time = now.tv_sec;
   }
   pthread_mutex_unlock(&deferred_dvry_data.lock);

   if (waited) err_printf(">> queue_delivery waited %d times\n", waited);
   dataP->header.returncode = TPCC_SUCCESS;
}

/*
 * start_deferred_delivery_threads
 *
 *                Initialize the deferred delivery data structure and start
 *                a background thread to process the delivery requests
 */
static void start_deferred_delivery_threads()
{
   pthread_t thread;
   int i;
   int rc;

   pthread_mutex_init(&deferred_dvry_data.lock, pthread_mutexattr_default);
   pthread_cond_init(&deferred_dvry_data.work_cond, pthread_condattr_default);
   pthread_cond_init(&deferred_dvry_data.q_cond, pthread_condattr_default);
   deferred_dvry_data.num_waiters = 0;
   deferred_dvry_data.head = 0;
   deferred_dvry_data.tail = 0;
   deferred_dvry_data.allocated = dvry_queue_size;
   deferred_dvry_data.data =
                (delivery_data_t *)malloc(dvry_queue_size * sizeof(delivery_data_t));

   /*
    *  Create the background delivery thread.
    */
   err_printf("Starting %d deferred delivery threads, queue size %d\n",
                    num_deferred_dvry_threads,
                    dvry_queue_size);
   for (i=0; i<num_deferred_dvry_threads; i++) {
      if ((rc = pthread_create(&thread,
                                            pthread_attr_default,
                                            (pthread_startroutine_t)deferred_delivery,
                                            (pthread_addr_t)0)) != 0) {
                WARNING("Failed to create delivery thread rc=%d\n", rc);
                exit(1);
      }
      (void)pthread_detach(&thread);
   }
}

extern char *strchr(char *, int);

void exit_program(code)
    int code;
{
   char errMsg[55];
   sprintf(errMsg, "exit_program called with code %d", code);
   fprintf(stderr, "%s\n", errMsg);

   TPCexit();

   mon_TerminateServer(errMsg);
}
#ifdef            COLLECT_TIMESTAMPS
static void pre_oracle(name, headerP)
     char *name;
     data_header *headerP;
{
  struct timeval tp;
  struct timezone tz;
  DPRINT(("> %s", name));
  gettimeofday(&tp, &tz);
  headerP->start_time.sec = tp.tv_sec;
  headerP->start_time.usec = tp.tv_usec;
}

static void post_oracle(name, headerP)
     data_header *headerP;
     char *name;
{
  struct timeval tp;
  struct timezone tz;
  DPRINT(("< %s\n", name));
  gettimeofday(&tp, &tz);
  headerP->end_time.sec = tp.tv_sec;
  headerP->end_time.usec = tp.tv_usec;
  headerP->dtype = serverIdNumber;
}
#endif  /* COLLECT_TIMESTAMPS */

/*
 * ------------------  The following are the entry points
 *                                                  for the RPCs arriving at the Server
 */

void impTPCCDbInfo(dataP,trpcStatus)
  dbInfo_data_t *dataP;
  trpc_status_t  *trpcStatus;
{
    UNCOND_EVENT("> impTPCCDbInfo");
    dataP->server_id = serverIdNumber;
}



void expTPCCDbInfo(handle, dataP,trpcStatus)
  trpc_handle_t  handle;
  dbInfo_data_t *dataP;
  trpc_status_t  *trpcStatus;
{
    impTPCCDbInfo(dataP,trpcStatus);
}

void expTPCCNOInfo(handle, dataP,trpcStatus)
  trpc_handle_t  handle;
  dbInfo_data_t *dataP;
  trpc_status_t  *trpcStatus;
{
    impTPCCNOInfo(dataP,trpcStatus);
}

void impTPCCNOInfo(dataP,trpcStatus)
  dbInfo_data_t *dataP;
  trpc_status_t  *trpcStatus;
{
    impTPCCDbInfo(dataP,trpcStatus);
}

void expTPCCPayInfo(handle, dataP,trpcStatus)
  trpc_handle_t  handle;
  dbInfo_data_t *dataP;
  trpc_status_t  *trpcStatus;
{
    impTPCCDbInfo(dataP,trpcStatus);
}

void expTPCCOSInfo(handle, dataP,trpcStatus)
  trpc_handle_t  handle;
  dbInfo_data_t *dataP;
  trpc_status_t  *trpcStatus;
{
    impTPCCDbInfo(dataP,trpcStatus);
}

void expTPCCDvryInfo(handle, dataP,trpcStatus)
  trpc_handle_t  handle;
  dbInfo_data_t *dataP;
  trpc_status_t  *trpcStatus;
{
    impTPCCDbInfo(dataP,trpcStatus);
}

void expTPCCSLInfo(handle, dataP,trpcStatus)
  trpc_handle_t  handle;
```

```
 dbInfo_data_t *dataP;
 trpc_status_t  *trpcStatus;
{
   impTPCCDbInfo(dataP,trpcStatus);
}




void impTPCCNewOrder(dataP,trpcStatus)
 newOrder_data_t *dataP;
 trpc_status_t  *trpcStatus;
{
   static int numCalls = 0;
   FUNCTION_BEGIN("NewOrder", dataP);
   do_new_order(dataP, 0);

   if ((dataP->header.returncode != TPCC_SUCCESS) &&
                (dataP->header.returncode != INVALID_NEWO)) {
    logprintf("< impTPCCNewOrder; rc=%d, sql=%d, isam=%d\n",
                                dataP->header.returncode,
                                dataP->header.sql_code,
                                dataP->header.isam_code);
   } else if (dataP->header.returncode == INVALID_NEWO) {
                DPRINT(("< impTPCCNewOrder INVALID_NEWO\n"));
   }
   if (++numCalls % 1000 == 0) {
                err_printf("impTPCCNewOrder so far %d\n", numCalls);
   }
   FUNCTION_END("NewOrder", dataP);
}

void expTPCCNewOrder(handle,dataP,trpcStatus)
 trpc_handle_t   handle;
 newOrder_data_t *dataP;
 trpc_status_t  *trpcStatus;
{
   impTPCCNewOrder(dataP,trpcStatus);
}

void impTPCCPayment(dataP,trpcStatus)
 payment_data_t *dataP;
 trpc_status_t  *trpcStatus;
{
   static int numCalls = 0;
   FUNCTION_BEGIN("Payment", dataP);
   do_payment(dataP, 0);

   if (dataP->header.returncode != TPCC_SUCCESS){
    logprintf("< impTPCCPayment; rc=%d, sql=%d, isam=%d\n",
                                dataP->header.returncode,
                                dataP->header.sql_code,
                                dataP->header.isam_code);
   }
   if (++numCalls % 1000 == 0) {
      err_printf("impTPCCPayment so far %d\n", numCalls);
   }
   FUNCTION_END("Payment", dataP);
}

void expTPCCPayment(handle,dataP,trpcStatus)
 trpc_handle_t   handle;
 payment_data_t *dataP;
 trpc_status_t  *trpcStatus;
{
   impTPCCPayment(dataP,trpcStatus);
}

void impTPCCOrderStatus(dataP,trpcStatus)
 orderStatus_data_t *dataP;
 trpc_status_t  *trpcStatus;
{
   FUNCTION_BEGIN("OrderStatus", dataP);
   do_order_status(dataP);

   if (dataP->header.returncode != TPCC_SUCCESS){
    logprintf("< impTPCCOrderStatus; rc=%d, sql=%d, isam=%d\n",
                                dataP->header.returncode,
                                dataP->header.sql_code,
                                dataP->header.isam_code);
   }
   FUNCTION_END("OrderStatus", dataP);
}

void expTPCCOrderStatus(handle,dataP,trpcStatus)
 trpc_handle_t handle;
 orderStatus_data_t *dataP;
 trpc_status_t   *trpcStatus;
{
  impTPCCOrderStatus(dataP,trpcStatus);
}


void impTPCCStockLevel(dataP,trpcStatus)
 stockLevel_data_t *dataP;
 trpc_status_t   *trpcStatus;
{
   FUNCTION_BEGIN("StockLevel", dataP);
   do_stock_level(dataP);
   if (dataP->header.returncode != TPCC_SUCCESS){
    logprintf("< impTPCCStockLevel; rc=%d, sql=%d, isam=%d\n",
                                dataP->header.returncode,
                                dataP->header.sql_code,
                                dataP->header.isam_code);
   }
   FUNCTION_END("StockLevel", dataP);
}

void expTPCCStockLevel(handle,dataP,trpcStatus)
 trpc_handle_t handle;
 stockLevel_data_t *dataP;
 trpc_status_t  *trpcStatus;
{
   impTPCCStockLevel(dataP,trpcStatus);
}

void impTPCCDelivery(dataP,trpcStatus)
 delivery_data_t *dataP;
 trpc_status_t   *trpcStatus;
{

   FUNCTION_BEGIN("DELIVERY", dataP);
   if (num_deferred_dvry_threads > 0) {
      queue_delivery(dataP);
   } else {
      do_delivery(dataP);
   }


   if (dataP->header.returncode != TPCC_SUCCESS){
    logprintf("< impTPCCDelivery; rc=%d, sql=%d, isam=%d\n",
                                dataP->header.returncode,
                                dataP->header.sql_code,
                                dataP->header.isam_code);
   }
   FUNCTION_END("DELIVERY", dataP);
}

void expTPCCDelivery(handle,dataP,trpcStatus)
 trpc_handle_t handle;
 delivery_data_t *dataP;
 trpc_status_t  *trpcStatus;
{
   impTPCCDelivery(dataP,trpcStatus);
}
```

Socket.C

```
/*****************************************************************************/
/* Socket.C                                  Audit: 05/30/96*/
/*****************************************************************************/

/* (C) 1993-1997 IBM Corporaton*/

static char *rcsid="$Id: socket.C,v 1.1 1999/02/22 06:30:54  channui Exp $";

#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#define COMPAT_43
#include <sys/socket.h>
#undef COMPAT_43
#include <sys/select.h>
#include <sys/socketvar.h>
#include <netinet/tcp.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>
#include <memory.h>
#include <strings.h>
#include "rte.h"
#include "socket.h"
#include "data.h"
#include "packet.h"
#include "misc.h"

extern FILE *iprintfile;

//int packet_number = 0;

int Socket::read_wait(void *buffer, int size) {
   int rc;
   int count = 0;
   // TODO:  Use select to wait for input
   while (1) {
                fd_set read_set;
                rc = read((char *)buffer+count, size - count);
#if 0
                if (rc != size-count) {
                   iprint(__FILE__,__LINE__,IPRINT_ERROR,  "But that's ok I'm in read_wait\n");
                }
#endif
                if (rc == 0)
```

```
                    return -2;
        count += rc;
//                  iprint (IPRINT_TRACE, ".%d", count); fflush(iprintfile);
                    if (count >= size)
                        break;
                    // Wait until we get more info
                    FD_ZERO (&read_set);
                    FD_SET  (fd, &read_set);
                    rc = select(fd+1, &read_set, NULL, NULL, NULL);
    }
    return count;
}

int Socket::read(void *buffer, int size) {
    int rc;
    errno = 0;
    do {
                    rc = ::read(fd, buffer, size);
    } while (rc <= 0 && errno == EINTR);
#if 0
    if (rc != size) {
                    iprint(__FILE__,__LINE__,IPRINT_ERROR,  "Only read %d of %d bytes!
(errno=%d(%s))\n", rc, size, errno, strerror(errno));
    }
#endif
    return rc;
}

int Socket::send(void *buffer, int size) {
    int rc;
    errno = 0;
    do {
                    rc = ::write(fd, buffer, size);
    } while (rc <= 0 && errno == EINTR);
#if 0
    if (rc != size) {
                    iprint(__FILE__,__LINE__,IPRINT_ERROR,  "Only wrote %d of %d bytes!
(errno=%d(%s))\n", rc, size, errno, strerror(errno));
    }
#endif
    return rc;
}

int Socket::send(char *buffer) {
    int rc;
    errno = 0;
    do {
                    rc = ::write(fd, buffer, strlen(buffer));
    } while (rc <= 0 && errno == EINTR);
#if 0
    if (rc != strlen(buffer)) {
                    iprint(__FILE__,__LINE__,IPRINT_ERROR,  "Only rote %d of %d bytes!
(errno=%d(%s))\n", rc, strlen(buffer),errno, strerror(errno));
    }
#endif
    return rc;
}

int Socket::send(Packet *pack) {
    int  checksum = 0, i;
    char *ptr = (char *)pack;
    send (pack, pack->packet_length);

    if (pack->packet_length > MAX_PACKET_LEN) {
                    iprint(IPRINT_ERROR, "Packet Length (%d) > MAX_PACKET_LEN (send)",
                        pack->packet_length);
                    abort();
    }

//  iprint(IPRINT_TRACE, "Writing Packet: length=%d, num=%d\n", pack->packet_length,
packet_number); fflush(iprintfile);
    for (i = 0; i < pack->packet_length; i++)
                    checksum += *ptr++;
    send(&checksum, sizeof(checksum));
//  send(&packet_number, sizeof(packet_number));
//  packet_number++;
    return RTE_OK;
}
int Socket::read(Packet *pack) {
    int rc1, rc2, rc3;
    int toread, i;
    int checksum;
    char *ptr = (char *)pack;

//  iprint(IPRINT_TRACE, "Reading Packet: "); fflush(iprintfile);
    rc1 =read_wait(&pack->packet_length, sizeof(pack->packet_length));
    if (rc1 <= 0) {
                    iprint(__FILE__,__LINE__,IPRINT_TRACE,  "ERROR: %s\n", strerror(errno));
                    return RTE_ERROR;
    }

    if (pack->packet_length > MAX_PACKET_LEN) {
                    iprint(IPRINT_ERROR, "Packet Length (%d) from > MAX_PACKET_LEN (read)",
                        pack->packet_length);
                    abort();
    }

//  iprint(IPRINT_TRACE, "length=%d, ", pack->packet_length); fflush(iprintfile);
    toread = pack->packet_length - sizeof(pack->packet_length);
    rc2 =read_wait(&pack->type, toread);
    if (rc2 <= 0) {
                    iprint(__FILE__,__LINE__,IPRINT_TRACE,  "ERROR: %s\n", strerror(errno));
                    return RTE_ERROR;
    }
//  iprint(IPRINT_TRACE, "read=%d\n", rc2+rc1); fflush(iprintfile);

    rc3 =read_wait(&checksum, sizeof(int));
    if (rc3 <= 0) {
                    iprint(__FILE__,__LINE__,IPRINT_TRACE,  "ERROR: %s\n", strerror(errno));
                    return RTE_ERROR;
    }

    for (i = 0; i < pack->packet_length; i++)
                    checksum -= *ptr++;
    if (checksum)
                    iprint(IPRINT_ERROR, "  **** CHECKSUM ERROR!\n");
//  read_wait(&packet_number, sizeof(packet_number));
//  iprint(IPRINT_TRACE, "packet_number=%d\n", packet_number); fflush(iprintfile);

    return RTE_OK;
}

int Socket::send(Packet &pack) {
    send(&pack);
    return RTE_OK;
}

Socket::Socket(char *host, int port) {
    fd = -1;
    open(host, port);
}

int Socket::close() {
    int rc = 0;
    if ( fd >=0 )
                    rc= ::close(fd);
    fd = -1;
    return rc;
}

int Socket::open(char *host, int port) {
    unsigned long       inaddr;
    struct sockaddr_in  addr,
                        mysock;
    struct hostent      *he;
    struct linger linger;
    int                 tcp_nodelay = 1,
                        retval = 0,
                        socknum = 0;


    if (fd >= 0)
                    ::close (fd);

    memset (&addr, '\0', sizeof(addr));
    memset (&mysock, '\0', sizeof(addr));
    addr.sin_family = AF_INET;
    mysock.sin_family = AF_INET;
    mysock.sin_addr.s_addr  = htonl(INADDR_ANY);

    /* Resolve host's internet address here */
    if ((inaddr = inet_addr(host)) != INADDR_NONE) /* dotted decimal */
        memcpy (&addr.sin_addr, &inaddr, sizeof(addr.sin_addr));
    else if ((he = gethostbyname(host)) != NULL)  /* resolve name */
        memcpy (&addr.sin_addr, he->h_addr, sizeof(addr.sin_addr));
    else {                                  /* Can't resolve name */
                    fd = -1;
                    iprint(IPRINT_TRACE, "\t Can't resolve name\n");
        return ERROR;
    }

    /* Put port number into address */
    addr.sin_port = htons(port);

    /* Create a socket to use */
    if ((fd = socket (AF_INET, SOCK_STREAM, 0)) < 0) {
                    fd = -1;
        return ERROR;
    }

    /* Name the socket.
     * For now, bind to a random socket, where 10000 < socket < 65535. When
     * we become multi-threaded, we can just use a variable.
     */
    retval = EADDRINUSE;
    /* There's a chance that this loop might not exit, but I think it's
     * pretty unlikely
     */
    while ((retval == EADDRINUSE) ||
           (retval == EACCES) || /* Should never happen */
           (retval == EADDRNOTAVAIL)) {
        /* srand() has already been called (in init_random()), so hopefully
         * clients won't fight over ports.  If so, it'll need to be worked
         * around.
         */
        mysock.sin_port = (rand() + 10000) % 65000; /* not thread-safe */
        retval = bind(fd, (struct sockaddr *)&mysock, sizeof(mysock));
    }
```

```
  /* Flush remaining data when socket is closed */
  linger.l_onoff = 1;
  linger.l_linger = 0;
  if (setsockopt (fd, SOL_SOCKET, SO_LINGER, (char *)&linger, sizeof(linger))) {
     ::close (fd);
                fd = -1;
     return ERROR;
  }

  /* When using TCP to exchange request/response messages, the application */
  /* must use setsockopt to turn on the TCP_NODELAY option. This cause TCP */
  /* to send the message immediatly (within the constraints of the sliding */
  /* window), even though it is less than the MTU-size. Otherwise, TCP     */
  /* would wait for up to 200 ms for more data to send before transmitting*/
  /* the message. The consequences for performance are obvious.            */
  if (setsockopt(fd, IPPROTO_TCP, TCP_NODELAY, (char *)&tcp_nodelay, sizeof(tcp_nodelay)) <
0) {
     ::close (fd);
     iprint(IPRINT_TRACE, "\t Unable to set TCP_NODELAY on\n");
                fd = -1;
     return ERROR;
  }


  /* Connect to remote host*/
  if (connect (fd, (struct sockaddr *) &addr, sizeof (addr)) < 0) {
     ::close (fd);
                fd = -1;
     return ERROR;
  }
  return RTE_OK;
}


int Socket::ack(int return_code) {
   Packet_acknowledge packet;
   packet.packet_length = sizeof(packet);
   packet.type        = PACKET_TYPE_ACK;
   packet.retcode     = return_code;
   send(&packet);
   return RTE_OK;
}

int Socket::ack() {
   return ack(PA_OK);
```

Stocklevel.c

```
#ifndef TRANSARC_stocklevel_h
#define TRANSARC_stocklevel_h

#include <trpc/trpc.h>
#include "_stocklevel.h"

#include <encina/c_prologue.h>

#define stocklevel_v1_0_c_ifspec            _stocklevel_v1_0_c_ifspec
#define stocklevel_v1_0_s_ifspec            _stocklevel_v1_0_s_ifspec

typedef struct stocklevel_v1_0_epv {
void (ENCINA_DCE_FUNCTION_PTR_CALLING *impTPCCStockLevel) (
#ifdef IDL_PROTOTYPES

                                stockLevel_data_t *dataP,
                                trpc_status_t *trpcStatus
#endif
);

} stocklevel_v1_0_epv_t;

extern void impTPCCStockLevel (
#ifdef IDL_PROTOTYPES

                                stockLevel_data_t *dataP,
                                trpc_status_t *trpcStatus
#endif
);

trpc_handle_tmon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

void            mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_handle_ttrpcHandle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

trpc_handle_tmon_handle_t_tranBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

void            mon_handle_t_tranUnBind(
#ifdef IDL_PROTOTYPES
                mon_handle_t                handle,
                trpc_handle_ttrpcHandle,
                trpc_tranInfo_t             *tranInfoP,
                trpc_ifSpec_t *ifSpecP
#endif
                );

extern stocklevel_v1_0_epv_t                stocklevel_v1_0_client_epv;
extern _stocklevel_v1_0_epv_t               stocklevel_v1_0_manager_epv;
extern rpc_mgr_epv_t                        stocklevel_v1_0_mgr_epv;

#include <encina/c_epilogue.h>
#endif /* TRANSARC_stocklevel_h */
```

tpcc.h

```
#if !defined(TPCC_H_INCLUDED)
#define TPCC_H_INCLUDED
/*****************************************************************************/
/*                                                                           */
/*      File: tpcc.h                                                         */
/*      created: 8-26-91                                                     */
/*                                                                           */
/*      program description:                                                 */
/*                                                                           */
/*      This module contains global variables and data definitions           */
/*      for the tpcc application.                                            */
/*                                                                           */
/*****************************************************************************/

#include "../tpcc_type.h"

#define TPCCH

/*-------------------------------------------------------*/
/*              Global numbers, constants,...            */
/*-------------------------------------------------------*/

#define INVALID_ITEM            100
#define TRAN_OK                 0
#define REMOTE_WAREHOUSE        17

#define FORM_DATE               1
#define FORM_DATETIME           2

#define MAX_ITEMS 15

/*-------------------------------------------------------*/
/*              transaction structures                   */
/*-------------------------------------------------------*/

typedef orderStatus_data_t OrderStatus_data;
typedef newOrder_data_t NewOrder_data;
typedef stockLevel_data_t StockLevel_data;
typedef delivery_data_t Delivery_data;
typedef payment_data_t Payment_data;

/*****************************************************************************
  Compatibility for older.sqc files
 *****************************************************************************/
#define s_C_BALANCE c_balance
#define s_C_CITY c_city
#define s_C_CREDIT c_credit
#define s_C_CREDIT_LIM c_credit_lim
#define s_C_DATA c_data
#define s_C_DISCOUNT c_discount
#define s_C_D_ID c_d_id
#define s_C_FIRST c_first
#define s_C_ID c_id
#define s_C_LAST c_last
#define s_C_MIDDLE c_middle
#define s_C_PHONE c_phone
#define s_C_SINCE c_date
#define s_C_STATE c_state
#define s_C_STREET_1 c_street_1
#define s_C_STREET_2 c_street_2
#define s_C_W_ID c_w_id
#define s_C_ZIP c_zip
#define s_D_CITY d_city
```

```
#define s_D_ID d_id
#define s_D_STATE d_state
#define s_D_STREET_1 d_street_1
#define s_D_STREET_2 d_street_2
#define s_D_TAX d_tax
#define s_D_ZIP d_zip
#define s_H_AMOUNT h_amount
#define s_H_DATE pay_date
#define s_I_NAME name_i
#define s_I_PRICE price
#define s_OL_AMOUNT ol_amount
#define s_OL_DELIVERY_D delivery_date
#define s_OL_I_ID ol_i_id
#define s_OL_QUANTITY ol_quantity
#define s_OL_SUPPLY_W_ID ol_supply_w_id
#define s_O_CARRIER_ID o_carrier_id
#define s_O_ENTRY_D entry_date
#define s_O_ID o_id
#define s_O_OL_CNT o_ol_cnt
#define s_S_QUANTITY s_quantity
#define S_QUANTITY quantity
#define s_W_CITY w_city
#define s_W_ID w_id
#define s_W_STATE w_state
#define s_W_STREET_1 w_street_1
#define s_W_STREET_2 w_street_2
#define s_W_TAX w_tax
#define s_W_ZIP w_zip
#define s_all_local o_all_local
#define s_brand_generic brand_generic
#define s_exec_status exec_status
#define s_low_stock stock_count
#define s_ol_cnt o_ol_cnt
#define s_queued_time queued_time
#define s_status_line statusline
#define s_threshold threshold
#define s_total_amount total
#define s_transtatus header.returncode

#if 0
#define NEWORDER_SERVICE    "NEWORD"
#define PAYMENT_SERVICE     "PAYMENT"
#define DELIVERY_SERVICE    "DELIVERY"
#define STOCKLEVEL_SERVICE  "STOCKLEV"
#define ORDERSTATUS_SERVICE "ORDSTAT"
#else
#define NEWORDER_SERVICE    "neword_sql"
#define PAYMENT_SERVICE     "payment_sql"
#define DELIVERY_SERVICE    "delivery_sql"
#define STOCKLEVEL_SERVICE  "stocklev_sql"
#define ORDERSTATUS_SERVICE "ordstat_sql"
#endif

#endif /* TPCC_H_INCLUDED */
```

Tpcc_monitor.c

```
/*
*                tpcc_monitor.c
*
* $Revision: 1.5 $
* $Date: 1998/07/08 18:15:42 $
* $Log:             $
*
*
* $TALog: tpcc_monitor.c,v $
* Revision 1.5  1998/07/08  18:15:42   wenjian
* Minor change for print_header().
* [from r1.4 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.6]
*
* Revision 1.4  1998/07/02  18:28:52   wenjian
* - Change the display to table format.
* - Display more information of client process.
* - Rewrite code for better modularity.
* [from r1.3 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.5]
*
* Revision 1.1  1998/04/29  19:47:43   wenjian
* - Copy open_socket from terminal.c
* - Communicate with tpcc_client, calculate the TPM and response time,
*   and display the information
* [added by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.1]
*
* Revision 1.18  1998/02/17  22:07:07   wenjian
* - Add head files
* - Define macros to dead with the different function names on NT
* [from r1.17 by delta wenjian-21750-TPCC-changes-for-porting-on-NT,r1.1]
*
*/

/*
*   This is the monitor program for checking status of each client machine
*/

#include <sys/types.h>
#include <sys/stat.h>
#ifndef WIN32
#include <sys/select.h>
#include <termios.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/un.h>
#else /* WIN32 */
#include <io.h>
#include <process.h>
#include <winsock.h>
#include "tran_stat.h"
#endif /* WIN32 */
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#if defined (solaris)
#include <dce/pthread.h>
#else   /* solaris */
#include <pthread.h>
#endif  /* solaris */
#include "terminal.h"
#include "client_utils.h"
#include "tpcc_type.h"
#include "tpcc_monitor.h"

#define BIND_LOCAL

#ifdef WIN32
#define EADDRINUSE      WSAEADDRINUSE
#define SLEEP(A)        Sleep(A*1000)
#define RANDOM                  rand
#define SRANDOM                 srand
#define stat                    _stat
#define close                   _close
#define read(A,B,C) recv(A,B,C,0)
#define write(A,B,C) send(A,B,C,0)
#define STRCMP          _stricmp
#else
#define SLEEP(A)        sleep(A)
#define RANDOM                  random
#define SRANDOM                 srandom
#define STRCMP          strcasecmp
#endif

#define MAX_PORTS 20
#define MAX_CLIENTS 30
#define MAX_FILE_NAME 30

#define AUTOMATIC    0
#define MANUAL       1
#define ALL      2
#define INDIVIDUAL   3

#define QUIT     10001
#define MODE_CHANGE  10002
#define WRONG_INPUT  10003
#define REFRESH      10004
#define HELP         10005
#define LEADING_SPACE "          "

#define EXIT(code) exit_program(code)
#define TRIGGERc '\021'

static void write_to_all_clients(int clnt_id, int *out, char *buffer, int len);
static void write_to_client(int out, char *buffer, int len);
static int read_from_all_clients(int clnt_id, int *sock, int initialized);
static int read_from_client(int sock, char *buffer, int buf_len);
void   print_header(char *prefix, int rt_mode, int display, double interval);
void   print_info(char *, client_status_t client_st);
static void show_info();
static int open_socket(int id, char *address);
static void exit_program(int err);
void   one_print_rt_avg(client_status_t, int, int);
int    get_next_choice();
void   get_client_info(client_status_t *, char *, int, int);
void   clients_status_summary(client_status_t  **client_stat);
void   display_client(int clnt, client_status_t **client_stat);
void   display_all_clients(client_status_t *client_stat[]);
void   print_errs(client_status_t **client_stat);
void   set_client_st_rt(client_status_t  *, total_tran_count_t *, total_tran_count_t *);
void   set_client_st(client_status_t  *, double, int, char *, double, int, int, double);
void   client_stat_init(client_status_t **);

/* global variable */
static char *sys_errlist[256];
int    user_id;         /*  this terminal's ID (what file to write)  */
char   *user_code = "T"; /*  Prefix for output to identify this
                          *  process as a client or a terminal
                         */
FILE   *logtpcc = NULL;   /*  The log file for debug output          */
int    debug = 0;                /*  Debug mode -- set to 0 for real runs    */
int    initialized=0;    /*  The variable for initialization        */
int    client_id;                /*  which client to talk to                                 */
```

```
int    next_port_to_try;
char   client_address[MAX_CLIENTS+1][MAX_FILE_NAME];
int    num_client;
int    client_port;
int    num_port;
int    input_mode=AUTOMATIC;              /*  control auto or manual input mode*/
int    display_mode=ALL;          /*  control display of client(s) */
int    rt_mode=0;                 /*  0-client RT, 1-server RT */
struct timeval timeout={60,0};
FILE   *logfd = NULL;

/* the followings are global vars to summarize the TPM and RTs for all clnts */
client_status_t  *client_stat[MAX_CLIENTS+1];

/*
 *  Open a socket to the client whose id is 'id'
 */
static int open_socket(port, address)
  int port;
  char *address;
{
    int retries = 0;
    int sock;   /* socket */
    struct sockaddr_in sockAddrIn;
    struct sockaddr *sockAddrP;
    int sockAddrLen;
    int ip_address;

    if (getenv("TPCC_DEBUG") != NULL)
      debug = 1;

    /* now open a stream socket */
    ip_address = inet_addr(address);
    if (ip_address == -1) {
                struct hostent *hp;
                struct in_addr in;
                char *buf;
                hp = gethostbyname(address);
                if (hp == NULL) {
                   err_printf("Could not resolve %s\n", address);
                   exit(2);
                }
                buf = hp->h_addr_list[0];
                (void) memcpy(&in.s_addr, buf, sizeof (in.s_addr));
                ip_address = inet_addr(inet_ntoa(in));
                if (ip_address == -1) {
                   err_printf("host %s resolved to %s which failed.  OOPS\n",
                                address, inet_ntoa(in));
                   exit(2);
                }
    }

#ifdef WIN32
  if (1) {
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;

    wVersionRequested = MAKEWORD( 2, 2 );

    err = WSAStartup( wVersionRequested, &wsaData );
    if ( err != 0 ) {
                printf("could not find a usable WinSock DLL\n");
                exit(2);
    }

    /* Confirm that the WinSock DLL supports 2.0.*/
    /* Note that if the DLL supports versions greater    */
    /* than 2.2 in addition to 2.2, it will still return */
    /* 2.2 in wVersion since that is the version we      */
    /* requested.                                        */

    if ( LOBYTE( wsaData.wVersion ) != 2 ||
                 HIBYTE( wsaData.wVersion ) != 2 ) {
                printf("could not find a usable WinSock DLL\n");
                WSACleanup();
       exit(2);
    }
}
#endif

    /* The WinSock DLL is acceptable. Proceed. */
    sock = socket(AF_INET, SOCK_STREAM, 0);
/*    printf("open_sock: socket is %d\n", sock); */
    if (sock == -1) {
                err_printf("open_socket: failed (errno=%s (%d)).\n",
                              sys_errlist[errno], errno);
                exit(2);
    }
#ifdef BIND_LOCAL
    /*  Bind the socket (There is a limit of less than
     *  4000 on ephemeral sockets
     */
    while (1) {
       int rc;
       int port = next_port_to_try ++;
       sockAddrIn.sin_family = AF_INET;
       sockAddrIn.sin_port = port;
       sockAddrIn.sin_addr.s_addr = INADDR_ANY;
       rc = bind(sock,
                                (struct sockaddr *)&sockAddrIn,
                sizeof(sockAddrIn));
       if (rc == 0) {
                    DPRINT(("Bound socket %d to port %d\n", sock, port));
                    break;
                }
                if (errno != EADDRINUSE) {
          err_printf("open_socket: bind to port %d failed errno=%s(%d), sock %d, retry %d\n",
                                port,
                    sys_errlist[errno], errno, sock, retries);
                    exit(13);
       }
                if (port > 65000) {
                    err_printf("open_socket: bind to port %d failed - too many portstried\n",
                                port);
                    exit(13);
                }
    }
#endif

    sockAddrIn.sin_family = AF_INET;
    sockAddrIn.sin_port = port;
    sockAddrIn.sin_addr.s_addr = ip_address;
    sockAddrP = (struct sockaddr *)&sockAddrIn;
    sockAddrLen = sizeof(sockAddrIn);

    while(connect(sock, sockAddrP, sockAddrLen) == -1) {
                /*
                 *  Try up to 15 times to connect to the socket
                 *  before giving up
                 */
                if (retries++ < 15) {
                    SLEEP(1+ retries/3);
                    err_printf("open_socket: connect failed errno=%s(%d), sock %d, retry %d\n",
                                sys_errlist[errno], errno, sock, retries);
                } else {
                    err_printf("open_socket: connect failed errno=%s(%d)\n",
                                sys_errlist[errno], errno);
                    close(sock);
                    EXIT(1);
                }
    }
    return(sock);
}

#ifndef WIN32
static void set_term_for_input(int in_fd, struct termios *save_term)
{
    struct termios tc_new;
    int rc;
    if ((rc = tcgetattr(in_fd, save_term)) < 0) {
                return;
    }

    tc_new = *save_term;

#if 1
    tc_new.c_lflag &= ~(ECHO | ICANON); /* echo off, canonical mode off*/

    tc_new.c_cc[VMIN] = 1;    /* Case B: 1 byte at a time, no timer*/
    tc_new.c_cc[VTIME] = 0;
#else
    /* New TTY stuff */
    tc_new.c_cflag &= ~(CBAUD | HUPCL);
    tc_new.c_cflag &= ~(CBAUD);
    tc_new.c_cflag |= CLOCAL | B38400;
    tc_new.c_iflag = IGNBRK;
    tc_new.c_iflag &= ~(IXON|IXOFF|IXANY);
    tc_new.c_oflag = tc_new.c_lflag = 0;
    tc_new.c_cc[VMIN] = 1;
    tc_new.c_cc[VTIME] = 0;
#endif

    if (tcsetattr(in_fd, TCSANOW, &tc_new) < 0)
                return;
}
#endif

static void get_term_env(argc, argv)
                int argc;
                char *argv[];

{
    int  i, next_arg;
    char *filename, *log_filename;
    FILE *inFile;

    if (argc < 4) {
                fprintf(stderr, "Usage: %s num_port start_port num_clients filename\n",argv[0]);
                exit(1);
    }
    num_port = atoi(argv[1]);
    if (num_port > MAX_PORTS) {
                fprintf(stderr, "Too many ports for eachclient.\n");
                exit(1);
    }
    client_port = atoi(argv[2]);  /* first client port */
    num_client = atoi(argv[3]);   /* number of clients */
```

```
  if (num_client <=0) {
                fprintf(stderr, "Number of clients has to be greater than 0.\n");
                exit(1);
  }
  if (num_client > MAX_CLIENTS) {
                fprintf(stderr, "Too many clients.\n");
                exit(1);
  }

  /* get the filename and read out the client names*/
  filename = argv[4];
  inFile = fopen(filename,"r");
  if (inFile == NULL) {
                fprintf(stderr,"File '%s' does NOT exist.\n", filename);
                exit(1);
  }
  for (i=0; i<num_client; i++) {
    int length;
    length = fscanf(inFile,"%s", client_address[i]);
    if (length<=0) {  /* the end of file */
                        num_client = i;
                        break;
                }
  }

  /* contiune to read more parameters if any*/
  next_arg = 5;
  while (next_arg < argc) {
                if (!STRCMP("-i", argv[next_arg])) {
                        /* set the interval for get_next_choice */
                        timeout.tv_sec = atol(argv[++next_arg]);
    }
                else if (!STRCMP("-l", argv[next_arg])) {
          log_filename = argv[++next_arg];
                        logfd=fopen(log_filename,"a");
          if(logfd == NULL) {
                        fprintf(stderr,"WARNING: cannot open log file %s\n", log_filename);
                        }
    }
                next_arg++;
  }

}

static void show_info()
{
  int sock[MAX_PORTS*MAX_CLIENTS];
  int clnt_id, i, j;
  char outbuf[128];
  int  len, w;
  static last_choice=0;

  for (clnt_id=0; clnt_id<num_client; clnt_id++) {
    for (i=0; i<num_port; i++) {
                int sock_id = clnt_id*num_port + i;
                sock[sock_id]=open_socket(client_port+i, client_address[clnt_id]);
    }
  }

  /* first read and write login as a special terminal */
  read_from_all_clients(0, sock, 0);
  sprintf(outbuf, "%d\t%d\n", 0, 1);
  len = strlen(outbuf);
  write_to_all_clients(0, sock, outbuf, len);

  /* initial data are garbage from thelogin session, so we skip them  */
  read_from_all_clients(0, sock, 0);

  while (1) {

    int         choice;

    choice = get_next_choice();

                if ( choice == QUIT ) {
                        /* exit the tpcc_monitor program */
                write_to_all_clients(0, sock, "q", 1);
                        break;
                }
                else if ( choice == WRONG_INPUT ) {
                        fprintf(stderr,"Wrong input. Try again.\n");
    }
    else if ( choice == REFRESH ) {
          write_to_all_clients(choice, sock, "n", 1);
          read_from_all_clients(choice, sock, 1);
          if(last_choice==0)
                        display_all_clients(client_stat);
          else
                        display_client(last_choice, client_stat);
    }
    else if ( choice == HELP ) {
                        fprintf(stderr,"Help info is not available yet.\n");
    }
    else if ( choice != MODE_CHANGE ) {
          if(choice==0)
                        display_all_clients(client_stat);
          else
                        display_client(choice, client_stat);
                        last_choice = choice;
        }
        /* other inputs will not be responsed here */
    }
  }

  /*
   *    main program
   *
   */
  int main(argc, argv)
    int argc;
    char *argv[];
  {
    int sock;
    int next_arg = 1;
    int num_local_wares;
    int i;

    SRANDOM(getpid());
    next_port_to_try = 13000 + (RANDOM() % 120) * 150;

    get_term_env(argc, argv);
    client_stat_init(client_stat);
  #ifdef WIN32
    input_mode = MANUAL;
  #endif
    show_info();
    exit_program(0);
  }



  /* read_from_all_clients:
   *
   * num_print: 1 means the login data,
   *          0 means the first staus data to be skipped,
   *          2 means the normal data to be displayed
   */
  static int read_from_all_clients(int clnt_id, int *sock, int initialized)
  {
    char buffer[MAX_PORTS][4096];
    int num;
    int i,j, first_clnt, last_clnt, cur_clnt;
    int len[MAX_PORTS];

    for (cur_clnt=0; cur_clnt<num_client; cur_clnt++) {
      for (i=0; i<num_port; i++) {
        int s_id = cur_clnt * num_port + i;
                        len[i] = read_from_client(sock[s_id],buffer[i],sizeof(buffer[i]));
                        if ( initialized ) {
          /* Individual client and port start from 1.
           * Index 0 is reserved for summary
           */
                        if (logfd) {
                                write_to_log(logfd, buffer[i], len[i]);
          }
                        get_client_info(&client_stat[cur_clnt+1][i+1], buffer[i], cur_clnt, i);
                      }
      }
    } /* for cur_clnt */

    if (initialized)
                        clients_status_summary(&client_stat[0]);
    return 1;
  }


  static void write_to_all_clients(int clnt_id, int *out, char *buffer, int len)
  {
    int i;
    int first_clnt, last_clnt, cur_clnt;

    for (cur_clnt=0; cur_clnt<num_client; cur_clnt++) {
      for (i=0; i<num_port; i++) {
        int sock_id = cur_clnt*num_port + i;
  /*                    fprintf(stderr, "write_to_all, cur_clnt is %d, num_port is %d, socket is %d\n",
  cur_clnt, num_port, sock_id); */
                        write_to_client(out[sock_id], buffer, len);
      }
    }
  }

  /*
   * write_to_client
   *
   *  Input: out: the file descriptor to be used to send the output
   *                  Buffer: The buffer to be sent
   *                  len: The length of the above buffer
   */
  static void write_to_client(int out, char *buffer, int len)
  {
    int w, i;

    DPRINT((">> write_to_client. len:%d\n", len));
    w = write(out, buffer, len);
    if (w != len) {
                        fprintf(stderr," write_to_client write returned: %d instead of %d\n",
                                w, len);
                        EXIT(4);
```

```
    }
    DPRINT(("<< write_to_client\n"));
}

/*
 *  read_from_client
 *                  Process data returning from the client.
 *                  At this point we simply send the data to standard out
 *
 *  Input: sock: The file descriptor to be used to read the data
 *                   Buffer: A buffer allocated by the caller to store the data read
 *                   len: The length of the above buffer
 *  OUTPUT: Returns the size it actually read.
 *
 *  The input from the client is a null terminated string.
 */
static int read_from_client(int sock, char *buffer, int buf_len)
{
    int r;
    char *nextP = buffer;
    int len = 0;
    int done = 0;
    char  mark[10]=ENDMSG;

    do {
                int i;
                DPRINT(("before Reading from client, len %d\n", len));
                r = read(sock, nextP, buf_len - len);
                if (r < 1) {
                    if (r == 0) {
                                fprintf(stderr,"Connection closed.  Bye!\n");
                                EXIT(5);
                    }

                    fprintf(stderr,"process_client_data read returned: %d, err_code=%d\n", r,errno);
                    EXIT(6);
                }
                for (i=0; i<r; i++) {
                    /* a mark to make sure it is the end of message */
                    if ( (i+3 <r) && (nextP[i] == mark[0]) && (nextP[i+1]==mark[1])
                                 && (nextP[i+2]==mark[2]) && (nextP[i+3] == mark[3]) ) {
                                nextP[i] = '\0';
                                done = 2;
                                break;
                    }
                    if (nextP[i] == TRIGGERc) {
                                done = 1;
                                break;
                    }
                }
                nextP += r;
                len += r;
    } while ((len < buf_len) && !done);
    if (len < buf_len-1) {
      if (done==1)
        buffer[len] = '\0';
      else
                 buffer[len-strlen(mark)] = '\0';
    }

    DPRINT(("read_from_client returning %d\n", len));
    return(len);
}

static void exit_program(int err)
{
    if ( err )
                fprintf( stderr, "exit_program: Error Code = %d\n", err );
    exit( err );
}


/* skip_endline:
 *
 */

char *skip_endline(char *curP)
{
    int  i;

    for (i=0; i<strlen(curP); i++) {
                if (curP[i]=='\n')
                    return curP+i+1;
    }
    printf("ERROR in skip_endline\n");
}

/* get_client_info:
 * calculate TPM and response time
 * tag: is used to for future expandation
 */

void get_client_info(client_status_t *client_rec,
                                    char *buf, int clnt, int port)
{
    int             i, j;
    int             type;
    long   time_sec, time_usec;
    double time_diff1, time_diff2;
    char   prefix1[50], prefix2[50], prefix3[50];
    char   *bufP = buf;
    total_tran_count_t          tran_ct;
    tran_info_t  *curP;
    int                         total_newo;
    double tpm1, tpm2;
    int    err_diff1, err_diff2;
    static init_state = 1;
    static total_tran_count_t  tran_reported[MAX_CLIENTS][MAX_PORTS][2];
    static struct timeval  oldUserTime[MAX_CLIENTS][MAX_PORTS];
    static struct timeval  oldSysTime[MAX_CLIENTS][MAX_PORTS];
    static struct timeval  newUserTime[MAX_CLIENTS][MAX_PORTS];
    static struct timeval  newSysTime[MAX_CLIENTS][MAX_PORTS];
    int    num_threads, num_threadsInit;
    double cpuUsagePercent;

    /* initialiation */
    if (init_state) {
       for (i=0;i<MAX_CLIENTS; i++)
         for (j=0;j<MAX_PORTS; j++) {
                memset(&tran_reported[i][j][0], 0, sizeof(tran_ct));
                memset(&tran_reported[i][j][1], 0, sizeof(tran_ct));
         }
                init_state = 0;
    }

    /* read information from buf line by line */
    /* this part has to be consistent with client_status_report() */
    memset(&tran_ct, 0, sizeof(tran_ct));
    sscanf(bufP,"%s %s %s",prefix1,prefix2,prefix3);
    bufP = skip_endline(bufP);
    sscanf(bufP, "%d %d %d %d %d %d",&time_sec, &time_usec,
                                                    &total_newo, &tran_ct.errors,
                                                    &num_threads, &num_threadsInit);
    bufP = skip_endline(bufP);
    sscanf(bufP, "%d %d %d %d",
                  &newUserTime[clnt][port].tv_sec, &newUserTime[clnt][port].tv_usec,
                  &newSysTime[clnt][port].tv_sec, &newSysTime[clnt][port].tv_usec);
    bufP = skip_endline(bufP);
    for (i=1, curP=tran_ct.tran+1; i<=MAX_TRAN_TYPE; i++, curP++) {
                float  clnt_id, srv;
                sscanf(bufP,"%d %d %d %f %f",&type, &(curP->num), &(curP->errs),
                                &clnt_id, &srv);
      curP->RT[0] = clnt_id;
                curP->RT[1] = srv;
      fflush(stdout);

      if (i<MAX_TRAN_TYPE)  bufP = skip_endline(bufP);
    }

    /* calculate the TPM */
    /* only the last interval is interested. */
    /* time_diff1, tpm1, and err_diff1 should be removed */
    tran_ct.time = time_sec + (double)time_usec / 1e6;
    tran_ct.tran[NEWO_TRANS].num= total_newo;
    time_diff1 = tran_ct.time - tran_reported[clnt][port][0].time;
    time_diff2 = tran_ct.time - tran_reported[clnt][port][1].time;
    tpm2 = (double)(total_newo - tran_reported[clnt][port][1].tran[NEWO_TRANS].num)/ time_diff2
* 60;

    /* calculate the errors */
    err_diff1 = tran_ct.errors-tran_reported[clnt][port][0].errors;
    err_diff2 = tran_ct.errors-tran_reported[clnt][port][1].errors;

    /* calculate the CPU usage */
    if ((int)tran_reported[clnt][port][1].time==0)
                cpuUsagePercent = 0.0;
    else {
                double user_time_diff, sys_time_diff;
      user_time_diff = (double)newUserTime[clnt][port].tv_usec/1e6
                                        + newUserTime[clnt][port].tv_sec
                                    -(double)oldUserTime[clnt][port].tv_usec / 1e6
                                        - oldUserTime[clnt][port].tv_sec;
      sys_time_diff = (double)newSysTime[clnt][port].tv_usec/1e6
                                        + newSysTime[clnt][port].tv_sec
                                    -(double)oldSysTime[clnt][port].tv_usec / 1e6
                                        - oldSysTime[clnt][port].tv_sec;
      cpuUsagePercent = (user_time_diff + sys_time_diff) / time_diff2 * 100;
    }

    /* keep all the information except RT in client_rec */
    set_client_st(client_rec, tpm2, err_diff2, prefix2,
                                time_diff2, num_threads-num_threadsInit, num_threadsInit,
                                cpuUsagePercent);
    set_client_st_rt(client_rec, &tran_ct,&tran_reported[clnt][port][1]);

    /* keep the old values */
    tran_reported[clnt][port][0]= tran_reported[clnt][port][1];
    tran_reported[clnt][port][1]= tran_ct;
    oldUserTime[clnt][port] = newUserTime[clnt][port];
    oldSysTime[clnt][port] = newSysTime[clnt][port];
}


void clients_status_summary(client_status_t  *client_stat[])
{
    int  i, j, k;

    /* initialization before each summary */
```

```
  /* client_stat[i][0] is the summary of client i */
  /* client_stat[0][0] is the summary of all clients */
  for (i=0; i<=num_client; i++) {
    client_stat[i][0].tpm = 0;
    client_stat[i][0].err_diff = 0;
    client_stat[i][0].num_thrds = 0;
    client_stat[i][0].num_thrdsInit = 0;
    client_stat[i][0].cpu_usage_percent = 0;
    for (k=1; k<=MAX_TRAN_TYPE; k++) {
                client_stat[i][0].num_trans[k] = 0;
                client_stat[i][0].rt_diff[k][0] = 0;
                client_stat[i][0].rt_diff[k][1] = 0;
    }
  }

  for (i=1; i<=num_client; i++) {
    for (j=1; j<=num_port; j++) {
      /* calculate the summary for client i */
              client_stat[i][0].tpm += client_stat[i][j].tpm;
              client_stat[i][0].err_diff += client_stat[i][j].err_diff;
              client_stat[i][0].total_errs += client_stat[i][j].err_diff;
              client_stat[i][0].cpu_usage_percent += client_stat[i][j].cpu_usage_percent;
      if (j==1) {
                strcpy(client_stat[i][0].cur_time,client_stat[i][1].cur_time);
                client_stat[i][0].interval = client_stat[i][j].interval;
      }
              client_stat[i][0].num_thrds += client_stat[i][j].num_thrds;
              client_stat[i][0].num_thrdsInit += client_stat[i][j].num_thrdsInit;

      /* response time */
      for (k=1; k<=MAX_TRAN_TYPE; k++) {
                client_stat[i][0].num_trans[k]+=client_stat[i][j].num_trans[k];
                client_stat[i][0].rt_diff[k][0]+=client_stat[i][j].rt_diff[k][0];
                client_stat[i][0].rt_diff[k][1]+=client_stat[i][j].rt_diff[k][1];
      }
    }

    /* caclulate the summary for all clients */
    client_stat[0][0].tpm += client_stat[i][0].tpm;
    client_stat[0][0].err_diff += client_stat[i][0].err_diff;
    client_stat[0][0].total_errs += client_stat[i][0].err_diff;
    if (i==1) {
                strcpy(client_stat[0][0].cur_time,client_stat[i][0].cur_time);
                client_stat[0][0].interval = client_stat[i][0].interval;
    }
    client_stat[0][0].num_thrds += client_stat[i][0].num_thrds;
    client_stat[0][0].num_thrdsInit += client_stat[i][0].num_thrdsInit;
    /* response time */
    for (k=1; k<=MAX_TRAN_TYPE; k++) {
              client_stat[0][0].num_trans[k]+=client_stat[i][0].num_trans[k];
              client_stat[0][0].rt_diff[k][0]+=client_stat[i][0].rt_diff[k][0];
      client_stat[0][0].rt_diff[k][1]+=client_stat[i][0].rt_diff[k][1];
    }
  }
}


void display_client(int clnt, client_status_t *client_stat[])
{
  int i, j;
  char prefix[20];

  print_header(client_stat[clnt][0].cur_time, rt_mode, clnt,
          client_stat[clnt][0].interval);
  /* print out information for each client*/
  for (i=0; i<num_port; i++) {
    sprintf(prefix, "(port-%1d)", client_port+i);
              print_info(prefix,client_stat[clnt][i+1]);
    one_print_rt_avg(client_stat[clnt][i+1], rt_mode, 0);
  }
  /* print out information for all the clients*/
  printf ("-------------------------------------------------------------------------------\n");
  print_info("Total",client_stat[clnt][0]);
  one_print_rt_avg(client_stat[clnt][0], rt_mode, 1);
  fflush(stdout);
}

void display_all_clients(client_status_t *client_stat[])
{
  int i, j;

  print_header(client_stat[0][0].cur_time, rt_mode, 0,
          client_stat[0][0].interval);
  /* print out information for each client*/
  for (i=0; i<num_client; i++) {
              print_info(client_address[i],client_stat[i+1][0]);
    one_print_rt_avg(client_stat[i+1][0], rt_mode, 0);
  }
  /* print out information for all the clients*/
  printf ("-------------------------------------------------------------------------------\n");
  print_info("Total",client_stat[0][0]);
  one_print_rt_avg(client_stat[0][0], rt_mode, 1);
  fflush(stdout);
  fflush(stdout);
}


static void one_print_rt_avg(client_status_t client_st,
                             int rt_mode, int both)
{
  int i;
  static char  *srvPrefix = "Server RT                    ";
  static char  *clnPrefix = "Client RT                    ";
  for (i=1; i<=MAX_TRAN_TYPE; i++) {
              if ( client_st.num_trans[i]>0 )
      printf(" %4.3f", (double)(client_st.rt_diff[i][rt_mode])/(client_st.num_trans[i]*1000));
    else
              printf("   -- ");
  }
  if (client_st.cpu_usage_percent>0)
    printf("%5.1f", client_st.cpu_usage_percent);
  else printf("   --");
  printf("\n");
  if (both) {
      rt_mode = 1 - rt_mode;  /* print SRV or CLNT RT based on rt_mode */
      printf("%s", rt_mode ? srvPrefix : clnPrefix);
      for (i=1;i<=MAX_TRAN_TYPE; i++) {
              if ( client_st.num_trans[i]>0 )
        printf(" %4.3f", (double)(client_st.rt_diff[i][rt_mode])/(client_st.num_trans[i]*1000));
        else
              printf("   -- ");
      }
      printf("\n");
  }
}

void set_client_st(client_status_t  *client_rec, double tpm, int errs,
                   char *time, double interval, int num_thr, int num_thrInit,
                   double cpuUsagePercent)
{
  client_rec->tpm = tpm;
  client_rec->err_diff = errs;
  client_rec->total_errs += errs;
  strcpy(client_rec->cur_time,time);
  client_rec->interval = interval;
  client_rec->num_thrds = num_thr;
  client_rec->num_thrdsInit = num_thrInit;
  client_rec->cpu_usage_percent = cpuUsagePercent;
}


void set_client_st_rt(client_status_t  *client_rec,
          total_tran_count_t *curP, total_tran_count_t *prevP)
{
  int i,j;

  for (i=1; i<=MAX_TRAN_TYPE; i++) {
    for (j=0; j<2; j++) {/* for client RT and server RT */
      client_rec->num_trans[i]=curP->tran[i].num - prevP->tran[i].num;
      client_rec->rt_diff[i][j]=curP->tran[i].RT[j]- prevP->tran[i].RT[j];
    }
  }
}

int get_next_choice()
{

  static  int choice=0;
  static  int  new_start=1;
  static  int  interval=-1;
  static  struct  timeval time1, time2;
  char cmdbuf[256];
  fd_set  readfds;
  int     j;
  int     isNT=0;
  struct  timezone tz;


#ifdef WIN32
  isNT = 1;
#endif

  if (interval == -1)  interval=timeout.tv_sec;  /* for initialization*/
  /* print out interactive input choices*/
  if (input_mode==AUTOMATIC){
    fprintf(stderr, "\n(AUTO-%d) [Interval Manual ",interval);
  }
  else if (!isNT) { /* MANUAL_MODE */
    fprintf(stderr, "\n(MANUAL) [Auto ");
  }
  fprintf(stderr, "Refresh Client-rt Server-rt Quit 0...%1d] > ", num_client);
  FD_ZERO(&readfds);
  FD_SET(0, &readfds);

  if ( input_mode==AUTOMATIC ) {
    if (new_start) {
              gettimeofday(&time1, &tz);
              timeout.tv_sec = interval;  /* resume the original value for each new start*/
              new_start = 0;
    }
    else {
      float time_diff;
              gettimeofday(&time2, &tz);
      time_diff = time_diff_ms(&time2, &time1)/1000;  /* in seconds */
      if (time_diff < timeout.tv_sec) {
                  timeout.tv_sec -= (int)time_diff;
      } else {
                  timeout.tv_sec = 0;
```

```
        }
        time1 = time2;
    }
    if (select(1,&readfds,NULL,NULL,&timeout) > 0) {
      int cc;
      cc = read(0, cmdbuf, sizeof(cmdbuf));
      if (cc <= 0)
            fprintf(stderr, "\n Reading error from stdin\n");
      else {
                  if ( cmdbuf[0]=='m' ) {
                              input_mode = MANUAL;
                              return(MODE_CHANGE);
                  }
          else if ( cmdbuf[0] == 'i' || cmdbuf[0] == 'I' ) {
                              fprintf(stderr, "New interval in seconds: ");
                              scanf("%d", &timeout.tv_sec);
                              if (timeout.tv_sec < 5) {
                                timeout.tv_sec = 5;
                              }
                              fprintf(stderr, "interval change successful.\n");
                              new_start = 1;
                              interval = timeout.tv_sec;
                              return(MODE_CHANGE);
          }
                  else if ( cmdbuf[0] == '\n' ) { /* for a single return key */
                              new_start = 1;
                              return(REFRESH);
          }
      }
    } else { /* no input so far,keep the last input_mode and display_mode */
          new_start = 1;
                  return(REFRESH);
    }
  }
  else { /* MANUAL mode */
    scanf("%s", cmdbuf);
    if ((cmdbuf[0] == 'a') && (!isNT) ) {
                  input_mode = AUTOMATIC;
      new_start = 1;
                  return(MODE_CHANGE);
    }
    else if (cmdbuf[0] == '\n') {
                  return(choice);
    }
  }

  /* take care of the common input*/
  if ( cmdbuf[0]<='9' && cmdbuf[0]>='0' ) {
        choice = atoi(cmdbuf);
                  if (choice==0) {
                              display_mode = ALL;
                              return(choice);
        }
                  else if (choice <= num_client) {
                      display_mode = INDIVIDUAL;
                       return choice;
        }
                  else {
                              choice = 0;
                              return WRONG_INPUT;
        }
  }
  else if (cmdbuf[0] == 'q' || cmdbuf[0] == 'Q') {
                  return(QUIT);
  }
  else if (cmdbuf[0] == 'c' || cmdbuf[0] == 'C') {
                  rt_mode = 0;
                  return(choice);
  }
  else if (cmdbuf[0] == 's' || cmdbuf[0] == 'S') {
                  rt_mode = 1;
                  return(choice);
  }
  else if (cmdbuf[0] == 'r' || cmdbuf[0] == 'R') {
                  new_start = 1;
                  return(REFRESH);
  }
  else if (cmdbuf[0] == 'h' || cmdbuf[0] == 'H') {
                  return(HELP);
  }
  else   return(WRONG_INPUT);
}


/* type=1 means server RT; type=0 means client RT */
void print_header(char *prefix, int rt_mode, int clnt, double time_diff)
{
   int i;
   /* TPM ErrPm Act Init NewO Pay SL OS DVRY */
   printf("\n\n");
/*    printf("%s ", input_mode==AUTOMATIC ? "[AUTO]" : "[MANUAL]"); */

   printf("%s  %s - %s RT, Interval %.0f sec \n", prefix,
                                        clnt==0 ? "All clients" : client_address[clnt-1],
                                    rt_mode ? "Server" : "Client", time_diff);
   printf ("%s  TPM ErrPm Errs  Init   Act    NO    PA    OS    DV    SL
%s\n",LEADING_SPACE,"cpu%");
   printf ("-------------------------------------------------------------------------------\n");
}


void print_info(char *prefix, client_status_t client_st)
{
      int i;

                  printf("%s", prefix);
                  for (i=0; i<strlen(LEADING_SPACE)-strlen(prefix); i++)
                              printf(" ");
      printf("%5.0f ",client_st.tpm);
      /* print out err if any */
      if (client_st.err_diff != 0)
                  printf("%5.1f ", (double)client_st.err_diff / client_st.interval * 60);
      else
                  printf("   -- ");
      if (client_st.total_errs !=0 )
                  printf("%4d ", client_st.total_errs);
      else
                  printf("  -- ");
      printf("%5d %5d ", client_st.num_thrdsInit, client_st.num_thrds);
}

void print_errs(client_status_t **client_stat)
{
   int i, j;
   char *prefix="Total Errors     ";

   if (client_stat[0][0].total_errs>0) {
                  printf("Total Errors     %4d\n", client_stat[0][0].total_errs);

      for (i=0;i<num_client; i++) {
                  if (client_stat[i+1][0].total_errs>0)
                              printf("%s", client_address[i]);
        for (j=0;j<strlen(prefix)-strlen(client_address[i]); j++)
                              printf(" ");
        printf("%4d\n", client_stat[i+1][0].total_errs);
      }
   }
}


void client_stat_init(client_status_t *client_stat[])
{
   int  i,j;

   /* allocate space first */
   for (i=0; i<=num_client; i++) {
      client_stat[i] = (client_status_t *)malloc((num_port+1)*sizeof(client_status_t));
                  if (client_stat[i]==NULL) {
        fprintf(stderr,"client_stat_init: malloc failed\n");
        exit(1);
      }
      for (j=0; j<=num_port; j++) {
                  client_stat[i][j].total_errs = 0;
                  client_stat[i][j].cpu_usage_percent = 0;
                }
   }
}

write_to_log(FILE *fd, char *buf, int len)
{
  fprintf(fd, "%s", buf);
  fprintf(fd, "%s\n", ENDMSG);  /* special symbol means the end of a buffer */
}
```

Tpcc_monitor.h

```
/*
 *                tpcc_monitor.h
 *
 * $Revision: 1.1 $
 * $Date: 1998/07/02 18:28:53 $
 * $Log:             $
 *
 *
 * $TALog: tpcc_monitor.h,v $
 * Revision 1.1  1998/07/02  18:28:53wenjian
 * Define client_status_t to keep the information (e.g, tpm and response
 * time) per tpcc_client.
 * [added by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients,r1.5]
 *
 *
 */

/*
 *   tpcc_monitor.h
 */


#ifndef TPCC_MONITOR_H
#define TPCC_MONITOR_H

typedef struct {
  double   tpm;
  int      err_diff;
```

```
 int     total_errs;
 char    cur_time[30];
 double  interval;
 int     num_thrds;
 int     num_thrdsInit;
 double  cpu_usage_percent;
 int     rt_diff[MAX_TRAN_TYPE+1][2]; /* 0 for client and 1 for server*/
 int     num_trans[MAX_TRAN_TYPE+1];
} client_status_t;

#endif /* TPCC_MONITOR_H */
```

Tpcc_type.h

```
/* Generated by IDL compiler version OSF DCE T1.2.0-09 */
#ifndef tpcc_types_v1_0_included
#define tpcc_types_v1_0_included
#ifndef IDLBASE_H
#include <dce/idlbase.h>
#endif

#ifdef __cplusplus
    extern "C" {
#endif

#ifndef nbase_v0_0_included
#include <dce/nbase.h>
#endif
#define NAME_LENGTH (32)
#define NEWO_INTERFACE (1)
#define PAYMENT_INTERFACE (2)
#define ORDER_STAT_INTERFACE (4)
#define DELIVERY_INTERFACE (8)
#define STOCK_INTERFACE (16)
#define ONLINE_INTERFACES (23)
#define ALL_INTERFACE (65535)
#define NEWO_TRANS (1)
#define PAYMENT_TRANS (2)
#define ORDER_STAT_TRANS (3)
#define DELIVERY_TRANS (4)
#define STOCK_TRANS (5)
#define MAX_TRAN_TYPE (5)
typedef struct  {
idl_long_int sec;
idl_long_int usec;
} time_type;
typedef struct  {
idl_short_int dtype;
idl_short_int returncode;
idl_long_int sql_code;
idl_long_int isam_code;
idl_long_int num_rms;
time_type start_time;
time_type end_time;
} data_header;
typedef struct  {
data_header header;
idl_short_int w_id;
idl_short_int d_id;
idl_short_int c_id;
idl_short_int c_w_id;
idl_short_int c_d_id;
idl_short_int byname;
idl_long_float h_amount;
idl_char pay_date[20];
idl_char w_name[11];
idl_char w_street_1[21];
idl_char w_street_2[21];
idl_char w_city[21];
idl_char w_state[3];
idl_char w_zip[10];
idl_char d_name[11];
idl_char d_street_1[21];
idl_char d_street_2[21];
idl_char d_city[21];
idl_char d_state[3];
idl_char d_zip[10];
idl_char c_first[17];
idl_char c_middle[3];
idl_char c_last[17];
idl_char c_phone[17];
idl_char c_credit[3];
idl_char c_street_1[21];
idl_char c_street_2[21];
idl_char c_city[21];
idl_char c_state[3];
idl_char c_zip[10];
idl_long_float c_credit_lim;
idl_long_float c_balance;
idl_long_float c_discount;
idl_long_float c_ytd_payment;
idl_short_int c_payment_cnt;
idl_char c_date[20];
idl_char c_data[201];
} payment_data_t;
typedef struct  {
idl_short_int ol_supply_w_id;
idl_short_int ol_quantity;
idl_short_int s_quantity;
idl_long_int ol_i_id;
idl_char name_i[25];
idl_char brand_generic[2];
idl_long_float price;
idl_long_float ol_amount;
idl_long_int s_idx;
idl_char s_dist[25];
} OL_TABLE;
typedef OL_TABLE newOrder_item_t;
typedef struct  {
data_header header;
idl_short_int w_id;
idl_short_int d_id;
idl_short_int c_id;
idl_short_int o_ol_cnt;
idl_short_int o_all_local;
idl_short_int items_valid;
idl_short_int total_items;
idl_long_int o_id;
idl_long_float w_tax;
idl_long_float d_tax;
idl_long_float total;
idl_long_float c_discount;
idl_char entry_date[20];
idl_char c_last[17];
idl_char c_credit[3];
idl_char statusline[26];
OL_TABLE item[15];
} newOrder_data_t;
typedef struct  {
idl_long_int ol_i_id;
idl_short_int ol_supply_w_id;
idl_short_int ol_quantity;
idl_long_float ol_amount;
idl_char delivery_date[20];
} orderStatusItem_t;
typedef struct  {
data_header header;
idl_short_int w_id;
idl_short_int d_id;
idl_short_int c_id;
idl_short_int o_id;
idl_short_int o_ol_cnt;
idl_short_int byname;
idl_short_int o_carrier_id;
idl_char c_last[17];
idl_char c_first[17];
idl_char c_middle[3];
idl_char entry_date[20];
idl_long_float c_balance;
orderStatusItem_t item[15];
} orderStatus_data_t;
typedef struct  {
data_header header;
idl_short_int w_id;
idl_short_int d_id;
idl_short_int threshold;
idl_long_int stock_count;
} stockLevel_data_t;
typedef struct  {
data_header header;
idl_short_int w_id;
idl_short_int o_carrier_id;
idl_long_int queued_time;
idl_short_int status;
idl_char exec_status[50];
idl_long_float start_queue;
} delivery_data_t;
typedef struct  {
idl_long_int first_wh;
idl_long_int last_wh;
idl_long_int server_id;
} dbInfo_data_t;
typedef struct  {
idl_long_int tran_type;
union  {
/* case(s): 1 */
newOrder_data_t new_order;
/* case(s): 2 */
payment_data_t payment;
/* case(s): 3 */
orderStatus_data_t order_status;
/* case(s): 4 */
delivery_data_t delivery;
/* case(s): 5 */
stockLevel_data_t stock_level;
} data;
} tpcc_data_t;

#ifdef __cplusplus
    }

#endif
#endif
```

Tran_stat.c

```
/*
 *                  tran_stat.c
 *
 * $Revision: 1.4 $
 * $Date: 1998/01/23 15:09:12 $
 * $Log:              $
 *
 *
 * $TALog: tran_stat.c,v $
 * Revision 1.4  1998/01/23  15:09:12  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 * Revision 1.1  1997/04/20  11:58:07  oz
 * - This is the code base modified at IBM Poughkeepsie
 *   by Ofer Zajicek and Radha Sivaramakrishnan for the
 *   SP scaling test for TPCC.
 * [added by delta oz-19782-TPCC-add-ibm-sp-code, r1.1]
 *
 * Revision 1.2  1995/07/31  16:53:23  oz
 * - Modified the RT90 constants to be in seconds (instead of ticks)
 * [from r1.1 by delta oz-16044-TPCC-add-intermediate-results-to-client, r1.6]
 *
 * Revision 1.1  1995/07/26  18:00:54  oz
 * - Moved all the transaction statistics into its own module
 * [added by delta oz-16044-TPCC-add-intermediate-results-to-client, r1.1]
 *
 *
 *
 */


#include "tran_stat.h"
#include "do_tpcc.h"
#include "tpcc_type.h"
```

```
/*
 *  tran_stat_init -- Initialize a transaction statistic block
 *
 *  Some of the entry in the statistic block are transaction
 *  dependent such as the response time requiredment (RT90).
 *  The structure holds the transaction type and its name for
 *  good measure.
 */
void tran_stat_init(statP, type)
 tran_stat_t *statP;
 int type;
{
   int i;
   pthread_mutex_init(&statP->lock, pthread_mutexattr_default);
   statP->trans = statP->resptime = 0;

   for (i=0; i<5; i++)
                statP->resp_num[i] = 0;

   statP->err = 0;
   statP->roll_backs = 0;
   statP->wait_time = 0;
   statP->type = type;
   statP->client_resp_time = 0;
   statP->client_resp_num = 0;
   statP->num_last_printed = 0;
   statP->num_err_last_printed = 0;
   switch(type) {
    case NEWO_TRANS:
                statP->name = "NEWORD";
                statP->RT90 = NEWORD_90RT * 1000;
                break;
    case PAYMENT_TRANS:
                statP->name = "PAYMENT";
                statP->RT90 = PAYMENT_90RT * 1000;
                break;
    case ORDER_STAT_TRANS:
                statP->name = "ORDSTAT";
                statP->RT90 = ORDSTAT_90RT * 1000;
                break;
    case DELIVERY_TRANS:
                statP->name = "DELIVERY";
                statP->RT90 = DELIVERY_90RT * 1000;
                break;
    case STOCK_TRANS:
                statP->name = "STOCKLEV";
                statP->RT90 = STOCKLEV_90RT * 1000;
                break;
    default:
                err_printf("Invalid tran type %d at tran_stat_init.\n", type);
                exit(1);
                break;
   }
}

static void report_tpm(tran_stat_t *statP)
{
   struct timeval cur_time;
   struct timezone tzp;
   gettimeofday(&cur_time, &tzp);
   if (statP->num_last_printed == 0) {
      statP->num_last_printed = statP->trans;
      statP->last_time = cur_time;
   } else {
      int tran_diff = statP->trans - statP->num_last_printed;
                int err_diff = statP->err - statP->num_err_last_printed;
      int time_diff = cur_time.tv_sec - statP->last_time.tv_sec;
      if (time_diff > 30) {
         err_printf("%d trans in %d sec, \t---->  %7.1f tpmC %7.1f errPm\n",
            tran_diff, time_diff,
            ((double)tran_diff / time_diff) * 60,
            ((double)err_diff / time_diff) * 60);
         statP->num_last_printed = statP->trans;
                   statP->num_err_last_printed = statP->err;
         statP->last_time = cur_time;
      }
   }
}
/*
 *  tran_stat_add
 *                Update the statistics block of a transaction with a new
 *                data point.
 *                The difference between end_time and start_time is the response
 *                time for this transaction.  status is the TPCC Status code.
 */
void tran_stat_add(statP, tran_timesP, wait_time, status)
 tran_stat_t *statP;
 tran_times_t *tran_timesP;
 int wait_time;
 int status;
{
   int resp_time;                               /*  In miliseconds  */
   int client_resp_time;        /*  In miliseconds */

   pthread_mutex_lock(&statP->lock);

   resp_time = time_diff_ms(&tran_timesP->end_time, &tran_timesP->start_time);
   client_resp_time = time_diff_ms(&tran_timesP->receive_time,
                                                &tran_timesP->send_time);
   if ((status == TPCC_SUCCESS) ||
                (status == INVALID_NEWO) && (statP->type == NEWO_TRANS)) {

                if ((status == INVALID_NEWO) && (statP->type == NEWO_TRANS))
                   statP->roll_backs++;

                if (resp_time <= statP->RT90) {
                   if (resp_time <= statP->RT90 / 4)
                                statP->resp_num[0]++;
                   else if (resp_time <= statP->RT90 / 2)
                                statP->resp_num[1]++;
                   else
                                statP->resp_num[2]++;
                } else if (resp_time <= statP->RT90 * 2) {
                   statP->resp_num[3]++;
                } else if (resp_time <= statP->RT90 * 4) {
                   statP->resp_num[4]++;
                }
                statP->trans++;
                statP->resptime += resp_time;
                statP->wait_time += wait_time;
                statP->client_resp_time += client_resp_time;
                if (client_resp_time < statP->RT90)
                   statP->client_resp_num++;
   } else {
                statP->err++;
   }
   if (((statP->trans + statP->err) % 10 == 1) &&
      (strcmp(statP->name, "NEWORD") == 0)) {
                report_tpm(statP);
   }

   pthread_mutex_unlock(&statP->lock);
}

/*
 *  tran_stat_print
 *
 *                Print the statistics gathered for one transaction.
 *
 *                The new print format (enabled by thesplit_resp
 *                global variable) prints the response time in 5 catagories.
 *                When the variable is not set the old format is used.
 */
void tran_stat_print(out, statP, run_time, split_resp)
 FILE *out;
 tran_stat_t *statP;
 int run_time;                  /*  in seconds  */
 int split_resp;
{
   int i;
   double    ave, tpm, pct90;

   pthread_mutex_lock(&statP->lock);

   if (statP->trans == 0) {
                ave=0.0;
                pct90=0.0;
   } else {
                ave=(((double)statP->resptime) / 1000.0) / ((double)statP->trans);
   }
   fprintf(out,"%-12s  %5d  %6.2f",
                   statP->name, statP->trans, ave);
   if (split_resp) {
                double average_wait;
                double client_resp_avg;         /*  Average response time as seen
                                                        *  by the client
                                                        */
                double client_good_ratio;  /*  The ratio of transactions that had
                                                        *  good client response time
                                                        */
                if (statP->trans) {
                   average_wait = (double)statP->wait_time / ((double)statP->trans);
                   client_good_ratio =
                                (double)statP->client_resp_num / (double)statP->trans;
                   client_resp_avg =
                                (double)statP->client_resp_time / 1000. / (double)statP->trans;
                } else {
                   average_wait = 0.0;
                   client_good_ratio = 0.0;
                   client_resp_avg = 0.0;
                }
                fprintf(out, "  %6.2f %6.2f  %4.2f",
                                average_wait, client_resp_avg, client_good_ratio);
                for (i=0; i<5; i++)
                   fprintf(out, " %5d", statP->resp_num[i]);
                fprintf(out, "\n");
   } else {
                int resp90 = statP->resp_num[0] + statP->resp_num[1] + statP->resp_num[2];
                tpm = ((double)statP->trans / ((double)run_time/60));

                pct90=statP->trans ? ((double)resp90) / ((double)statP->trans) : 0.00;
                fprintf(out, "  %6.2f %.2f\n", tpm, pct90);
   }
   pthread_mutex_unlock(&statP->lock);
}
```

Tran_stat.h

```
/*
*                tran_stat.h
*
* $Revision: 1.4 $
* $Date: 1998/01/23 15:09:15 $
* $Log:             $
*
*
* $TALog: tran_stat.h,v $
* Revision 1.4  1998/01/23  15:09:15  oz
* - Updated the SP TPCC directory to the latest files used
*   during the SP tpcc audit.
* [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27,r1.1]
*
* Revision 1.1  1997/04/20  11:58:08  oz
* - This is the code base modified at IBM Poughkeepsie
*   by Ofer Zajicek and Radha Sivaramakrishnan for the
*   SP scaling test for TPCC.
* [added by delta oz-19782-TPCC-add-ibm-sp-code, r1.1]
*
* Revision 1.1  1995/07/26  18:00:57  oz
* - Moved all the transaction statistics into its own module
* [added by delta oz-16044-TPCC-add-intermediate-results-to-client,r1.1]
*
*
*
*/


/*
*  tran_stat
*
*                A module for keeping track of transaction staticstics.
*/

#ifndef TPCC_TRAN_STAT_H
#define TPCC_TRAN_STAT_H

#include <pthread.h>
#include <sys/time.h>
#include <time.h>
#include <stdio.h>

/*
*  tran_times_t
*                Time stamps along the transaction path.
*                The response time is the difference between the end_time
*                and the start_time;
*/
typedef struct {
   struct timeval start_time;   /* Time transaction was generated        */
   struct timeval entry_time;   /* Time it was received by the client    */
   struct timeval send_time;    /* Time it was sent to the server        */
   struct timeval receive_time; /* Time it was received from the server  */
   struct timeval end_time;     /* Time it was received by the terminal  */
}  tran_times_t;


/*
*  tran_stat_t
*                A structure to keep track of the transactions
*  Entries:
*     - trans: The total number of transactions of this type that completed
*     - resptime: The cumulative response time.  The average
*                                response time is determined by dividing this number
*                                by the number of trans.
*     - resp_num: The number of transactions that completed under a fraction
*                                of the response time specified by the benchmark.
*                                resp_num[0]: 1/4 of RT,
*                                resp_num[1]: 1/2 of RT
*                                resp_num[2]: RT
*                                resp_num[3]  2 * RT
*                                resp_num[4]  4 * RT
*     - err: The number of transactions that failed.
*     - roll_backs: The number of transactions that rolled back.
*                                This is significant only for the new order transaction.
*     - wait_time:  The cumulative wait times
*     - client_resp_time: The cumulative response times at the client:
*                                This is the difference between the time the transaction
*                                was sent to the server and the time the reply was
*                                received by the client.
*                                This time is the response time as seen by the client.
*                                This is equivalent to the response time we were
*                                seeing before when we ran with no waits and no
*                                delays and with a smal number of terminals.
*     - client_resp_num: The number of transactions that met the response
*                                time requirement as seen by the client (in other
*                                words, the number of transactions for which
*                                client_wait_time is less than the required response time.
*     - RT90: The responsed time required by the spec.
*     - type: The transaction type (integer)
*     - name: The transaction name.
*
*/
typedef struct {
   long trans;
   long resptime;
   long resp_num[5];
   long err;
   long roll_backs;
   long wait_time;
   long client_resp_time;
   long client_resp_num;
   int RT90;
   int type;
   char *name;
   pthread_mutex_t lock;
   int num_last_printed;    /*  Number of transaction that was reported
                                                        *  during the last report  */
   int num_err_last_printed;/*  Number of errors that were reported
                                                        *  during the last report  */
   struct timeval last_time; /*  Time of last report   */
} tran_stat_t;


/*  Transaction types are from 1 through 5  */

#define TOTAL_ERRORS(statP) \
  (statP[1].err + statP[2].err + statP[3].err + statP[4].err + statP[5].err)

#define TOTAL_TRANS(statP)\
  (statP[1].trans + statP[2].trans + statP[3].trans + statP[4].trans + statP[5].trans)

void tran_stat_add(tran_stat_t  *statP,
                                        tran_times_t *tran_timesP,
                                        int wait_time,
                                        int status);
void tran_stat_init(tran_stat_t *statP, int type);
void tran_stat_print(FILE *out,
                                         tran_stat_t *statP,
                                         int run_time,
                                         int split_resp);


#endif  /**  TPCC_TRAN_STAT_H **/
```

tpccpl.c

```
 #ifdef RCSID
static char *RCSid =
  "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/tpccpl.c,v 1.3 1998/01/24 14:17:05 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*  For now: Preallocate all the connections that we will need  */
#if 1
#define PREALLOC_CN
#endif

/*+=================================================================+
|       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
|                OPEN SYSTEMS PERFORMANCE GROUP                   |
|                    All Rights Reserved                          |
 +=================================================================+
 | FILENAME
 |    tpccpl.c
 | DESCRIPTION
 |    TPC-C transactions in PL/SQL.
 +=================================================================*/

#ifdef ENCINA
#include <dce/pthread.h>
#define TRACE_WITHOUT_TPP 1
#include <utils/trace.h>
#endif
#include <stdio.h>
#include <time.h>
#include "tpcc.h"
#include "tpcc_info.h"
#include "tpccpl.h"
#include "plora.h"

#define SQLTXT "alter session set isolation_level = serializable"
#ifdef SQL_TRACE
#define SQLTXT1 "alter session set sql_trace = true"
#endif

void print_rt_avg(total_tran_count_t *curP, total_tran_count_t *prevP);
static void check_threads(total_tran_count_t *tran_ctP);

#define PRINT_AV(total, num, str)                   \
{                                                   \
```

```
   if ((num) > 0) {                              \
      fprintf(stderr, " %s %.3f,",str, (double)(total)/(num));     \
   }                                             \
}

/*  ORA_RECONNECT_THRESHHOLD:
 *              Try reconnecting after this many consecutive errors.
 *  MIN_TIME_BETWEEN_RECONNECTS
 *              Min time in seconds to elapse between the last connection time before
 *              we are allowed to try and reconnect again.
 */
#define ORA_RECONNECT_THRESHHOLD        20
#define MIN_TIME_BETWEEN_RECONNECTS     20
#ifdef ENCINA
/* thread slot data that contains OCI handles and thread specific vars */
pthread_key_t thread_key;
int key_init = 0;
pthread_mutex_t key_lock;
pthread_mutex_t init_lock;
pthread_mutex_t dvry_log_lock;
#else
ora_cn_data_t *connectionP = NULL;
#endif
static char delivery_file_name[80];

FILE *lfp;
FILE *fopen ();
double gettime ();
int proc_no = 0;
char *TPC_uid, *TPC_pwd;

static void init_global_data(void);
#ifdef ENCINA
static ora_cn_data_t *get_cn(int tran, void *dataP);
static void done_with_cn(ora_cn_data_t *cnP, int tran, void *dataP);
#ifdef PREALLOC_CN
ora_cn_data_t *cn_array = NULL;
int num_connections = 0;
int cn_id = 0;
#endif
#else
#define get_cn(a,b) connectionP
#define done_with_cn(c,a,b) connectionP
#endif
static void init_cn_data(ora_cn_data_t *dataP);
static void clean_cn(void *ptr);

errrpt (lda, cur)

ldadef *lda;
csrdef *cur;

{

  text msg[2048];

  if (cur->rc) {
     oerhms (lda, cur->rc, msg, 2048);
     fprintf (stderr, "Error in TPC-C server %d: %s\n", proc_no, msg);
  }
  if ((cur->rc == DEADLOCK) || (cur->rc == SNAPSHOT_TOO_OLD))
     return (RECOVERR);
  else
     return (IRRECERR);

}


/* vmm313 void ocierror(fname, lineno, errhp, status)  */
int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
  text errbuf[512];
  ub4 buflen;
  sb4 errcode;

  switch (status) {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_SUCCESS_WITH_INFO\n");
  break;
  case OCI_NEED_DATA:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_NEED_DATA\n");
  break;
  case OCI_NO_DATA:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_NO_DATA\n");
    return IRRECERR;
  break;
  case OCI_ERROR:
    (void) OCIErrorGet (errhp, (ub4) 1,
                        (text *) NULL, &errcode, errbuf,
                        (ub4) sizeof(errbuf), OCI_HTYPE_ERR);
    if (errcode != 8177)
    {
      (void) err_printf("Module %s Line %d\n", fname, lineno);
      (void) err_printf("Warning - %s\n", errbuf);
    }
    if ( errcode == SNAPSHOT_TOO_OLD ) {
      return RECOVERR;
    }
    if ( errcode == DEADLOCK ) {
      return RECOVERR;
    }
    return errcode;
/* vmm313    TPCexit(1); */
/* vmm313    exit(1); */
  break;
  case OCI_INVALID_HANDLE:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_INVALID_HANDLE\n");
    TPCexit(1);
    exit(-1);
  break;
  case OCI_STILL_EXECUTING:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_STILL_EXECUTE\n");
  break;
  case OCI_CONTINUE:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_CONTINUE\n");
  break;
  default:
    break;
  }
  return RECOVERR;
}


FILE  *vopen(fnam,mode)
char  *fnam;
char  *mode;
{
FILE *fd;

#ifdef  DEBUG
  fprintf(stderr, "tkvuopen() fnam: %s,   mode:%s\n", fnam, mode);
#endif

    fd = fopen((char *)fnam,(char *)mode);
    if (!fd){
       fprintf(stderr," fopen on %s failed %d\n",fnam,fd);
       exit(-1);
    }
    return(fd);
}

int sqlfile(fnam,linebuf)
char   *fnam;
text   *linebuf;
{
FILE *fd;
int nulpt = 0;

#ifdef  DEBUG
  fprintf(stderr, "sqlfile() fnam: %s,  linebuf: %#x\n", fnam, linebuf);
#endif

    fd = vopen(fnam,"r");
    while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
    {
       nulpt = strlen((char *)linebuf);
    }
    return(nulpt);
}

/*
void
vgetdate(unsigned char *buf) {
  time_t tloc;
  char temp[5];
  int cen;

  if(time(&tloc) == (time_t)-1) {
    err_printf("Error getting date\n");
    exit(1);
  }
  cftime(temp,"%d",&tloc);
  buf[3] = (unsigned char)atoi(temp);
  cftime(temp,"%m",&tloc);
  buf[2] = (unsigned char)atoi(temp);
  cftime(temp,"%Y",&tloc);
  cen = atoi(temp);
  buf[0] = (unsigned char)((cen/100)+100);
```

```
 buf[1] = (unsigned char)((cen%100)+100);
 cftime(temp,"%H",&tloc);
 buf[4] = (unsigned char)(atoi(temp) + 1);
 cftime(temp,"%M",&tloc);
 buf[5] = (unsigned char)(atoi(temp) + 1);
 cftime(temp,"%S",&tloc);
 buf[6] = (unsigned char)(atoi(temp) + 1);
}
*/
void vgetdate (unsigned char *oradt)
{
 struct tm timebuf;
 struct tm *loctime = &timebuf;
 time_t  int_time;

 struct ORADATE {
 unsigned char          century;
 unsigned char          year;
 unsigned char          month;
 unsigned char          day;
 unsigned char          hour;
 unsigned char          minute;
 unsigned char          second;
 } Date;
int century;
int cnvrtOK;

/* assume convert is successful */
cnvrtOK = 1;

/* get the current date and time as an integer */
time( &int_time);

/* Convert the current date and time into local time */
localtime_r(&int_time, &timebuf);

century = (1900+loctime->tm_year) / 100;

Date.century = (unsigned char)(century + 100);
if (Date.century < 119 || Date.century > 120) cnvrtOK = 0;
Date.year  = (unsigned char)(loctime->tm_year+100);
if (Date.year < 100 || Date.year > 199) cnvrtOK = 0;
Date.month = (unsigned char)(loctime->tm_mon + 1);
if (Date.month < 1 || Date.month > 12) cnvrtOK = 0;
Date.day   = (unsigned char)loctime->tm_mday;
if (Date.day < 1 || Date.day > 31) cnvrtOK = 0;
Date.hour  = (unsigned char)(loctime->tm_hour + 1);
if (Date.hour < 1 || Date.hour > 24) cnvrtOK = 0;
Date.minute= (unsigned char)(loctime->tm_min + 1);
if (Date.minute < 1 || Date.minute > 60) cnvrtOK = 0;
Date.second= (unsigned char)(loctime->tm_sec + 1);
if (Date.second < 1 || Date.second > 60) cnvrtOK = 0;

if (cnvrtOK)
  memcpy(oradt,&Date,7);
 else
   *oradt = '\0';

 return;

}
void cvtdmy (unsigned char *oradt, char *outdate)
{

      struct ORADATE {
            unsigned char   century;
            unsigned char   year;
            unsigned char   month;
            unsigned char   day;
            unsigned char   hour;
            unsigned char   minute;
            unsigned char   second;
      } Date;


      int day,month,year;

      memcpy(&Date,oradt,7);

      year = (Date.century-100)*100 + Date.year-100;
      month = Date.month;
      day = Date.day;
      /* sprintf(outdate,"%02d-%02d-%4d\0",day,month,year); */
      sprintf(outdate,"%02d-%02d-%4d",day,month,year);

      return;
}

void cvtdmyhms (unsigned char *oradt, char *outdate)
{

      struct ORADATE {
            unsigned char   century;
            unsigned char   year;
            unsigned char   month;
            unsigned char   day;
            unsigned char   hour;
            unsigned char   minute;
            unsigned char   second;
      } Date;


      int day,month,year;
      int hour,min,sec;

      memcpy(&Date,oradt,7);

      year = (Date.century-100)*100 + Date.year-100;
      month = Date.month;
      day = Date.day;
      hour = Date.hour - 1;
      min = Date.minute - 1;
      sec = Date.second - 1;

      /*sprintf(outdate,"%02d-%02d-%4d %02d:%02d:%02d\0";*/
      sprintf(outdate,"%02d-%02d-%4d %02d:%02d:%02d",
                day,month,year,hour,min,sec);

      return;
}

/*  Each server may have multiple connections to the DB.
 *  The OCI handles, which used to be global, are now grouped together
 *  In a data structure.  There is one such structure per DB connection.
 *
 *  There are two routines to deal with them:
 *          initOCIhandles: Initializes all the handles a connection needs
 *          freeOCIhandles: Frees all those handles.
 *  When the program is initialized it initializes an array of connections.
 *  Each thread is then assigned one of these connections and keeps reusing
 *  that connection.  The same connection may be shared by multiple threads.
 */
static void initOCIhandles(ora_cn_data_t *cn_dataP)
{
  text stmbuf[SQL_BUF_SIZE];
  OCIEnv *tpcenv;
  OCIServer *tpcsrv;
  OCIError *errhp;
  OCISvcCtx *tpcsvc;
  OCISession *tpcusr;
  OCIStmt *curi;
  dvoid *xmem;


  /* Initialize OCI handles in thread slot data.
   * This is called once per thread.
   * Specify that OCI library should not handle mutexing
   */
  OCIEnvInit(&tpcenv, OCI_ENV_NO_MUTEX, 0, (dvoid **)0);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid **)0);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid **)0);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid **)0);
  OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);

  OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SRVRCTXT, errhp);

  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)TPC_uid,
(ub4)strlen(TPC_uid),OCI_ATTR_USERNAME, errhp);
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)TPC_pwd,
(ub4)strlen(TPC_pwd), OCI_ATTR_PASSWORD, errhp);
  OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));
  OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_USERCTXT, errhp);

  /* run all transaction in serializable mode */
  OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);

  sprintf ((char *) stmbuf, SQLTXT);
  OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT);

  OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));

  OCIHandleFree(curi, OCI_HTYPE_STMT);
#ifdef SQL_TRACE
  /* Turn on the SQL_TRACE */
  OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, &xmem);
  sprintf ((char *) stmbuf, SQLTXT1);
  OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT);
  OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
  OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
#endif /* End SQL_TRACE */

  /*  Store the handles just initialized in the thread slot
   */
  cn_dataP->tpcenv = tpcenv;
```

```
  cn_dataP->tpcsrv = tpcsrv;
  cn_dataP->errhp = errhp;
  cn_dataP->tpcsvc = tpcsvc;
  cn_dataP->tpcusr = tpcusr;
  cn_dataP->curi = curi;
  cn_dataP->xmem = xmem;
}


static void freeOCIhandles(ora_cn_data_t *cn_dataP)
{

  OCIServer *tpcsrv;
  OCISession *tpcusr;
  OCIEnv *tpcenv;
  OCIError *errhp;
  OCISvcCtx *tpcsvc;

  if (tpcusr = cn_dataP->tpcusr) {
     err_printf("free_handles> OCIHandleFree tpcusr\n");
     OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
  }
  if (tpcsvc = cn_dataP->tpcsvc) {
     err_printf("free_handles> OCIHandleFree tpcsvc\n");
     OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
  }
  if (errhp = cn_dataP->errhp) {
     err_printf("free_handles> OCIHandleFree errhp\n");
     OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
  }
  if (tpcsrv = cn_dataP->tpcsrv) {
     err_printf("free_handles> OCIHandleFree tpcsrv\n");
     OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
  }
  if (tpcenv = cn_dataP->tpcenv) {
     err_printf("free_handles> OCIHandleFree tpcenv\n");
     OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);
  }
}

TPCexit ()
{
   if (lfp) {
      fclose (lfp);
      lfp = NULL;
   }
#ifndef ENCINA
   clean_cn((void *)connectionP);
   connectionP = NULL;
#endif
}

/*  clean_cn
 *
 *  Called to clean a connection.
 *  When using pthread this is registered during pthread_create
 *  and called automatically by pthread when the thread exits.
 */
static void clean_cn(void *ptr)
{

   /* free trans specific cursor handles first and later the ora handles */
   ora_cn_data_t *cn_dataP = (ora_cn_data_t *)ptr;

   if (cn_dataP != NULL) {
                  err_printf("clean_cn, Freeing OCI handles\n");
                  plnewdone(cn_dataP);
                  plpaydone(cn_dataP);
                  plorddone(cn_dataP);
                  pldeldone(cn_dataP);
                  plstodone(cn_dataP);

                  freeOCIhandles(cn_dataP);

                  err_printf("free_handles> free cn_dataP\n");
   }
}

static char *thread_state_to_str(int state)
{
    char *retval;
    switch(state) {
                  case SVR_STATE_NONE: retval = "None"; break;
                  case SVR_STATE_SENT: retval = "Sent"; break;
                  case SVR_STATE_REPLIED: retval = "Replied"; break;
                  case SVR_STATE_ERR: retval = "Err"; break;
                  default: retval = "unknown"; break;
    }
    return retval;
}


void print_rt_avg(total_tran_count_t *curP, total_tran_count_t *prevP)
{
    int i;
    double avg_queued_time;
    static char *names[] = {"id", "no", "pa", "os", "dl", "sl", "dn", "dp"};
    err_printf("bg_thread RT avg: ");

    for (i=0; i<=MAX_TRAN_TYPE; i++) {
       int num_trans = curP->tran[i].num - prevP->tran[i].num;
       double rt_diff = curP->tran[i].RT - prevP->tran[i].RT;
       PRINT_AV(rt_diff, num_trans, names[i]);
                  if (i == DELIVERY_TRANS && num_trans > 0) {
                     rt_diff = curP->dvry_queue_time - prevP->dvry_queue_time;
                     PRINT_AV(rt_diff, num_trans, "qd");
                  }
    }
    fprintf(stderr, "\n");
}

void print_tran_info(tran_spec_info_t *tran_copy)
{
    int i;
    struct payinstruct *paP;
    struct newinstruct *noP;

    switch (tran_copy->tran_type) {
                  case DIST_NEWO_TRANS:
       case NEWO_TRANS:
          noP = &(tran_copy->tran_info.no.newin);
          err_printf("bg_thread: TPCnew : w_id %d, d_id %d, c_id %d\n",
            noP->w_id,
            noP->d_id,
            noP->c_id);
                  for (i=0; i<15; i++) {
                              if (noP->ol_i_id[i] == 0) {
                  break;
                              }
                              err_printf("bg_thread TPCnew : (%d)i_id %d sup_w_id %d Qty
%d\n",
                                          i,
                                          noP->ol_i_id[i],
                                          noP->ol_supply_w_id[i],
                                          noP->ol_quantity[i]);
                  }
            break;

                  case DIST_PAY_TRANS:
       case PAYMENT_TRANS:
           paP = &(tran_copy->tran_info.pa.payin);
           err_printf("bg_thread: TPCpay: w_id %d, D_id %d, C_w_id %d, c_id %d,
bylastname %d, amount %.2f, c_last %s \n",
            paP->w_id,
            paP->d_id,
            paP->c_w_id,
            paP->c_id,
            paP->bylastname,
            paP->h_amount,
            paP->c_last);
            break;

      case ORDER_STAT_TRANS:
           err_printf("bg_thread: TPCord: w_id %d, d_id %d, c_id %d, bylastname %d, c_last
%s \n",
            tran_copy->tran_info.os.ordin.w_id,
            tran_copy->tran_info.os.ordin.d_id,
            tran_copy->tran_info.os.ordin.c_id,
            tran_copy->tran_info.os.ordin.bylastname,
            tran_copy->tran_info.os.ordin.c_last);
            break;

       case DELIVERY_TRANS:
          err_printf("bg_thread: TPCdel: w_id %d, o_carrier_id %d, %fqtime, %d
in_timing_int\n",
            tran_copy->tran_info.dl.delin.w_id,
            tran_copy->tran_info.dl.delin.o_carrier_id,
            tran_copy->tran_info.dl.delin.qtime,
            tran_copy->tran_info.dl.delin.in_timing_int);
            break;

       case STOCK_TRANS:
          err_printf("bg_thread: TPCsto: w_id %d, d_id %d, threashold %d\n",
            tran_copy->tran_info.sl.stoin.w_id,
            tran_copy->tran_info.sl.stoin.d_id,
            tran_copy->tran_info.sl.stoin.threshold);
            break;
       default:
            err_printf("bg_thread: bad tran\n");
                              break;
                  }
}

static void copy_tran_info(void *dataP, int type, tran_spec_info_t *outP)
{
    outP->tran_type = type;
    switch (type) {
                  case NEWO_TRANS:
                  case DIST_NEWO_TRANS:
                     outP->tran_info.no = *(struct newstruct *)dataP;
```

```
                break;
            case PAYMENT_TRANS:
            case DIST_PAY_TRANS:
                outP->tran_info.pa = *(struct paystruct *)dataP;
                break;
            case ORDER_STAT_TRANS:
                outP->tran_info.os = *(struct ordstruct *)dataP;
                break;
            case STOCK_TRANS:
                outP->tran_info.sl = *(struct stostruct *)dataP;
                break;
            case DELIVERY_TRANS:
                outP->tran_info.dl = *(struct delstruct *)dataP;
                break;
            defaut:
                memset(&outP->tran_info, '\0', sizeof(outP->tran_info));
                break;
    }
}

static void check_threads(total_tran_count_t *tran_ctP)
{
    struct timezone tz;
    int num_per_state[NUM_STATES];
    int total_stuck = 0;
    static int init_printed = 0;
    int total_tran_err;
    tran_spec_info_t tran_copy;

    pthread_mutex_lock(&init_lock);

    if (cn_array && (num_connections > 0)) {
        int i,j;

                memset(num_per_state, '\0', sizeof(num_per_state));
        memset(tran_ctP, '\0', sizeof(*tran_ctP));
        for (i=0; i<num_connections; i++) {
            struct timeval cur_time;
            struct timeval *client_timeP;
                    ora_cn_data_t *cnP = &cn_array[i];
            int time_diff;
            int delta = 60;
                    total_tran_count_t *statP = &cnP->stat;

            for (j=0; j<=MAX_TRAN_TYPE; j++) {
                tran_ctP->tran[j].num += statP->tran[j].num;
                tran_ctP->tran[j].errs += statP->tran[j].errs;
                tran_ctP->tran[j].RT += statP->tran[j].RT;
                tran_ctP->errors += statP->tran[j].errs;
            }
                    tran_ctP->dvry_queue_time += statP->dvry_queue_time;

                    /*  Make a copy of the tran specifc data structure here
                     *  Since we are not performing any locking or any other
                     *  synchronization, we have to be careful: Copy the tran
                     *  data before getting the current time.  If the tran
                     *  has been stuck for a while, it is likely that
                     *  the copy is good.
                     */
            copy_tran_info(cnP->cur_tran_dataP, cnP->cur_tran_type, &tran_copy);
            gettimeofday(&cur_time, &tz);

            time_diff = cur_time.tv_sec - cnP->tran_time.tv_sec;
            if (time_diff > delta) {
                num_per_state[cnP->state] ++;
                total_stuck++;
                if (!cnP->printed) {
                    err_printf("bg_thread: thread index %d state %s tran %d stuck for %d sec\n",
                            i,
                            thread_state_to_str(cnP->state),
                            cnP->cur_tran_type,
                            time_diff);
                                /*  Print some tran specific info here off the copy  */
                                if (cnP->state == SVR_STATE_SENT)
                        print_tran_info(&tran_copy);
                    cnP->printed = 1;
                }
            } else if (cnP->printed) {
                err_printf("bg_thread: thread index %d state %s tran %d unstuck.\n",
                            i,
                            thread_state_to_str(cnP->state),
                            cnP->cur_tran_type);

                cnP->printed = 0;
            }
        }
        if (total_stuck > 0) {
            err_printf("bg_thread: Summary %d stuck: ",total_stuck);
            for (i=0; i<NUM_STATES; i++) {
                if (num_per_state[i] > 0) {
                    fprintf(stderr, "%d %s, ",
                            num_per_state[i], thread_state_to_str(i));
                }
            }
            fprintf(stderr, "\n");
        }
                total_tran_err = 0;
            for (i=0; i<=MAX_TRAN_TYPE; i++)
                total_tran_err += tran_ctP->tran[i].errs;
            if (total_tran_err > 0) {
                err_printf("bg_thread: %d errs: %d no, %d pa, %d os, %d sl\n",
                        total_tran_err,
                        tran_ctP->tran[NEWO_TRANS].errs,
                        tran_ctP->tran[PAYMENT_TRANS].errs,
                        tran_ctP->tran[ORDER_STAT_TRANS].errs,
                        tran_ctP->tran[STOCK_TRANS].errs);
            }
        }
        pthread_mutex_unlock(&init_lock);
}

/*
 *  time_diff_ms
 *      Return the difference in miliseconds between two times
 */
int time_diff_ms(t2, t1)
  struct timeval *t2, *t1;
{
    int t_diff;

    t_diff = (t2->tv_usec + 1000000 - t1->tv_usec + 500) / 1000 +
            (t2->tv_sec - t1->tv_sec - 1) * 1000;

    return(t_diff);
}

/*
 *  A background thread that keeps tabs on the state of all the
 *  threads of the server.  (For Debug)
 */
static void *bg_thread(void *argP)
{
    int i;
    int total_newo, total_tran_err;
    int total_trans;
    struct timeval cur_time;
    struct timezone tz;
    struct timeval time_reported[2];
    total_tran_count_t tran_ct, tran_reported[2];

    gettimeofday(&time_reported[0],&tz);
    time_reported[1] = time_reported[0];

    memset(&tran_reported[0], '\0', 2 * sizeof(tran_reported[0]));

    while (1) {
        double time_diff1, time_diff2;
        double tran_diff1, tran_diff2;
        double err_diff1, err_diff2;

        check_threads(&tran_ct);

        total_tran_err = tran_ct.errors;
        total_newo = tran_ct.tran[NEWO_TRANS].num;
                total_trans = 0;
                for (i=0; i<=MAX_TRAN_TYPE; i++) total_trans += tran_ct.tran[i].num;

        gettimeofday(&cur_time,&tz);
        time_diff1 = time_diff_ms(&cur_time,&time_reported[0]);
        tran_diff1 = total_newo - tran_reported[0].tran[NEWO_TRANS].num;
        err_diff1 = total_tran_err - tran_reported[0].errors;
        time_diff2 = time_diff_ms(&cur_time,&time_reported[1]);
        tran_diff2 = total_newo - tran_reported[1].tran[NEWO_TRANS].num;
        err_diff2 = total_tran_err - tran_reported[1].errors;
        if (total_trans != 0) {
            err_printf("bg_thread: TPM: %.0f (last %.0f sec), %.0f (last %.0f sec)\n",
                    tran_diff1 / time_diff1 * 60000, time_diff1 / 1000.,
                    tran_diff2 / time_diff2 * 60000, time_diff2 / 1000.);
            /* print av server response time for all transactions */
                    print_rt_avg(&tran_ct, &tran_reported[1]);
        }
        if (err_diff2 != 0) {
            err_printf("bg_thread: errPM %.1f (last %.0f sec)\n",
                    err_diff2 / time_diff2 * 60000, time_diff2 / 1000.);
        }
        tran_reported[0] = tran_reported[1];
        tran_reported[1] = tran_ct;
        time_reported[0] = time_reported[1];
        time_reported[1] = cur_time;
        sleep(60);
    }
    pthread_mutex_lock(&init_lock);
    pthread_mutex_unlock(&init_lock);
}

void start_bg_thread( void )
{
    int rc;
    pthread_attr_t attr;
    pthread_t thread;
```

```
    err_printf("> start_bg_thread\n");
    if (rc = pthread_attr_create(&attr)) {
        err_printf("start_bg_debug_thread: pthread_attr_create failed: %d\n", rc);
        return;
    }
    UNCOND_EVENT("Creating thread for bg_thread");
    if ((rc = pthread_create(&thread,
                        attr,
                        bg_thread,
                        (pthread_addr_t)NULL)) != 0) {
        err_printf("start_bg_debug_thread: pthread_create failed: %d\n", rc);
        return;
    }
    UNCOND_EVENT("Detaching bg thread");
    if ( rc = pthread_detach(&thread) != 0) {
        err_printf("start_bd_debug_thread: pthread_detach failed %d\n", rc);
        return;
    }
    err_printf("< start_bg_thread\n");
}



/*
 *  init_global_data
 *
 *  Called once during initialization to initialize the (thread) global data:
 *                  OCI Handles
 *                  Transaction data structures and cursors
 */
static void init_global_data(void)
{
#ifdef ENCINA
  int status = 0;
  pthread_mutex_init(&dvry_log_lock, pthread_mutexattr_default);
  pthread_mutex_init(&key_lock, pthread_mutexattr_default);
  pthread_mutex_init(&init_lock, pthread_mutexattr_default);
  err_printf("init_global_data> grabbing pthread_mutex_lock\n");
  pthread_mutex_lock(&key_lock);
  if (!key_init) {
      if(status = pthread_keycreate(&thread_key, clean_cn)) {
          fprintf(stderr, "init_global_data : pthread_keycreate failed: %d\n", status);
          exit(20);
      }
      key_init = 1;
  }
  pthread_mutex_unlock(&key_lock);
#ifdef PREALLOC_CN
  {
    int i, env_val;
    char *env_str;
    extern int num_deferred_dvry_threads;
    extern int num_worker_threads;

    pthread_mutex_lock(&init_lock);
    if (num_worker_threads > 0 &&
        num_worker_threads < 200) {
        num_connections += num_worker_threads;
    }

    if (num_deferred_dvry_threads > 0 &&
        num_deferred_dvry_threads < 20) {
        num_connections += num_deferred_dvry_threads;
    }

    if (num_connections < 1) num_connections = 1;
    cn_array = (ora_cn_data_t *)calloc(num_connections, sizeof(ora_cn_data_t));
    err_printf("Preallocating %d connections tooracle\n", num_connections);
    if (cn_array == NULL) {
                err_printf("Failed to allocated %dentried for CN array\n",
                                num_connections);
                exit(3);
    }
    for (i=0; i<num_connections; i++) {
        init_cn_data(&cn_array[i]);
    }
    pthread_mutex_unlock(&init_lock);
    start_bg_thread();
  }
#endif
#else
  connectionP = malloc(sizeof(*connectionP));
  memset(connectionP, (char)0, sizeof(*connectionP));
  init_cn_data(connectionP);
#endif
}

TPCinit (id, uid, pwd)
int id;
char *uid;
char *pwd;
{

    int i;
    extern char *tpcc_serverName;
    char *home_dir = getenv("HOME");

    err_printf("TPCinit id %d, uid %s pwd %s\n", id, uid, pwd);
    proc_no = id;
    sprintf (delivery_file_name,
                    "%s/runs/deliveries/tpcc_%d.%s.del",
                    home_dir ? home_dir : "/home/encina",
                    proc_no, tpcc_serverName);
    lfp = NULL;  /*  The file will be opened on demand  */

    /* Using multithreaded Oracle clients, OCI is multithreaded mode. */
    OCIInitialize(OCI_THREADED,(dvoid *)0,0,0,0);

    TPC_uid = uid;
    TPC_pwd = pwd;

    init_global_data();

    return (0);
}

/*
 *  init_cn_data
 *              Called once for each thread to initialize the thread
 *              global data structure.
 */
static void init_cn_data(ora_cn_data_t *dataP)
{
    UNCOND_EVENT("init_cn_data: Initializing connection to DB.");
    initOCIhandles(dataP);

    UNCOND_EVENT("init_cn_data: plnewinit");
    plnewinit(dataP);
    UNCOND_EVENT("init_cn_data: plpayinit");
    plpayinit(dataP);
    UNCOND_EVENT("init_cn_data: plordinit");
    plordinit(dataP);
    UNCOND_EVENT("init_cn_data: pldelinit");
    pldelinit(dataP);
    UNCOND_EVENT("init_cn_data: plstoinit");
    plstoinit(dataP);
    UNCOND_EVENT("Initialized connection to DB.");
}

#ifdef ENCINA
/*
 * get_cn - Gets a connection to the DB.
 * Each thread is assigned a connection and keeps reusing it.
 *
 *  For debug: each connections contains some state about the
 *              thread which includes the time this call was made, the transaction
 *              being performed and some tran and response time stats.
 */
static ora_cn_data_t *get_cn( int tran, void *tran_dataP )
{
    ora_cn_data_t *dataP;
    struct timezone tz;
    struct timeval cur_time;

    /*  Get a connection structure.
     *  Each thread always uses the same connection.
     *  The first time the thread tries to talk to the DB it creates
     *  a connection, initializes it and stores it in a thread global
     *  data structure.
     */
    if (num_connections == 1) {
                dataP = &cn_array[0];
    } else {
        pthread_getspecific(thread_key, (pthread_addr_t *)&dataP);
    }
    if (dataP == NULL) {  /* No connection assigned to this thread */
                gettimeofday(&cur_time,&tz);

                pthread_mutex_lock(&init_lock);  /*  Initialize the connections one at a time */
        err_printf("get_cn> initializing thread slot\n");
#ifdef PREALLOC_CN
                if (cn_id >= num_connections) {
                    err_printf("Too many threads, not enough connections\n");
                    exit(3);
                }
                dataP = &cn_array[cn_id++];
#else
                dataP = (ora_cn_data_t *)malloc(sizeof(ora_cn_data_t));
                memset(dataP, (char)0, sizeof(*dataP));
                init_cn_data(dataP);
#endif
                dataP->connect_time = cur_time.tv_sec;
                dataP->calls = 0;
        err_printf("get_cn> initialized connection 0x%x\n", dataP);
                pthread_mutex_unlock(&init_lock);

        pthread_setspecific(thread_key, dataP);  /*  Store it  */

                err_printf("get_cn> initialized connection\n");
```

```
  }


  gettimeofday(&cur_time,&tz);

  /*  Keep track of how much time the thread is idle */
  if (dataP->state != SVR_STATE_NONE) {
              tran_info_t *statP = &dataP->stat.tran[0];
              double RT;
    RT = cur_time.tv_usec - dataP->tran_time.tv_usec;
    RT = RT / 1e6 + cur_time.tv_sec - dataP->tran_time.tv_sec;

              statP->RT += RT;
              statP->num++;
  }

  /*  Keep some state for debug  */
  dataP->state = SVR_STATE_SENT;
  dataP->cur_tran_type = tran;
  dataP->cur_tran_dataP = tran_dataP;
  dataP->tran_time = cur_time;
  dataP->calls++;
  return dataP;
}

/*
 * done_with_cn - Done with a conection - -keep stats -- FOR DEBUG ONLY --
 */
static void done_with_cn(ora_cn_data_t *dataP, int tran, void *tran_dataP )
{
  struct timezone tz;
  struct timeval cur_time;
  tran_info_t *statP = &dataP->stat.tran[tran];
  double RT;

  gettimeofday(&cur_time,&tz);

  RT = cur_time.tv_usec - dataP->tran_time.tv_usec;
  RT = RT / 1e6 + cur_time.tv_sec - dataP->tran_time.tv_sec;

  statP->RT += RT;
  statP->num++;
  dataP->tran_time = cur_time;
  dataP->state = SVR_STATE_REPLIED;
  if (tran == DELIVERY_TRANS && tran_dataP) {
              /*  This is a delivery transaction.
               *  keep track of the average time it spent on the queue
               */
              struct delstruct *str = (struct delstruct *)tran_dataP;
              double ct = cur_time.tv_usec;
              ct = ct / 1e6 + cur_time.tv_sec;
              dataP->stat.dvry_queue_time += (ct - str->delin.qtime);
  }

  if (RT > 45) {
              err_printf("STATS: Tran %d RT %.3f, cn: %d trans RT total %.3f avg %.3f\n",
                          tran, RT, statP->num, statP->RT, statP->RT / statP->num);
  }
}

/*
 *  ora_cn_err
 *
 *  Called to keep track of errors with oracle connections and possibly
 *  reestablish a connection in case it is bad.
 */
static void ora_cn_err(ora_cn_data_t *dataP)
{
  struct timezone tz;
  struct timeval cur_time;
  gettimeofday(&cur_time,&tz);

  dataP->errors ++;
  if (dataP->calls-1 == dataP->calls_last_err)
    dataP->consecutive_errs ++;
  else
              dataP->consecutive_errs = 1;
  dataP->calls_last_err = dataP->calls;
  err_printf("ora_cn_err %d errors (%d consecutive) connected %d sec, %d calls, %d
cals_last_err\n",
                dataP->errors, dataP->consecutive_errs,
                cur_time.tv_sec - dataP->connect_time,
                dataP->calls, dataP->calls_last_err);
  if (dataP->consecutive_errs > ORA_RECONNECT_THRESHHOLD &&
              (cur_time.tv_sec - dataP->connect_time) >
MIN_TIME_BETWEEN_RECONNECTS) {
      /*  This connection is not behaving, free it.
               *  The next time this thread needs a connection it will reconnect
               */
              err_printf("ora_cn_err: Giving up on the connection\n");
              clean_cn((void *)dataP);
              pthread_setspecific(thread_key, NULL);
  }
  dataP->state = SVR_STATE_ERR;
  dataP->tran_time = cur_time;
}
#endif

TPCnew (str)
struct newstruct *str;
{
 ora_cn_data_t *cn_dataP = get_cn(NEWO_TRANS, (void *)str);
 struct timeval cur_time;
 global_newOrder_t *newP = cn_dataP->globals;
 int i;

#if defined(ISO1) || defined(ISO7)
  int reread;
  char sdate[30];
#endif

  /*
   * copy from struct str to previous globals
   * and chnaged the globals to local vars */
  newP->w_id = str->newin.w_id;
  newP->d_id = str->newin.d_id;
  newP->c_id = str->newin.c_id;

  for (i = 0; i < 15; i++) {
    newP->nol_i_id[i] = str->newin.ol_i_id[i];
    newP->nol_supply_w_id[i] = str->newin.ol_supply_w_id[i];
    newP->nol_quantity[i] = str->newin.ol_quantity[i];
  }
  newP->retries = 0;
  vgetdate(newP->cr_date);

  if (str->newout.terror = plnew(cn_dataP)) {
    err_printf("plnew> returning, terror %d, retries %d\n",
                                str->newout.terror, newP->retries);
    if (str->newout.terror != RECOVERR)
      str->newout.terror = IRRECERR;
    str->newout.retry = newP->retries;
    ora_cn_err(cn_dataP);
    return (-1);
  }

  /* fill in date for o_entry_d from time in beginning of txn*/
  cvtdmyhms(newP->cr_date,newP->o_entry_d);

  str->newout.terror = NOERR;
  str->newout.o_id = newP->o_id;
  str->newout.o_ol_cnt = newP->o_ol_cnt;
  strncpy (str->newout.c_last, newP->c_last, 17);
  strncpy (str->newout.c_credit, newP->c_credit, 3);
  str->newout.c_discount = (float)(newP->c_discount)/10000;
  str->newout.w_tax = (float)(newP->w_tax)/10000;
  str->newout.d_tax = (float)(newP->d_tax)/10000;
  strncpy (str->newout.o_entry_d, newP->o_entry_d, 20);
  str->newout.total_amount = newP->total_amount;

  for (i = 0; i < newP->o_ol_cnt; i++) {
    strncpy (str->newout.i_name[i], newP->i_name[i], 25);
    str->newout.s_quantity[i] = newP->s_quantity[i];
    str->newout.brand_generic[i] = newP->brand_gen[i];
    str->newout.i_price[i] = (float)(newP->i_price[i])/100;
    str->newout.ol_amount[i] = (float)(newP->nol_amount[i])/100;
  }
  if (newP->status) {
    strcpy (str->newout.status, "Item number is not valid");
  } else {
    str->newout.status[0] = '\0';
  }
  str->newout.retry = newP->retries;
  done_with_cn(cn_dataP,
                              newP->o_all_local ? NEWO_TRANS : DIST_NEWO_TRANS,
                              (void *)str);
  return(0);
}



TPCpay (str)
struct paystruct *str;
{
  ora_cn_data_t *cn_dataP =
            get_cn(str->payin.w_id == str->payin.c_w_id ? PAYMENT_TRANS :
DIST_PAY_TRANS,
                  (void *)str);
  global_payment_t *payP = cn_dataP->payP;

  payP->w_id = str->payin.w_id;
  payP->d_id = str->payin.d_id;
  payP->c_w_id = str->payin.c_w_id;
  payP->c_d_id = str->payin.c_d_id;
  payP->h_amount = str->payin.h_amount;
  payP->bylastname = str->payin.bylastname;

  vgetdate(payP->cr_date);

  if (payP->bylastname) {
```

```
    payP->c_id = 0;
    strncpy (payP->c_last, str->payin.c_last, 17);
  }
  else {
    payP->c_id = str->payin.c_id;
    strcpy (payP->c_last, " ");
  }
  payP->retries = 0;

  if (str->payout.terror = plpay(cn_dataP)) {
    err_printf("plpay> returning, terror %d, retries %d\n",
            str->payout.terror, payP->retries);
    if (str->payout.terror != RECOVERR)
      str->payout.terror = IRRECERR;
    str->payout.retry = payP->retries;
    ora_cn_err(cn_dataP);
    return (-1);
  }

  cvtdmyhms(payP->cr_date,payP->h_date);
  cvtdmy(payP->c_since,payP->c_since_d);


  str->payout.terror = NOERR;
  strncpy (str->payout.w_street_1, payP->w_street_1, 21);
  strncpy (str->payout.w_street_2, payP->w_street_2, 21);
  strncpy (str->payout.w_city, payP->w_city, 21);
  strncpy (str->payout.w_state, payP->w_state, 3);
  strncpy (str->payout.w_zip, payP->w_zip, 10);
  strncpy (str->payout.d_street_1, payP->d_street_1, 21);
  strncpy (str->payout.d_street_2, payP->d_street_2, 21);
  strncpy (str->payout.d_city, payP->d_city, 21);
  strncpy (str->payout.d_state, payP->d_state, 3);
  strncpy (str->payout.d_zip, payP->d_zip, 10);
  str->payout.c_id = payP->c_id;
  strncpy (str->payout.c_first, payP->c_first, 17);
  strncpy (str->payout.c_middle, payP->c_middle, 3);
  strncpy (str->payout.c_last, payP->c_last, 17);
  strncpy (str->payout.c_street_1, payP->c_street_1, 21);
  strncpy (str->payout.c_street_2, payP->c_street_2, 21);
  strncpy (str->payout.c_city, payP->c_city, 21);
  strncpy (str->payout.c_state, payP->c_state, 3);
  strncpy (str->payout.c_zip, payP->c_zip, 10);
  strncpy (str->payout.c_phone, payP->c_phone, 17);
  strncpy (str->payout.c_since, payP->c_since_d, 11);
  strncpy (str->payout.c_credit, payP->c_credit, 3);
  str->payout.c_credit_lim = (float)(payP->c_credit_lim)/100;
  str->payout.c_discount = (float)(payP->c_discount)/10000;
  str->payout.c_balance = (float)(payP->c_balance)/100;
  strncpy (str->payout.c_data, payP->c_data, 201);
  strncpy (str->payout.h_date, payP->h_date, 20);
  str->payout.retry = payP->retries;
  done_with_cn(cn_dataP,
                        str->payin.w_id == str->payin.c_w_id ? PAYMENT_TRANS :
DIST_PAY_TRANS,
                        (void *)str);
  return (0);

}



TPCord (str)
struct ordstruct *str;
{
  ora_cn_data_t *cn_dataP = get_cn(ORDER_STAT_TRANS, (void *)str);
  global_order_t *ordP = cn_dataP->ordP;
  int i;


  ordP->w_id = str->ordin.w_id;
  ordP->d_id = str->ordin.d_id;
  ordP->bylastname = str->ordin.bylastname;
  if (ordP->bylastname) {
    ordP->c_id = 0;
    strncpy (ordP->c_last, str->ordin.c_last, 17);
  }
  else {
    ordP->c_id = str->ordin.c_id;
    strcpy (ordP->c_last, " ");
  }
  ordP->retries = 0;

  if (str->ordout.terror = plord (cn_dataP)) {
    err_printf("plord> returning, terror %d, retries %d\n",
            str->ordout.terror, ordP->retries);
    if (str->ordout.terror != RECOVERR)
      str->ordout.terror = IRRECERR;
    str->ordout.retry = ordP->retries;
    ora_cn_err(cn_dataP);
    return (-1);
  }

  str->ordout.terror = NOERR;
  str->ordout.c_id = ordP->c_id;
  strncpy (str->ordout.c_last, ordP->c_last, 17);
  strncpy (str->ordout.c_first, ordP->c_first, 17);
  strncpy (str->ordout.c_middle, ordP->c_middle, 3);
  str->ordout.c_balance = ordP->c_balance/100;
  str->ordout.o_id = ordP->o_id;
  strncpy (str->ordout.o_entry_d, ordP->o_entry_d, 20);
  if ( ordP->o_carrier_id == 11 )
     str->ordout.o_carrier_id = 0;
  else
     str->ordout.o_carrier_id = ordP->o_carrier_id;
  str->ordout.o_ol_cnt = ordP->o_ol_cnt;
  for (i = 0; i < ordP->o_ol_cnt; i++) {
    ordP->ol_delivery_d[i][10] = '\0';
    if ( !strcmp(ordP->ol_delivery_d[i],"15-09-1911") )
                strncpy(ordP->ol_delivery_d[i],"NOT DELIVR",10);
    str->ordout.ol_supply_w_id[i] = ordP->ol_supply_w_id[i];
    str->ordout.ol_i_id[i] = ordP->ol_i_id[i];
    str->ordout.ol_quantity[i] = ordP->ol_quantity[i];
    str->ordout.ol_amount[i] = (float)(ordP->ol_amount[i])/100;
    strncpy (str->ordout.ol_delivery_d[i], ordP->ol_delivery_d[i], 11);
  }
  str->ordout.retry = ordP->retries;
  done_with_cn(cn_dataP, ORDER_STAT_TRANS, (void *)str);
  return (0);

}



TPCdel (str)
struct delstruct *str;
{
  ora_cn_data_t *cn_dataP = get_cn(DELIVERY_TRANS, (void *)str);
  global_delivery_t *delP = cn_dataP->delP;
  double tr_end, tr_begin;
  int i, skipped;
  struct timeval cur_time;
  static int tran_cntr=0;
  int pos, len;
  int queue_time, start_time, end_time;
  char stdout_buf[1024];

  /*  Open the delivery log file if needed  */
  if (lfp == NULL) {
    pthread_mutex_lock(&dvry_log_lock);
    if (lfp == NULL) {
      if ((lfp = fopen (delivery_file_name, "w")) == NULL) {
        fprintf (stderr, "Error in TPC-C server: Failed to open %s\n",
                delivery_file_name);
        pthread_mutex_unlock(&dvry_log_lock);
                done_with_cn(cn_dataP, DELIVERY_TRANS, (void *)str);
        return(-1);
      }
                err_printf("Opened delivery file %s\n", delivery_file_name);
    }
    pthread_mutex_unlock(&dvry_log_lock);
  }


  gettimeofday(&cur_time,NULL);
  tr_begin = (double)cur_time.tv_sec + 1.0e-6 * (double)cur_time.tv_usec;
  start_time = cur_time.tv_sec;

  delP->w_id = str->delin.w_id;
  delP->o_carrier_id = str->delin.o_carrier_id;
  delP->retries = 0;
  vgetdate(delP->cr_date);

  str->delout.terror = pldel(cn_dataP);

  gettimeofday(&cur_time,NULL);
  tr_end = (double)cur_time.tv_sec + 1.0e-6 * (double)cur_time.tv_usec;
  end_time = cur_time.tv_sec;

  /*  Make sure all the data pertaining to a single
   *  delivery record is written atomically
   */
  pthread_mutex_lock(&dvry_log_lock);

#ifndef USE_ORACLE_DVRY_FORMAT
  queue_time = str->delin.qtime;
  pos = 0;
  ++tran_cntr;
#ifdef USE_LONG_DVRY_FORMAT
  sprintf(&stdout_buf[pos], "----Transaction %d started----\n", tran_cntr);
  pos += strlen(&stdout_buf[pos]);
  sprintf(&stdout_buf[pos], "queued-time: %.6f %s",
                str->delin.qtime, ctime((time_t*)&queue_time));
  pos += strlen(&stdout_buf[pos]);
  sprintf(&stdout_buf[pos], "start-time: %.6f %s",
                tr_begin, ctime((time_t*)&start_time));
  pos += strlen(&stdout_buf[pos]);
  sprintf(&stdout_buf[pos], "W_ID: %d, CARRIER_ID: %d \n",
                str->delin.w_id, str->delin.o_carrier_id);
  pos += strlen(&stdout_buf[pos]);
```

```
  if (str->delout.terror == DEL_ERROR) {
    sprintf(&stdout_buf[pos], "Delivery transaction failed (DEL_ERROR)");
    pos += strlen(&stdout_buf[pos]);
  } else if (str->delout.terror != 0) {
    sprintf(&stdout_buf[pos], "Delivery transaction failed (%d)",
                    str->delout.terror);
    pos += strlen(&stdout_buf[pos]);
  } else {
    skipped = 0;
    for (i = 0; i < 10; i++) {
      if (delP->del_o_id[i] <= 0) {
        skipped++;
        fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
                delP->w_id, i + 1);
        sprintf(&stdout_buf[pos], "D_ID %d has no neworders.\n", i+1);
        pos += strlen(&stdout_buf[pos]);
      } else {
        sprintf(&stdout_buf[pos], "D_ID: %d, O_ID: %d \n", i+1, delP->del_o_id[i]);
        pos += strlen(&stdout_buf[pos]);
      }
    }
    fprintf(lfp, "%send-time:%.6f %s\n", stdout_buf,
                    tr_end, ctime((time_t *)&end_time));
  }
#else
  pos += sprintf(&stdout_buf[pos], "--Tran %d Queue %.3f Start %.3f\n",
                      tran_cntr, str->delin.qtime, tr_begin);
  pos += sprintf(&stdout_buf[pos], "W_ID: %d, CARRIER_ID: %d",
              str->delin.w_id, str->delin.o_carrier_id);

  if (str->delout.terror == DEL_ERROR) {
    pos += sprintf(&stdout_buf[pos],
                                "\nDelivery transaction failed (DEL_ERROR)\n");
  } else if (str->delout.terror != 0) {
    pos += sprintf(&stdout_buf[pos], "Delivery transaction failed (%d)",
                str->delout.terror);
  } else {
    int skipped[10];
    int num_skipped = 0;
    for (i = 0; i < 10; i++) {
      if (delP->del_o_id[i] <= 0) {
        skipped[i] = 1;
                    num_skipped ++;
      } else {
                    skipped[i] = 0;
      }
      pos += sprintf(&stdout_buf[pos], " %d", delP->del_o_id[i]);
    }
    pos += sprintf(&stdout_buf[pos], "\n");
    if (num_skipped > 0) {
      for (i=0; i<10; i++) {
                if (skipped[i] == 1) {
                        pos += sprintf(&stdout_buf[pos],
                                                "D_ID %d has no neworders.\n", i+1);
                }
              }
    }
  }

  fprintf(lfp, "%send-time:%.3f\n", stdout_buf, tr_end);
#endif

  fflush (lfp);


#else
  fprintf(lfp, "%s%d%d %f %f %d %d",
                  str->delout.terror == DEL_ERROR ? "DEL_ERROR: " :
                  str->delout.terror != 0 ? "ERROR" : "",
                  str->delin.in_timing_int,
        (tr_end - str->delin.qtime) <= DELRT ? 1 : 0,
         str->delin.qtime, tr_end, delP->w_id, delP->o_carrier_id);
  if (str->delout.terror != DEL_ERROR) {
    for (i = 0; i < 10; i++) {
      fprintf (lfp, " %d %d",i + 1, delP->del_o_id[i]);
      if (delP->del_o_id[i] <= 0) {
        fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
                delP->w_id, i + 1);
      }
    }
  }
  fprintf (lfp, " %d\n", delP->retries);
  fflush (lfp);

#endif
  pthread_mutex_unlock(&dvry_log_lock);

  str->delout.terror = NOERR;
  str->delout.retry = delP->retries;
  done_with_cn(cn_dataP, DELIVERY_TRANS, (void *)str);
  return (0);

}
```

```
TPCsto (str)
struct stostruct *str;
{
  ora_cn_data_t *cn_dataP = get_cn(STOCK_TRANS, (void *)str);
  global_stock_t *stoP = cn_dataP->stoP;

  stoP->w_id = str->stoin.w_id;
  stoP->d_id = str->stoin.d_id;
  stoP->threshold = str->stoin.threshold;
  stoP->retries = 0;

  if (str->stoout.terror = plsto (cn_dataP)) {
    err_printf("plsto> returning, terror %d, retries %d\n",
            str->stoout.terror, stoP->retries);

    if (str->stoout.terror != RECOVERR)
      str->stoout.terror = IRRECERR;
    str->stoout.retry = stoP->retries;
    ora_cn_err(cn_dataP);
    return (-1);
  }

  str->stoout.terror = NOERR;
  str->stoout.low_stock = stoP->low_stock;
  str->stoout.retry = stoP->retries;
  done_with_cn(cn_dataP, STOCK_TRANS, (void *)str);
  return (0);

}
```

tpcc_trans.tidl

```
/*
 * Copyright (C) 1991, 1990 Transarc Corporation
 * All Rights Reserved
 */
/*
 *  tpcc.tacf -- attribute configuration file for tpcc server.
 * used for transparent binding
 *
 * $Revision: 1.1 $
 * $Date: 1997/04/20 11:58:06 $
 * $Log:        tpcc.tacf,v $
Revision 4.2  95/05/16  10:55:49  10:55:49  tpcc (TPCC Benchmark)
Added necessary RCS ident strings

 */

[implicit_handle (mon_handle_t handle)]
interface tpccTransactions
{
}
```

trans. tpcc_tidl

```
/*
 * id: $id: $
 *
 * component_name: encina benchmarks
 *
 * the following functions list may not be complete.
 * functions defined by/via macros may not be included.
 *
 * functions:
 *    <fill_me_in>
 *
 * origins: transarc corp.
 *
 * (c) copyright transarc corp. 1995, 1993
 * all rights reserved
 * licensed materials - property of transarc
 *
 * us government users restricted rights - use, duplication or
 * disclosure restricted by gsa adp schedule contract with transarc corp
 */
/*
 * history
 * $talog: $
 */

/*
 * tpcc_trans.tidl -- interface definition file for tpccserver.
 *
 * $revision: 1.11 $
 * $date: 1995/10/20 21:55:05 $
 * $log:        tpcc.tidl,v $
 */
```

```
[uuid(955d7288-e672-11d0-bcef-9e621234aa77), version(1.0)]

interface tpccTrans
{
import "tpm/mon/mon_handle.idl";
import "tpcc_type.idl";

[nontransactional] void
        expTPCCNewOrder([in]  trpc_handle_t                handle,
                        [in,out] newOrder_data_t *dataP,
                        [out] trpc_status_t  * trpcStatus);

[nontransactional] void
        impTPCCNewOrder([in,out]newOrder_data_t *dataP,
                        [out] trpc_status_t  * trpcStatus);

[nontransactional] void
    expTPCCNOInfo([in]  trpc_handle_t      handle,
        [out] dbInfo_data_t *dataP,
        [out] trpc_status_t  * trpcStatus);

[nontransactional] void
    impTPCCNOInfo([out]dbInfo_data_t *dataP,
        [out] trpc_status_t  * trpcStatus);

[nontransactional] void
        expTPCCPayment([in]  trpc_handle_t  handle,
                        [in,out] payment_data_t *dataP,
                        [out] trpc_status_t  * trpcStatus);

[nontransactional] void
        impTPCCPayment([in,out]payment_data_t *dataP,
                        [out] trpc_status_t  * trpcStatus);

[nontransactional] void
    expTPCCPayInfo([in]  trpc_handle_t      handle,
        [in,out] dbInfo_data_t *dataP,
        [out] trpc_status_t  * trpcStatus);


[nontransactional] void
        expTPCCOrderStatus([in]  trpc_handle_t                handle,
                        [in,out] orderStatus_data_t *dataP,
                        [out] trpc_status_t  * trpcStatus);

[nontransactional] void
        impTPCCOrderStatus([in,out]orderStatus_data_t *dataP,
                        [out] trpc_status_t  * trpcStatus);

[nontransactional] void
    expTPCCOSInfo([in]  trpc_handle_t      handle,
        [in,out] dbInfo_data_t *dataP,
        [out] trpc_status_t  * trpcStatus);


[nontransactional] void
        expTPCCStockLevel([in]  trpc_handle_t                handle,
                        [in,out] stockLevel_data_t *dataP,
                        [out] trpc_status_t  * trpcStatus);

[nontransactional] void
        impTPCCStockLevel([in,out] stockLevel_data_t *dataP,
                        [out] trpc_status_t  * trpcStatus);


[nontransactional] void
    expTPCCSLInfo([in]  trpc_handle_t      handle,
        [in,out] dbInfo_data_t *dataP,
        [out] trpc_status_t  * trpcStatus);


}
```

tpcc-type.idl

```
/*
*               tpcc_type.idl
*
* $Revision: 1.11 $
* $Date: 1998/01/24 14:17:07 $
* $Log:           $
*
*
* $TALog: tpcc_type.idl,v $
* Revision 1.11  1998/01/24  14:17:07  oz
* - User server name to identify server and name delivery file
* - Use env variable HOME instead of /home/encina if HOME is set
*
* - Added const ONLINE_INTERFACES
* [from r1.10 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]
*
* Revision 1.10  1998/01/23  15:09:11  oz
* - Updated the SP TPCC directory to the latest files used
*   during the SP tpcc audit.
* [from r1.9 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*
*
*/
[
uuid(008c6338-2b0a-1001-a9ab-02608c2f015a), version(1)
]
interface tpcc_types {

const long NAME_LENGTH = 32;

const long NEWO_INTERFACE =       0x01;
const long PAYMENT_INTERFACE =    0x02;
const long ORDER_STAT_INTERFACE = 0x04;
const long DELIVERY_INTERFACE =   0x08;
const long STOCK_INTERFACE =      0x10;
const long ONLINE_INTERFACES = NEWO_INTERFACE | PAYMENT_INTERFACE |
ORDER_STAT_INTERFACE | STOCK_INTERFACE;
const long ALL_INTERFACE =      0xffff;

const long NEWO_TRANS = 1;
const long PAYMENT_TRANS = 2;
const long ORDER_STAT_TRANS = 3;
const long DELIVERY_TRANS = 4;
const long STOCK_TRANS = 5;
const long MAX_TRAN_TYPE = 5;

typedef struct {
                long int sec;
                long int usec;
} time_type;

typedef struct  {
      short int  dtype;
      short int  returncode;
                long int   sql_code;
                long int   isam_code;
      long int   num_rms;

                time_type  start_time;      /*  For Debug Purposes only  */
                time_type  end_time;        /*  For Debug Purposes only  */
} data_header;

/* Definitions for payment transaction
 *
 *  payment_data_t
 *
 *  An in-out structure for payment transaction.
 *  It contains all the input parameters as well as the output parameters.
 */
typedef struct {
      data_header header;
      short int w_id;
      short int d_id;
      short int c_id;
      short int c_w_id;
      short int c_d_id;
      short int byname;
      double    h_amount;
      char      pay_date[20];

      char      w_name[11];
      char      w_street_1[21];
      char      w_street_2[21];
      char      w_city[21];
      char      w_state[3];
      char      w_zip[10];

      char      d_name[11];
      char      d_street_1[21];
      char      d_street_2[21];
      char      d_city[21];
      char      d_state[3];
      char      d_zip[10];

      char      c_first[17]; /* was C_LAST_LEN already includes +1 */
      char      c_middle[3];
      char      c_last[17];
      char      c_phone[17];
      char      c_credit[3];
      char      c_street_1[21];
      char      c_street_2[21];
      char      c_city[21];
      char      c_state[3];
      char      c_zip[10];
      double    c_credit_lim;
      double    c_balance;
      double    c_discount;
      double    c_ytd_payment;
      short int c_payment_cnt;
      char      c_date[20];
      char      c_data[201];
```

```
} payment_data_t;

/* Definitions for new order transaction */

typedef struct {
        short int  ol_supply_w_id;
        short int  ol_quantity;
        short int  s_quantity;
        long  int  ol_i_id;
        char       name_i[25];
        char       brand_generic[2];
        double     price;
        double     ol_amount;
                   long int   s_idx;
                   char       s_dist[25];
} OL_TABLE, newOrder_item_t;

typedef struct {
                   data_header     header;
                   short int       w_id;
                   short int       d_id;
                   short int       c_id;
                   short int       o_ol_cnt;
                   short int       o_all_local;
                   short int       items_valid; /* true if all valid */
                   short int       total_items;
                   long int        o_id;
        double          w_tax;
                   double          d_tax;
                   double          total;
                   double          c_discount;
                   char            entry_date[20];
                   char            c_last[17];
                   char            c_credit[3];
                   char            statusline[26];
                   OL_TABLE   item[15];
} newOrder_data_t;

/* Definitions for order status transaction */

typedef struct {
  long int  ol_i_id;
  short int  ol_supply_w_id;
  short int  ol_quantity;
  double     ol_amount;
  char       delivery_date[20];
} orderStatusItem_t;

typedef struct {
        data_header header;
        short int w_id;
        short int d_id;
        short int c_id;
        short int o_id;
        short int o_ol_cnt;
        short int byname;
        short     o_carrier_id;
        char      c_last[17];
        char      c_first[17];
        char      c_middle[3];
        char      entry_date[20];
        double    c_balance;
        orderStatusItem_t item[15];
} orderStatus_data_t;

/* Definitions for stock level transaction */

typedef struct {
  data_header header;
  short int w_id;
  short int d_id;
  short int threshold;
  long int  stock_count;
} stockLevel_data_t;

/* Definitions for delivery transaction */

typedef struct {
  data_header header;
  short int w_id;
  short int o_carrier_id;
  long int queued_time;
  short status;
  char exec_status[50];
  double start_queue;
} delivery_data_t;

typedef struct {
  long int first_wh;
  long int last_wh;
  long int server_id;
} dbInfo_data_t;

/*
 *  A union of all the transactions
 */
typedef union switch(long int tran_type) data {
  case NEWO_TRANS:       newOrder_data_t    new_order;
  case PAYMENT_TRANS:                payment_data_t    payment;
  case ORDER_STAT_TRANS: orderStatus_data_t order_status;
  case DELIVERY_TRANS:   delivery_data_t    delivery;
  case STOCK_TRANS:      stockLevel_data_t  stock_level;
} tpcc_data_t;


}
```

tpcc.h

```
#if !defined(TPCC_H_INCLUDED)
#define TPCC_H_INCLUDED
/*************************************************************************/
/*                                                                       */
/*      File: tpcc.h                                                     */
/*      created: 8-26-91                                                 */
/*                                                                       */
/*      program description:                                             */
/*                                                                       */
/*      This module contains global variables and data definitions       */
/*      for the tpcc application.                                        */
/*                                                                       */
/*************************************************************************/

#include "../tpcc_type.h"

#define TPCCH

/*---------------------------------------------------------*/
/*            Global numbers, constants,...                */
/*---------------------------------------------------------*/

#define INVALID_ITEM           100
#define TRAN_OK                0
#define REMOTE_WAREHOUSE       17

#define FORM_DATE              1
#define FORM_DATETIME          2

#define MAX_ITEMS 15

/*---------------------------------------------------------*/
/*            transaction structures                       */
/*---------------------------------------------------------*/

typedef orderStatus_data_t OrderStatus_data;
typedef newOrder_data_t NewOrder_data;
typedef stockLevel_data_t StockLevel_data;
typedef delivery_data_t Delivery_data;
typedef payment_data_t Payment_data;

/*****************************************************************************
  Compatibility for older .sqc files
 *****************************************************************************/
#define s_C_BALANCE c_balance
#define s_C_CITY c_city
#define s_C_CREDIT c_credit
#define s_C_CREDIT_LIM c_credit_lim
#define s_C_DATA c_data
#define s_C_DISCOUNT c_discount
#define s_C_D_ID c_d_id
#define s_C_FIRST c_first
#define s_C_ID c_id
#define s_C_LAST c_last
#define s_C_MIDDLE c_middle
#define s_C_PHONE c_phone
#define s_C_SINCE c_date
#define s_C_STATE c_state
#define s_C_STREET_1 c_street_1
#define s_C_STREET_2 c_street_2
#define s_C_W_ID c_w_id
#define s_C_ZIP c_zip
#define s_D_CITY d_city
#define s_D_ID d_id
#define s_D_STATE d_state
#define s_D_STREET_1 d_street_1
#define s_D_STREET_2 d_street_2
#define s_D_TAX d_tax
#define s_D_ZIP d_zip
#define s_H_AMOUNT h_amount
#define s_H_DATE pay_date
#define s_I_NAME name_i
#define s_I_PRICE price
#define s_OL_AMOUNT ol_amount
#define s_OL_DELIVERY_D delivery_date
#define s_OL_I_ID ol_i_id
#define s_OL_QUANTITY ol_quantity
#define s_OL_SUPPLY_W_ID ol_supply_w_id
#define s_O_CARRIER_ID o_carrier_id
#define s_O_ENTRY_D entry_date
```

```
#define s_O_ID o_id
#define s_O_OL_CNT o_ol_cnt
#define s_S_QUANTITY s_quantity
#define S_QUANTITY quantity
#define s_W_CITY w_city
#define s_W_ID w_id
#define s_W_STATE w_state
#define s_W_STREET_1 w_street_1
#define s_W_STREET_2 w_street_2
#define s_W_TAX w_tax
#define s_W_ZIP w_zip
#define s_all_local o_all_local
#define s_brand_generic brand_generic
#define s_exec_status exec_status
#define s_low_stock stock_count
#define s_ol_cnt o_ol_cnt
#define s_queued_time queued_time
#define s_status_line statusline
#define s_threshold threshold
#define s_total_amount total
#define s_transtatus header.returncode

#if 0
#define NEWORDER_SERVICE    "NEWORD"
#define PAYMENT_SERVICE     "PAYMENT"
#define DELIVERY_SERVICE    "DELIVERY"
#define STOCKLEVEL_SERVICE  "STOCKLEV"
#define ORDERSTATUS_SERVICE "ORDSTAT"
#else
#define NEWORDER_SERVICE    "neword_sql"
#define PAYMENT_SERVICE     "payment_sql"
#define DELIVERY_SERVICE    "delivery_sql"
#define STOCKLEVEL_SERVICE  "stocklev_sql"
#define ORDERSTATUS_SERVICE "ordstat_sql"
#endif

#endif /* TPCC_H_INCLUDED */
```

util_alloc.h

```
/*
 * util_alloc.h
 *
 * $Revision: 1.1 $
 * $Date: 1997/04/20 11:58:08 $
 * $Log:        util_alloc.h,v $
 * Revision 4.2  95/05/16  10:55:43  10:55:43  tpcc (TPCC Benchmark)
 * Added necessary RCS ident strings
 *
 *
 */

#ifndef TRANSARC_UTIL_ALLOC_H
#define TRANSARC_UTIL_ALLOC_H

/*
 * UTIL_[ALLOC, REALLOC, NEW, FREE] -- macros that wrap calls to
 * malloc, realloc, free. The allocation macros check the return
 * value, a NULL pointer is converted into a fatal error.
 */
#define UTIL_ALLOC_ROBUST(ptr, type, size)                              \
   ((ptr) = (type) malloc(size))

#define UTIL_ALLOC(ptr, type, size)                                     \
do {                                                                    \
   if (UTIL_ALLOC_ROBUST(ptr, type, size) == 0)                         \
               util_MemoryError("UTIL_ALLOC", __FILE__, __LINE__);      \
} while (0)

#define UTIL_REALLOC_ROBUST(ptr, type, size)                            \
   (ptr = (type) realloc((void *) ptr, size))

#define UTIL_REALLOC(ptr, type, size)                                                   \
do {                                                                                    \
   if (UTIL_REALLOC_ROBUST(ptr, type, size) == 0)                                       \
               util_MemoryError("UTIL_REALLOC", __FILE__, __LINE__);                    \
} while (0)

#define UTIL_FREE(ptr)                                                                  \
do {                                                                                    \
   if (!ptr) {                                                                          \
               util_MemoryError("UTIL_FREE", __FILE__, __LINE__);                       \
   }                                                                                    \
   free((void *) (ptr));                                                                \
   ptr = 0;     /* Make all free'd pointers zero. */                                    \
} while (0)

#define       UTIL_ALLOC_ARRAY_ROBUST(ptr,type, number)                 \
   ((ptr) = (type *)malloc(sizeof(type) * (number)))

#define       UTIL_ALLOC_ARRAY(ptr,type, number)                                    \
do {                                                                                \
   if (UTIL_ALLOC_ARRAY_ROBUST(ptr,type, number) == 0)     \
               util_MemoryError("UTIL_ALLOC_ARRAY",__FILE__, __LINE__); \
} while (0)

#define       UTIL_COPY_STRING_ROBUST(to,from)                                      \
   (((to) = (char *)malloc(strlen((char *)(from))+1)) ? \
     strcpy((char *)(to), (char *)(from)) : 0)

#define       UTIL_COPY_STRING(to, from)                                            \
do {                                                                                \
   if (UTIL_COPY_STRING_ROBUST(to, from) == 0)                                      \
               util_MemoryError("UTIL_COPY_STRING", __FILE__, __LINE__); \
} while (0)


#endif /* TRANSARC_UTIL_ALLOC_H*/
```

utilities.h

```
/*
 * ID: $Id: utilities.h,v 1.5 1998/01/23 15:09:16 oz Exp $
 *
 * COMPONENT_NAME: Encina Toolkit Server Core
 *
 * ORIGINS: Transarc Corp.
 *
 * (C) COPYRIGHT Transarc Corp. 1995
 * All Rights Reserved
 * Licensed Materials - Property of Transarc
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with Transarc Corp
 * $Revision: 1.5 $
 * $Log:        utilities.h,v $
 *
 *
 * $TALog: utilities.h,v $
 * Revision 1.5  1998/01/23  15:09:16  oz
 * - Updated the SP TPCC directory to the latest files used
 *   during the SP tpcc audit.
 * [from r1.4 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27,r1.1]
 *
 *
 */

/*
 * utilities.h -- holds declarations, macros, and constants used by the
 * telshop/merchandise client-server program.
 *
 * $Date: 1998/01/23 15:09:16 $
 */

#ifndef _UTILITIES_H_
#define _UTILITIES_H_

#include <dce/rpc.h>
#include <dce/dce_error.h>
#include <encina/encina.h>
#include <stdlib.h>

#include <utils/trace.h>
#include "util_alloc.h"

/*  Boolean type, and its constants  */

#define FALSE 0
#define TRUE  1

#if       ENCINA_C_ANSI_STRING_TOKEN_SUPPORT
#define UTIL_STRING(a)         # a
#define UTIL_CONCAT(a, b)       a ## b
#define UTIL_CONCAT3(a,b,c)    a ## b##c
#else   /* ENCINA_C_ANSI_STRING_TOKEN_SUPPORT */
#define UTIL_STRING(a)         "a"
#define UTIL_CONCAT(a, b)       UTIL_IDENT(a)b
#define UTIL_CONCAT3(a,b,c)    UTIL_CONCAT(a,b)c
#endif  /* ENCINA_C_ANSI_STRING_TOKEN_SUPPORT */

/*  ENCINA_CALL:  Make fail-fast calls on the various services. */

/*  Macro delimiters  */

#define BEGIN_MACRO do {

#define END_MACRO } while (0)

/*  FATAL --  Failure.  Print error message and exit the program */

void exit_program();
```

```
#ifndef FATAL
#define FATAL(args)
\
BEGIN_MACRO
                \
   printf args;                                                                    \
   exit(1);
\
END_MACRO
#endif

/*  ENCINA_CALL:  Make fail-fast calls on the various services. */

#define ENCINA_CALL_RC(proc_name,call,rc)                       \
BEGIN_MACRO                                                     \
   char _errorMsg[ENCINA_MAX_STATUS_STRING_SIZE];                                  \
   rc = (call);                                                                    \
   if (rc)  {                                                   \
      encina_StatusToString(rc, ENCINA_MAX_STATUS_STRING_SIZE,          \
                                                _errorMsg);
\
      fprintf(stderr, "%x \n", rc);                                                \
      fprintf(stderr, "%s \n", _errorMsg);                                         \
      fprintf(stderr, "%s \n", proc_name);                                         \
   }
                \
END_MACRO

#define ENCINA_CALL(proc_name,call)                     \
BEGIN_MACRO                                                                     \
   unsigned long _status;                                       \
   ENCINA_CALL_RC(proc_name,call,_status);                      \
   if (_status) exit_program(_status);                          \
END_MACRO

typedef enum {
   action_exit,
   action_continue
} error_action_t;

#define CHECK_DCE_STATUS(_status, _msg, _action)                \
{                                                               \
   int error_stat;                                              \
   unsigned long _rc = (_status);                               \
   unsigned char error_string[dce_c_error_string_len];          \
   if ((_status) != rpc_s_ok) {                                 \
                dce_error_inq_text(_rc, error_string, &error_stat);         \
                err_printf("%s failed, error: %s(%d)\n", _msg, error_string, _rc);\
                if ((_action) == action_exit)                   \
                   exit(-1);                                    \
   }                                                            \
}

#define DCE_CALL(call, args)                            \
{                                                       \
   call args;                                           \
   CHECK_DCE_STATUS(status, UTIL_STRING(call),action_exit);\
}


/*  MALLOC_CHECK --  Make sure there is memory to be allocated;
 *  fail if there is not. */

#define MALLOC_CHECK(memP)                                              \
BEGIN_MACRO                                                                     \
   if (!(memP))                                                                 \
      FATAL(("Out of memory.\n"));                              \
END_MACRO
/*  ASSERT --  internal checks that assure the program is running correctly.
 *  Use to check program correctness, not user input.  */

#ifndef ASSERT
#define ASSERT(condition)                                                       \
BEGIN_MACRO
                \
   if (!(condition))
\
      FATAL(("%s (%d):  Assertion failed.\n", __FILE__, __LINE__));     \
END_MACRO
#endif

#define RAND(lim1, lim2)  ((int)(drand48()*((lim2)-(lim1)+1))+ (lim1))

#ifndef BAD_STATUS
#define BAD_STATUS(call,status)                                 \
   BEGIN_MACRO                                                  \
      char _errorMsg[ENCINA_MAX_STATUS_STRING_SIZE];                    \
      encina_StatusToString(status, ENCINA_MAX_STATUS_STRING_SIZE,      \
                        _errorMsg);                             \
      logprintf("%s: %s (%d)\n", UTIL_STRING(call),_errorMsg, status);  \
      exit(1);                                                  \
   END_MACRO
#endif

#ifndef boolean_t
#define boolean_t int
#endif

#ifndef EXPORT
#define EXPORT
#endif

#ifndef IMPORT
#define IMPORT extern
#endif

#endif /*  _UTILITIES_H_  */
```

Util.s.h

```
/*
 *     do_tpcc.c
 *
 * $Revision: 1.2 $
 * $Date: 1997/07/06 13:43:46 $
 * $Log:         do_tpcc_batch.c,v $
 *
 * $TALog: utils.c,v $
 * Revision 1.2  1997/07/06  13:43:46  oz
 * - Integrate with the IBM RTE: Phase 1:
 *   changes to the idl file
 * - Field changes:
 *     brand_generic is now a 2 character string
 *     new_date -> entry_date
 *     ol_table -> item
 *     Added statusline to newOrder
 *     cur_date -> entry_date
 *     stkl_d_id  -> d_id
 *     carrier_id -> o_carrier_id
 * - The new client performs all the characterIO and screen
 *   manipulations.  There is no need for the curses and liberty
 *   programs.
 * [from r1.1 by delta oz-20306-TPCC-integrate-with-IBM-RTE-clients, r1.1]
 *
 */

#include <sys/time.h>
#include <sys/signal.h>
#include "databuf.h"
/*
 * Make the time stamp from the format: YYYY-MM-DD hh:mm:ss to the format:
 * DD-MM-YYYY hh:mm:ss. Precision is either YEAR_TO_DATE or YEAR_TO_SECOND.
 * If the precision is YEAR_TO_DATE, return a string with the format:
 * "DD-MM-YYYY".
 */
#define Y_POS   0
#define M_POS   5
#define D_POS   8

#define NEW_Y_POS   6
#define NEW_M_POS   3
#define NEW_D_POS   0

extern
void convert_datetime(orig_date, new_date, precision)
char *orig_date;
char *new_date;
short int  precision;
{
   /* Convert only if non-NULL  string */
   if (*orig_date != '\0') {
      strcpy(new_date, orig_date);
      /* make the string DD-MM-YYYY */
      *new_date++ = *(orig_date + D_POS);
      *new_date++ = *(orig_date + (D_POS + 1));
      *new_date++ = '-';
      *new_date++ = *(orig_date + M_POS);
      *new_date++ = *(orig_date + (M_POS + 1));
      *new_date++ = '-';
      *new_date++ = *(orig_date + Y_POS);
      *new_date++ = *(orig_date + (Y_POS + 1));
      *new_date++ = *(orig_date + (Y_POS + 2));
      *new_date++ = *(orig_date + (Y_POS + 3));
      if (precision == YEAR_TO_DATE)
         *new_date = '\0';
   } else {
     *new_date = '\0';
   }
}

extern
double get_time()
{
   struct  timeval now;
   struct  timezone tp;

   gettimeofday(&now, &tp);
   return ( (double) (now.tv_sec) +
     (now.tv_usec/1000000.0)  );
}

#ifdef NULL_TRANS
extern long savemask;
```

```
void sigalarm();
struct  sigvec vec_alarm = {sigalarm,0,0};

#define MASK(s) (1L << ((s) - 1))
void init_ms_sleep()
{
   savemask = sigblock(MASK(SIGALRM));
   if (sigvector(SIGALRM, &vec_alarm, 0) == -1) {
      perror("SIGALRM failed");
      exit(1);
   }
}


void sigalarm()
{
      /* do nothing to the alarm signal, need some work later to catch alarm
        signal and react to it*/
}

/*****************************************************************************
*  ms_sleep  executes setitimer to simulate "sleep", it can sleep fractional
*  seconds
*****************************************************************************/

void ms_sleep(  wait )
double wait;
{
   struct  itimerval value;

   /* it_value = 0.0 disables timer */
   if ( wait > 0.0 ) {
      value.it_value.tv_sec = (int) (wait);
      value.it_value.tv_usec = 1000000 * (wait - value.it_value.tv_sec);
      value.it_interval.tv_sec = 0;
      value.it_interval.tv_usec = 0;
      if (setitimer(ITIMER_REAL,&value,0) == -1) {
         perror("Setitimer failed");
         exit(1);
      }
      sigpause(savemask);
   }
}

#endif
```

A.2 Client Transaction Code

pldel.c

```
#ifdef RCSID
static char *RCSid =
  "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/pldel.c,v 1.2 1998/01/23 15:08:05 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*+=================================================================+
 |       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
 |                  OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                     All Rights Reserved                         |
 +=================================================================+
 | FILENAME
 |    pldel.c
 | DESCRIPTION
 |    OCI version of DELIVERY transaction in TPC-C benchmark.
 +=================================================================*/

#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
#define SQLTXT0 "SELECT substr(value,1,5) FROM v$parameter \
  WHERE name = 'instance_number'"
#endif

#define SQLTXT1 "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 1, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 1 AND o_w_id = :w_id AND o_d_id = 1 AND \
  o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 2, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 2 AND o_w_id = :w_id AND o_d_id = 2 AND \
  o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 3, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 3 AND o_w_id = :w_id AND o_d_id = 3 AND \
  o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 4, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 4 AND o_w_id = :w_id AND o_d_id = 4 AND \
  o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 5, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 5 AND o_w_id = :w_id AND o_d_id = 5 AND \
  o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 6, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 6 AND o_w_id = :w_id AND o_d_id = 6 AND \
  o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 7, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 7 AND o_w_id = :w_id AND o_d_id = 7 AND \
  o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 8, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 8 AND o_w_id = :w_id AND o_d_id = 8 AND \
  o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 9, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 9 AND o_w_id = :w_id AND o_d_id = 9 AND \
  o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 10, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 10 AND o_w_id = :w_id AND o_d_id = 10 AND \
  o_id = no_o_id AND rownum <= 1"

#define SQLTXTTEST "INSERT INTO vv VALUES (:no_rowid, :o_rowid)"
#define SQLTXT2 "DELETE FROM new_order WHERE rowid = :no_rowid"
/*
#define SQLTXT2 "DELETE FROM new_order WHERE no_o_id = :o_id and no_d_id = :d_id
and \
            no_w_id = :w_id"
*/
#define SQLTXT3 "UPDATE orders SET o_carrier_id = :carrier_id \
  WHERE rowid = :o_rowid"

/*
#define SQLTXT3 "UPDATE orders SET o_carrier_id = :carrier_id \
            WHERE o_id = :o_id and o_d_id = :d_id and o_w_id = :w_id \
            and o_c_id = :c_id"

*/
#define SQLTXT4 "UPDATE order_line SET ol_delivery_d = :cr_date \
  WHERE ol_w_id = :w_id AND ol_d_id = :d_id AND ol_o_id = :o_id"

#define SQLTXT5 "\
SELECT :d_id1, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id1 AND ol_o_id = :o_id1 UNION ALL \
SELECT :d_id2, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id2 AND ol_o_id = :o_id2 UNION ALL \
SELECT :d_id3, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id3 AND ol_o_id = :o_id3 UNION ALL \
SELECT :d_id4, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id4 AND ol_o_id = :o_id4 UNION ALL \
SELECT :d_id5, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id5 AND ol_o_id = :o_id5 UNION ALL \
SELECT :d_id6, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id6 AND ol_o_id = :o_id6 UNION ALL \
SELECT :d_id7, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id7 AND ol_o_id = :o_id7 UNION ALL \
SELECT :d_id8, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id8 AND ol_o_id = :o_id8 UNION ALL \
SELECT :d_id9, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id9 AND ol_o_id = :o_id9 UNION ALL \
SELECT :d_id10, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
  ol_d_id = :d_id10 AND ol_o_id = :o_id10"

#define SQLTXT6 "UPDATE customer SET c_balance = c_balance + :amt, \
  c_delivery_cnt = c_delivery_cnt + 1 WHERE c_w_id = :w_id AND \
  c_d_id = :d_id AND c_id = :c_id"

pldelinit (ora_cn_data_t *ora_SlotDataP)

{
  delctx *dctx;
  global_delivery_t *delP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  OCIError *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
```

```
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;

  int i,j;
  char bstr1[10];
  char bstr2[10];
  text stmbuf[4096];

  dctx = (delctx *) malloc (sizeof(delctx));
  memset(dctx,(char)0,sizeof(delctx));
  ora_SlotDataP->dctx = dctx;

  delP = (global_delivery_t *)malloc(sizeof(global_delivery_t));
  memset(delP, (char)0, sizeof(global_delivery_t));
  ora_SlotDataP->delP = delP;

  dctx->norow = 0;
  for(i=0;i<NDISTS;i++) {
/*
    dctx->o_rowid_ptr[i] = &(dctx->o_rowid[i][0]);
    dctx->no_rowid_ptr[i] = &(dctx->no_rowid[i][0]);
*/
   OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&dctx->o_rowid_ptr[i],
         OCI_DTYPE_ROWID,0,(dvoid**)0));
   OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&dctx->no_rowid_ptr[i],
         OCI_DTYPE_ROWID,0,(dvoid**)0));
  }

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd0), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT0);
  OCIStmtPrepare(dctx->curd0, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);

  OCIDFNRA(dctx->curd0, dctx->inum_dp,errhp,1,dctx->inum,SIZ(dctx->inum),SQLT_STR,
       &(dctx->inum_ind),&(dctx->inum_len),&(dctx->inum_rcode));
#endif

  /* open first cursor */

  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd1), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT1);
  OCIStmtPrepare(dctx->curd1, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);

  OCIERROR(errhp,
    OCIAttrSet(dctx->curd1,OCI_HTYPE_STM,(dvoid*)&dctx->norow,0,
          OCI_ATTR_PREFETCH_ROWS,errhp));



  /* bind variables */

  OCIBND(dctx->curd1,
dctx->w_id_bp,errhp,":w_id",ADR(delP->w_id),SIZ(delP->w_id),SQLT_INT);

  OCIDFNRA(dctx->curd1, dctx->d_id_dp,errhp,1,dctx->d_id,SIZ(dctx->d_id[0]),SQLT_INT,
        dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
  OCIDFNRA(dctx->curd1,
dctx->del_o_id_dp,errhp,2,dctx->del_o_id,SIZ(dctx->del_o_id[0]),
        SQLT_INT,dctx->del_o_id_ind,dctx->del_o_id_len,
        dctx->del_o_id_rcode);
  OCIDFNRA(dctx->curd1, dctx->no_rowid_dp,errhp,3,dctx->no_rowid_ptr,
        sizeof(dctx->no_rowid_ptr[0]), SQLT_RDD,dctx->no_rowid_ind,
        dctx->no_rowid_len, dctx->no_rowid_rcode);
  OCIDFNRA(dctx->curd1, dctx->c_id_dp,errhp,4,dctx->c_id,SIZ(dctx->c_id[0]),SQLT_INT,
        dctx->c_id_ind,dctx->c_id_len,dctx->c_id_rcode);
  OCIDFNRA(dctx->curd1, dctx->o_rowid_dp,errhp,5,dctx->o_rowid_ptr,
        sizeof(dctx->o_rowid_ptr[0]), SQLT_RDD,dctx->o_rowid_ind,
        dctx->o_rowid_len, dctx->o_rowid_rcode);

  /* open test cursor */
/*  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curdtest), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXTTEST);
  OCIStmtPrepare(dctx->curdtest, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);


OCIBNDRA(dctx->curdtest,dctx->no_rowid_bp,errhp,":no_rowid",&(dctx->no_rowid_ptr[0])
,
        SIZ(dctx->no_rowid_ptr[0]),SQLT_RDD,dctx->no_rowid_ind,
        dctx->no_rowid_len,dctx->no_rowid_rcode);
  OCIBNDRA(dctx->curdtest,dctx->o_rowid_bp,errhp,":o_rowid",&(dctx->o_rowid_ptr[0]),
        SIZ(dctx->o_rowid_ptr[0]),SQLT_RDD,dctx->o_rowid_ind,
        dctx->o_rowid_len,dctx->o_rowid_rcode);
*/
  /* open second cursor */

  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd2), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT2);
  OCIStmtPrepare(dctx->curd2, errhp, stmbuf, strlen((char *)stmbuf),
          OCI_NTV_SYNTAX, OCI_DEFAULT);

  /* bind variables */

  OCIBNDRA(dctx->curd2,
dctx->no_rowid_bp,errhp,":no_rowid",&(dctx->no_rowid_ptr[0]),
       SIZ(dctx->no_rowid_ptr[0]),SQLT_RDD,dctx->no_rowid_ind,
       dctx->no_rowid_len,dctx->no_rowid_rcode);
/*
  OCIBNDRA(dctx->curd2, dctx->w_id_bp, errhp, ":w_id", dctx->w_id,SIZ(dctx->w_id[0]),
        SQLT_INT,dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
  OCIBNDRA(dctx->curd2, dctx->d_id_bp, errhp, ":d_id", dctx->d_id,SIZ(dctx->d_id[0]),
        SQLT_INT,dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
  OCIBNDRA(dctx->curd2, dctx->del_o_id_bp, errhp, ":o_id", dctx->del_o_id,
        SIZ(dctx->del_o_id[0]),
        SQLT_INT,dctx->del_o_id_ind,dctx->del_o_id_len,dctx->del_o_id_rcode);
*/
  /* open third cursor */

  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd3), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT3);
  OCIStmtPrepare(dctx->curd3, errhp, stmbuf, strlen((char *)stmbuf),
          OCI_NTV_SYNTAX, OCI_DEFAULT);


  /* bind variables */

  OCIBNDRA(dctx->curd3, dctx->carrier_id_bp,errhp,":carrier_id",dctx->carrier_id,
       SIZ(dctx->carrier_id[0]),SQLT_INT,dctx->carrier_id_ind,
       dctx->carrier_id_len,dctx->carrier_id_rcode);
/*
  OCIBNDRA(dctx->curd3, dctx->w_id_bp, errhp, ":w_id", dctx->w_id,SIZ(dctx->w_id[0]),
        SQLT_INT,dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
  OCIBNDRA(dctx->curd3, dctx->d_id_bp, errhp, ":d_id", dctx->d_id,SIZ(dctx->d_id[0]),
        SQLT_INT,dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
  OCIBNDRA(dctx->curd3, dctx->del_o_id_bp, errhp, ":o_id", dctx->del_o_id,
        SIZ(dctx->del_o_id[0]),
        SQLT_INT,dctx->del_o_id_ind,dctx->del_o_id_len,dctx->del_o_id_rcode);
  OCIBNDRA(dctx->curd3, dctx->c_id_bp, errhp, ":c_id", dctx->c_id,SIZ(dctx->c_id[0]),
        SQLT_INT,dctx->c_id_ind,dctx->c_id_len,dctx->c_id_rcode);
*/

  OCIBNDRA(dctx->curd3, dctx->o_rowid_bp,errhp,":o_rowid",&(dctx->o_rowid_ptr[0]),
        SIZ(dctx->o_rowid_ptr[0]),SQLT_RDD,dctx->o_rowid_ind,
        dctx->o_rowid_ptr_len,dctx->o_rowid_rcode);
  /* open fourth cursor */

  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd4), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT4);
  OCIStmtPrepare(dctx->curd4, errhp, stmbuf, strlen((char *)stmbuf),
          OCI_NTV_SYNTAX, OCI_DEFAULT);

  /* bind variables */

  OCIBNDRA(dctx->curd4, dctx->w_id_bp,errhp,":w_id",dctx->w_id,SIZ(dctx->w_id[0]),
        SQLT_INT,dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
  OCIBNDRA(dctx->curd4, dctx->d_id_bp,errhp,":d_id",dctx->d_id,SIZ(dctx->d_id[0]),
        SQLT_INT,dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
  OCIBNDRA(dctx->curd4, dctx->o_id_bp,errhp,":o_id",dctx->del_o_id,
        SIZ(dctx->del_o_id[0]),SQLT_INT,dctx->del_o_id_ind,
        dctx->del_o_id_len,dctx->del_o_id_rcode);
  OCIBNDRA(dctx->curd4, dctx->cr_date_bp,errhp,":cr_date",
dctx->del_date,DEL_DATE_LEN,
        SQLT_DAT,dctx->del_date_ind, dctx->del_date_len,
        dctx->del_date_rcode);


  /* open fifth cursor */

  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd5), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT5);
  OCIStmtPrepare(dctx->curd5, errhp, stmbuf, strlen((char *)stmbuf),
          OCI_NTV_SYNTAX, OCI_DEFAULT);

  OCIERROR(errhp,
    OCIAttrSet(dctx->curd5,OCI_HTYPE_STM,(dvoid*)&dctx->norow,0,
          OCI_ATTR_PREFETCH_ROWS,errhp));

  /* bind variables */


OCIBND(dctx->curd5,dctx->w_id_bp,errhp,":w_id",ADR(delP->w_id),SIZ(delP->w_id),SQLT
_INT);
  for (i = 0; i < NDISTS; i++) {
    sprintf (bstr1, ":d_id%d",i + 1);
    sprintf (bstr2, ":o_id%d",i + 1);
    OCIBNDRA(dctx->curd5,dctx->bstr1_bp[i],errhp,bstr1,ADR(dctx->d_id[i]),
          SIZ(dctx->d_id[0]),SQLT_INT,&(dctx->d_id_ind[i]),
          &(dctx->d_id_len[i]),&(dctx->d_id_rcode[i]));
    OCIBNDRA(dctx->curd5,dctx->bstr2_bp[i],errhp,bstr2,ADR(dctx->del_o_id[i]),
          SIZ(dctx->del_o_id[0]),SQLT_INT,&(dctx->del_o_id_ind[i]),
          &(dctx->del_o_id_len[i]),&(dctx->del_o_id_rcode[i]));
  }

  OCIDFNRA(dctx->curd5,dctx->cons_dp,errhp,1,dctx->cons,SIZ(dctx->cons[0]),SQLT_INT,
       dctx->cons_ind,dctx->cons_len,dctx->cons_rcode);
  OCIDFNRA(dctx->curd5,dctx->amt_dp,errhp,2,dctx->amt,SIZ(dctx->amt[0]),SQLT_INT,
       dctx->amt_ind,dctx->amt_len,dctx->amt_rcode);
```

```
  /* open sixth cursor */

  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd6), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT6);
  OCIStmtPrepare(dctx->curd6, errhp, stmbuf, strlen((char *)stmbuf),
              OCI_NTV_SYNTAX, OCI_DEFAULT);

  /* bind variables */


OCIBNDRA(dctx->curd6,dctx->amt_bp,errhp,":amt",dctx->amt,SIZ(dctx->amt[0]),SQLT_INT
,
       dctx->amt_ind,dctx->amt_len,dctx->amt_rcode);
  OCIBNDRA(dctx->curd6,dctx->w_id_bp,errhp,":w_id",dctx->w_id,SIZ(dctx->w_id[0]),
        SQLT_INT,dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
  OCIBNDRA(dctx->curd6,dctx->d_id_bp,errhp,":d_id",dctx->d_id,SIZ(dctx->d_id[0]),
        SQLT_INT,dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
  OCIBNDRA(dctx->curd6,dctx->c_id_bp,errhp,":c_id",dctx->c_id,SIZ(dctx->c_id[0]),
        SQLT_INT, dctx->c_id_ind,dctx->c_id_len,dctx->c_id_rcode);

  return (0);

}



pldel (ora_cn_data_t *ora_SlotDataP)
{
  delctx *dctx = ora_SlotDataP->dctx;
  global_delivery_t *delP = ora_SlotDataP->delP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  OCIError *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;

  int i, j,v;
  int rpc,rcount;
  int invalid;
  int tmp_id;
  /* float tmp_amt; changed form float to int */
  int tmp_amt;

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  int hasno;
  int reread;
  char sdate[30];

  /*   oexfet (&curd0, 1, 0, 0); */
  OCIStmtExecute(tpcsvc,dctx->curd0,errhp,1,0,0,0,OCI_DEFAULT);
  sysdate (sdate);
  printf ("Delivery started at %s on node %s\n", sdate, dctx->inum);
#endif

retry:

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  reread = 1;
#endif

iso:

  invalid = 0;

  /* initialization for array operations */

  for (i = 0; i < NDISTS; i++) {
    dctx->del_o_id_ind[i] = TRUE;
    dctx->cons_ind[i] = TRUE;
    dctx->w_id_ind[i] = TRUE;
    dctx->d_id_ind[i] = TRUE;
    dctx->c_id_ind[i] = TRUE;
    dctx->del_date_ind[i] = TRUE;
    dctx->carrier_id_ind[i] = TRUE;
    dctx->amt_ind[i] = TRUE;
    dctx->no_rowid_ind[i] = TRUE;
    dctx->o_rowid_ind[i] = TRUE;

    dctx->del_o_id_len[i] = SIZ(dctx->del_o_id[0]);
    dctx->cons_len[i] = SIZ(dctx->cons[0]);
    dctx->w_id_len[i] = SIZ(dctx->w_id[0]);
    dctx->d_id_len[i] = SIZ(dctx->d_id[0]);
    dctx->c_id_len[i] = SIZ(dctx->c_id[0]);
    dctx->del_date_len[i] = DEL_DATE_LEN;
    dctx->carrier_id_len[i] = SIZ(dctx->carrier_id[0]);
    dctx->amt_len[i] = SIZ(dctx->amt[0]);
    dctx->no_rowid_len[i] = ROWIDLEN;
    dctx->o_rowid_len[i] = ROWIDLEN;
    dctx->o_rowid_ptr_len[i] = SIZ(dctx->o_rowid_ptr[0]);
    dctx->no_rowid_ptr_len[i] = SIZ(dctx->no_rowid_ptr[0]);

    dctx->w_id[i] = delP->w_id;
    dctx->carrier_id[i] = delP->o_carrier_id;
    memcpy(dctx->del_date[i],delP->cr_date,DEL_DATE_LEN);
  }

  /* array select from new_order and orders tables */


delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd1,errhp,NDISTS,0,0,0,OCI_DEFAULT)
;
  if((delP->execstatus != OCI_SUCCESS) && (delP->execstatus != OCI_NO_DATA)){
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if(delP->errcode == NOT_SERIALIZABLE){
      delP->retries++;
      goto retry;
    } else if (delP->errcode == RECOVERR) {
      delP->retries++;
      goto retry;
    } else {
      return -1;
    }
  }
  /* mark districts with no new order */
  OCIAttrGet(dctx->curd1,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);
  rpc = rcount;
  invalid = NDISTS - rcount;
  for (i = rpc; i < NDISTS; i++) {
    dctx->del_o_id_ind[i] = NA;
    dctx->w_id_ind[i] = NA;
    dctx->d_id_ind[i] = NA;
    dctx->c_id_ind[i] = NA;
    dctx->carrier_id_ind[i] = NA;
    dctx->no_rowid_ind[i] = NA;
    dctx->o_rowid_ind[i] = NA;
  }

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  if (invalid) {
    sysdate (sdate);
    for (i = 1; i <= NDISTS; i++) {
      hasno = 0;
      for (j = 0; j < rpc; j++) {
        if (dctx->d_id[j] == i) {
          hasno = 1;
          break;
        }
      }
      if (!hasno)
        printf ("Delivery [dist %d] found no new order at %s\n", i, sdate);
    }
    if (reread) {
      sleep (60);
      sysdate (sdate);
      printf ("Delivery wake up at %s\n", sdate);
      reread = 0;
      goto iso;
    }
  }
#endif

  /* array delete of new_order table */
  delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd2,errhp,rpc,0,0,0,OCI_DEFAULT);
  if(delP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if(delP->errcode == NOT_SERIALIZABLE){
      delP->retries++;
      goto retry;
    } else if (delP->errcode == RECOVERR) {
      delP->retries++;
      goto retry;
    } else {
      return -1;
    }
  }

  /* mark districts with no new order */
  OCIAttrGet(dctx->curd2,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);

  if (rcount != rpc) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    return (DEL_ERROR);
  }

  delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd3,errhp,rpc,0,0,0,OCI_DEFAULT);
  if (delP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if(delP->errcode == NOT_SERIALIZABLE){
      delP->retries++;
      goto retry;
    } else if (delP->errcode == RECOVERR) {
      delP->retries++;
      goto retry;
    } else {
      return -1;
    }
```

```
  }

  OCIAttrGet(dctx->curd3,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);

  if (rcount != rpc) {
    fprintf (stderr,
          "Error in TPC-C server %d: %d rows selected, %d rds updated\n",
          proc_no, rpc, rcount);
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    return (-1);
  }

  /* array update of order_line table */
  delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd4,errhp,rpc,0,0,0,OCI_DEFAULT);
  if(delP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if(delP->errcode == NOT_SERIALIZABLE){
      delP->retries++;
      goto retry;
    } else if (delP->errcode == RECOVERR) {
      delP->retries++;
      goto retry;
    } else {
      return -1;
    }
  }

  /* array select from order_line table */
  delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd5,errhp,rpc,0,0,0,OCI_DEFAULT);
  if((delP->execstatus != OCI_SUCCESS) && (delP->execstatus != OCI_NO_DATA)){
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if(delP->errcode == NOT_SERIALIZABLE){
      delP->retries++;
      goto retry;
    } else if (delP->errcode == RECOVERR) {
      delP->retries++;
      goto retry;
    } else {
      return -1;
    }
  }

  OCIAttrGet(dctx->curd5,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);
  if (rcount != rpc) {
    fprintf (stderr,
          "Error in TPC-C server %d: %d rows selected, %d rdl selected\n",
          delP->proc_no, rpc, rcount);
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    return (-1);
  }

  /* reorder amount selected if necessary */

  for (i = 0; i < rpc; i++) {
    if (dctx->cons[i] != dctx->d_id[i]) {
      fprintf (stderr, "TPC-C server %d: reordering amount\n", proc_no);
      for (j = i + 1; j < rpc; j++) {
        if (dctx->cons[j] == dctx->d_id[i]) {
          tmp_id = dctx->cons[i];
          dctx->cons[i] = dctx->cons[j];
          dctx->cons[j] = tmp_id;
          tmp_amt = dctx->amt[i];
          dctx->amt[i] = dctx->amt[j];
          dctx->amt[j] = tmp_amt;
          break;
        }
      }
      if (j >= rpc) {
        fprintf (stderr,
              "Error in TPC-C server %d: missing ordl?\n", delP->proc_no);
        OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
        return (-1);
      }
    }
  }

#if defined(ISO5) || defined(ISO6)
  printf ("d_id:amount\n");
  for (i = 0; i < rpc; i++)
    printf ("%d:%.2f", dctx->d_id[i], (float)dctx->amt[i]/100);
  printf ("\n");
#endif

  /* array update of customer table */
#if defined(ISO5) || defined (ISO6)
  delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,0,0,
                    OCI_DEFAULT);
#else
  delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,0,0,
                     OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
#endif

  if(delP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if(delP->errcode == NOT_SERIALIZABLE){
      delP->retries++;
      goto retry;
    } else if (delP->errcode == RECOVERR) {
      delP->retries++;
      goto retry;
    } else {
      return -1;
    }
  }

  OCIAttrGet(dctx->curd6,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);

  if (rcount != rpc) {
    fprintf (stderr,
          "Error in TPC-C server %d: %d rows selected, %d cust updated\n",
          delP->proc_no, rpc, rcount);
    OCITransRollback(tpcsvc, errhp, OCI_DEFAULT);
    return (-1);
  }

#if defined(ISO5) || defined(ISO6)
  sysdate (sdate);
#ifdef ISO5
  printf ("Delivery sleep before commit at %s\n", sdate);
#else
  printf ("Delivery sleep before abort at %s\n", sdate);
#endif
  sleep (60);
  sysdate (sdate);
  printf ("Delivery wake up at %s\n", sdate);
#endif

#ifdef ISO6
  printf("Delivery ISO6 Rolling back.\n");
  OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
#endif

#ifdef ISO5
  OCITransCommit(tpcsvc, errhp, OCI_DEFAULT);
#endif

#if defined(ISO5) || defined(ISO6)
  sysdate (sdate);
  printf ("Delivery completed at: %s\n", sdate);
#endif

  /* return o_id's in district id order */

  for (i = 0; i < NDISTS; i++)
    delP->del_o_id[i] = 0;
  for (i = 0; i < rpc; i++)
    delP->del_o_id[dctx->d_id[i] - 1] = dctx->del_o_id[i];

  return (0);

}



void pldeldone (ora_cn_data_t *ora_SlotDataP)
{
  delctx *dctx = ora_SlotDataP->dctx;
  global_delivery_t *delP = ora_SlotDataP->delP;

  if (dctx)
  {
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
    OCIHandleFree((dvoid *)dctx->curd0,OCI_HTYPE_STMT);
#endif
    OCIHandleFree((dvoid *)dctx->curd1,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd2,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd3,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd4,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd5,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd6,OCI_HTYPE_STMT);
    free (dctx);
    ora_SlotDataP->dctx = NULL;
  }

  if (delP) {
    free(delP);
    ora_SlotDataP->delP = NULL;
  }
}
```

pldel.h

```
#ifndef TPCC_PLDEL_H
#define TPCC_PLDEL_H
```

```
#define NDISTS 10
#define ROWIDLEN 20


struct delctx {
  sb2 del_o_id_ind[NDISTS];
  sb2 cons_ind[NDISTS];
  sb2 w_id_ind[NDISTS];
  sb2 d_id_ind[NDISTS];
  sb2 c_id_ind[NDISTS];
  sb2 del_date_ind[NDISTS];
  sb2 carrier_id_ind[NDISTS];
  sb2 amt_ind[NDISTS];
  sb2 no_rowid_ind[NDISTS];
  sb2 o_rowid_ind[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  sb2 inum_ind;
#endif

  ub2 del_o_id_len[NDISTS];
  ub2 cons_len[NDISTS];
  ub2 w_id_len[NDISTS];
  ub2 d_id_len[NDISTS];
  ub2 c_id_len[NDISTS];
  ub2 del_date_len[NDISTS];
  ub2 carrier_id_len[NDISTS];
  ub2 amt_len[NDISTS];
  ub2 no_rowid_len[NDISTS];
  ub2 no_rowid_ptr_len[NDISTS];
  ub2 o_rowid_len[NDISTS];
  ub2 o_rowid_ptr_len[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  ub2 inum_len;
#endif

  ub2 del_o_id_rcode[NDISTS];
  ub2 cons_rcode[NDISTS];
  ub2 w_id_rcode[NDISTS];
  ub2 d_id_rcode[NDISTS];
  ub2 c_id_rcode[NDISTS];
  ub2 del_date_rcode[NDISTS];
  ub2 carrier_id_rcode[NDISTS];
  ub2 amt_rcode[NDISTS];
  ub2 no_rowid_rcode[NDISTS];
  ub2 o_rowid_rcode[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  ub2 inum_rcode;
#endif

  int del_o_id[NDISTS];
  int cons[NDISTS];
  int w_id[NDISTS];
  int d_id[NDISTS];
  int c_id[NDISTS];
  int carrier_id[NDISTS];
  /* float amt[NDISTS]; Changed to int */
  int amt[NDISTS];
/*
  OCIRowid *no_rowid[NDISTS];  */
  OCIRowid *no_rowid_ptr[NDISTS];
/*
  OCIRowid *o_rowid[NDISTS];   */
  OCIRowid *o_rowid_ptr[NDISTS];
  unsigned char del_date[NDISTS][DEL_DATE_LEN];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  char inum[10];
#endif
  OCIStmt *curd0;
  OCIStmt *curd1;
  OCIStmt *curd2;
  OCIStmt *curd3;
  OCIStmt *curd4;
  OCIStmt *curd5;
  OCIStmt *curd6;
  OCIStmt *curdtest;

  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *o_id_bp;
  OCIBind *cr_date_bp;
  OCIBind *c_id_bp;
  OCIBind *no_rowid_bp;
  OCIBind *carrier_id_bp;
  OCIBind *o_rowid_bp;
  OCIBind *del_o_id_bp;
  OCIBind *amt_bp;
  OCIBind *bstr1_bp[10];
  OCIBind *bstr2_bp[10];
  OCIDefine *inum_dp;
  OCIDefine *d_id_dp;
  OCIDefine *del_o_id_dp;
  OCIDefine *no_rowid_dp;
  OCIDefine *c_id_dp;
  OCIDefine *o_rowid_dp;
  OCIDefine *cons_dp;
  OCIDefine *amt_dp;

  int norow;
};

typedef struct delctx delctx;

struct global_delivery_t {
  int w_id;
  int o_carrier_id;
  int retries;
  int del_o_id[10];
  int errcode;
  int execstatus;
  int proc_no;
  unsigned char cr_date[7];
};
typedef struct global_delivery_t global_delivery_t;

#endif   /*  TPCCPLDEL_H  */
```

plnew.c

```
#ifdef RCSID
static char *RCSid =
  "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/plnew.c,v 1.2 1998/01/23 15:08:07 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*+=============================================================+
 |       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
 |                OPEN SYSTEMS PERFORMANCE GROUP                |
 |                    All Rights Reserved                       |
 +=============================================================+
 | FILENAME
 |    plnew.c
 | DESCRIPTION
 |    OCI version (using PL/SQL stored procedure) of
 |    NEW ORDER transaction in TPC-C benchmark.
 +=============================================================*/

#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"

extern void err_printf(char *format, ...);

#ifdef OPS
#define SQLTXT2 "UPDATE stock SET s_order_cnt = s_order_cnt + 1, \
   s_ytd = s_ytd + :ol_quantity, s_remote_cnt = s_remote_cnt + :s_remote, \
   s_quantity = s_quantity - :ol_quantity + \
   DECODE (SIGN (s_quantity - :ol_quantity - 10), -1, 91, 0) \
   WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id"
#else
#define SQLTXT2 "UPDATE stock SET s_order_cnt = s_order_cnt + 1, \
   s_ytd = s_ytd + :ol_quantity, s_remote_cnt = s_remote_cnt + :s_remote, \
   s_quantity = :s_quantity \
   WHERE rowid = :s_rowid"
#endif

#define SQLTXT3 "\
SELECT 0,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :10 AND s_w_id = :30 AND s_i_id = i_id UNION ALL \
SELECT 1,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :11 AND s_w_id = :31 AND s_i_id = i_id UNION ALL \
SELECT 2,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :12 AND s_w_id = :32 AND s_i_id = i_id UNION ALL \
SELECT 3,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :13 AND s_w_id = :33 AND s_i_id = i_id UNION ALL \
SELECT 4,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :14 AND s_w_id = :34 AND s_i_id = i_id UNION ALL \
SELECT 5,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :15 AND s_w_id = :35 AND s_i_id = i_id UNION ALL \
SELECT 6,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :16 AND s_w_id = :36 AND s_i_id = i_id UNION ALL \
SELECT 7,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :17 AND s_w_id = :37 AND s_i_id = i_id UNION ALL \
SELECT 8,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :18 AND s_w_id = :38 AND s_i_id = i_id UNION ALL \
SELECT 9,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :19 AND s_w_id = :39 AND s_i_id = i_id UNION ALL \
SELECT 10,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :20 AND s_w_id = :40 AND s_i_id = i_id UNION ALL \
SELECT 11,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :21 AND s_w_id = :41 AND s_i_id = i_id UNION ALL \
SELECT 12,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :22 AND s_w_id = :42 AND s_i_id = i_id UNION ALL \
SELECT 13,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :23 AND s_w_id = :43 AND s_i_id = i_id UNION ALL \
SELECT 14,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :24 AND s_w_id = :44 AND s_i_id = i_id"
```

```
#define SQLTXT4 "INSERT INTO order_line \
  (ol_o_id, ol_d_id, ol_w_id, oL_number, ol_delivery_d, ol_i_id, \
   ol_supply_w_id, ol_quantity,ol_amount,ol_dist_info)\
  VALUES (:ol_o_id, :ol_d_id, \
  :ol_w_id, :ol_number, :null_date, :ol_i_id, :ol_supply_w_id, :ol_quantity, \
  :ol_amount, :ol_dist_info)"

static int SelItemStk (newctx *nctx,
                       global_newOrder_t *newP,
                       OCISvcCtx *tpcsvc,
                       OCIError *errhp);
static int UpdStk2 (newctx *nctx,
                    global_newOrder_t *newP,
                    OCISvcCtx *tpcsvc,
                    OCIError *errhp);


#define ROWIDLEN 20
#define OCIROWLEN 20

/*
 *  plnewinit: called once per thread to initialize the new order
 *  specific thread global data structures.
 */
plnewinit (ora_cn_data_t *ora_SlotDataP)
{

  int i, j;
  char *ora_home = getenv("ORACLE_HOME");
  char sql_file_name[256];

  text stmbuf[SQL_BUF_SIZE];
  char id[4];
  char sd[4];

  newctx *nctx;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  OCIError *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;
  global_newOrder_t *newP;

  ora_SlotDataP->nctx = (newctx *) malloc (sizeof(newctx));
  nctx = ora_SlotDataP->nctx;
  memset(nctx,(char)0,sizeof(newctx));

  ora_SlotDataP->globals = (global_newOrder_t *)malloc(sizeof(global_newOrder_t)
);
  newP = ora_SlotDataP->globals;
  memset(ora_SlotDataP->globals,(char)0,sizeof(global_newOrder_t));

  nctx->cs = 1;
  nctx->norow=0;
  for(i=0;i<NITEMS;i++) {
    OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&nctx->s_rowid_ptr[i],
      OCI_DTYPE_ROWID,0,(dvoid**)0));
  }
  nctx->w_id_ind = TRUE;
  nctx->w_id_len = sizeof(newP->w_id);
  nctx->d_id_ind = TRUE;
  nctx->d_id_len = sizeof(newP->d_id);
  nctx->c_id_ind = TRUE;
  nctx->c_id_len = sizeof(newP->c_id);
  nctx->o_all_local_ind = TRUE;
  nctx->o_all_local_len = sizeof(newP->o_all_local);
  nctx->o_ol_cnt_ind = TRUE;
  nctx->o_ol_cnt_len = sizeof(newP->o_ol_cnt);
  nctx->w_tax_ind = TRUE;
  nctx->w_tax_len = 0;
  nctx->d_tax_ind = TRUE;
  nctx->d_tax_len = 0;
  nctx->o_id_ind = TRUE;
  nctx->o_id_len = sizeof(newP->o_id);
  nctx->c_discount_ind = TRUE;
  nctx->c_discount_len = 0;
  nctx->c_credit_ind = TRUE;
  nctx->c_credit_len = 0;
/*  nctx->c_credit_len = 0; */
  nctx->c_last_ind = TRUE;
  nctx->c_last_len = 0;
  nctx->retries_ind = TRUE;
  nctx->retries_len = sizeof(newP->retries);
  nctx->cr_date_ind = TRUE;
  nctx->cr_date_len = sizeof(newP->cr_date);

  /* open first cursor */
  OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&nctx->curn1),
        OCI_HTYPE_STMT, 0, (dvoid**)0));
  if (!ora_home) {
    err_printf("Cannot find env variable ORACLE_HOME\n");
    exit(13);
  }
  sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/new.sql",ora_home);
    sqlfile(sql_file_name,stmbuf);
  /*    sqlfile("../blocks/new.sql",stmbuf); */

    OCIERROR(errhp,OCIStmtPrepare(nctx->curn1, errhp, stmbuf, strlen((char *)stmbuf),
                          OCI_NTV_SYNTAX, OCI_DEFAULT));

    /* bind variables */

    OCIBNDR(nctx->curn1, nctx->w_id_bp, errhp,
  ":w_id",ADR(newP->w_id),SIZ(newP->w_id),
          SQLT_INT, &nctx->w_id_ind, &nctx->w_id_len, &nctx->w_id_rc);
    OCIBNDR(nctx->curn1, nctx->d_id_bp, errhp, ":d_id",ADR(newP->d_id),SIZ(newP->d_id),
          SQLT_INT, &nctx->d_id_ind, &nctx->d_id_len, &nctx->d_id_rc);
    OCIBNDR(nctx->curn1, nctx->c_id_bp, errhp, ":c_id",ADR(newP->c_id),SIZ(newP->c_id),
          SQLT_INT, &nctx->c_id_ind, &nctx->c_id_len, &nctx->c_id_rc);
    OCIBNDR(nctx->curn1, nctx->o_all_local_bp, errhp, ":o_all_local",
          ADR(newP->o_all_local), SIZ(newP->o_all_local),SQLT_INT, &nctx->o_all_local_ind,
          &nctx->o_all_local_len, &nctx->o_all_local_rc);
    OCIBNDR(nctx->curn1, nctx->o_all_cnt_bp, errhp, ":o_ol_cnt",ADR(newP->o_ol_cnt),
          SIZ(newP->o_ol_cnt),SQLT_INT,
          &nctx->o_ol_cnt_ind, &nctx->o_ol_cnt_len, &nctx->o_ol_cnt_rc);
    OCIBNDR(nctx->curn1, nctx->w_tax_bp, errhp,
  ":w_tax",ADR(newP->w_tax),SIZ(newP->w_tax),
          SQLT_INT, &nctx->w_tax_ind, &nctx->w_tax_len, &nctx->w_tax_rc);
    OCIBNDR(nctx->curn1, nctx->d_tax_bp, errhp,
  ":d_tax",ADR(newP->d_tax),SIZ(newP->d_tax),
          SQLT_INT, &nctx->d_tax_ind, &nctx->d_tax_len, &nctx->d_tax_rc);
    OCIBNDR(nctx->curn1, nctx->o_id_bp, errhp, ":o_id",ADR(newP->o_id),SIZ(newP->o_id),
          SQLT_INT, &nctx->o_id_ind, &nctx->o_id_len, &nctx->o_id_rc);
    OCIBNDR(nctx->curn1, nctx->c_discount_bp, errhp, ":c_discount",
          ADR(newP->c_discount), SIZ(newP->c_discount),SQLT_INT,
          &nctx->c_discount_ind, &nctx->c_discount_len, &nctx->c_discount_rc);
    OCIBNDR(nctx->curn1, nctx->c_credit_bp, errhp, ":c_credit",newP->c_credit,
          SIZ(newP->c_credit),SQLT_CHR,
          &nctx->c_credit_ind, &nctx->c_credit_len, &nctx->c_credit_rc);
    OCIBNDR(nctx->curn1, nctx->c_last_bp, errhp, ":c_last",newP->c_last,SIZ(newP->c_last),
          SQLT_STR, &nctx->c_last_ind, &nctx->c_last_len, &nctx->c_last_rc);
    OCIBNDR(nctx->curn1, nctx->retries_bp, errhp, ":retry",ADR(newP->retries),
          SIZ(newP->retries),SQLT_INT,
          &nctx->retries_ind, &nctx->retries_len, &nctx->retries_rc);
    OCIBNDR(nctx->curn1, nctx->cr_date_bp, errhp,
  ":cr_date",newP->cr_date,SIZ(newP->cr_date),        SQLT_DAT, &nctx->cr_date_ind,
  &nctx->cr_date_len, &nctx->cr_date_rc);

    /* open second cursor */
    OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&nctx->curn2), OCI_HTYPE_STMT,
                        0, (dvoid**)0));
    sprintf ((char *) stmbuf, SQLTXT2);
    OCIERROR(errhp,OCIStmtPrepare(nctx->curn2, errhp, stmbuf,
          strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));



    /* bind variables */

  #ifdef OPS
    OCIBNDRA(nctx->curn2, nctx->ol_i_id_bp,errhp,":ol_i_id",newP->nol_i_id,
          SIZ(int), SQLT_INT, nctx->nol_i_id_ind,nctx->nol_i_id_len,
          nctx->nol_i_id_rcode);
    OCIBNDRA(nctx->curn2, nctx->ol_supply_w_id_bp, errhp, ":ol_supply_w_id",
          newP->nol_supply_w_id,SIZ(int),SQLT_INT,nctx->nol_supply_w_id_ind,
          nctx->nol_supply_w_id_len, nctx->nol_supply_w_id_rcode);
  #else
    OCIBNDRA(nctx->curn2, nctx->s_quantity_bp,errhp,":s_quantity",newP->s_quantity,
          SIZ(int), SQLT_INT,nctx->s_quant_ind,nctx->s_quant_len,
          nctx->s_quant_rcode);
    OCIBNDRA(nctx->curn2, nctx->s_rowid_bp, errhp, ":s_rowid",nctx->s_rowid_ptr,
          sizeof(nctx->s_rowid_ptr[0]),SQLT_RDD,nctx->s_rowid_ind,
          nctx->s_rowid_len,nctx->s_rowid_rcode);
  #endif
    OCIBNDRA(nctx->curn2, nctx->ol_quantity_bp,errhp,":ol_quantity",newP->nol_quantity,
          SIZ(int),SQLT_INT,nctx->nol_quantity_ind,nctx->nol_quantity_len,
          nctx->nol_quantity_rcode);
    OCIBNDRA(nctx->curn2, nctx->s_remote_bp, errhp, ":s_remote",nctx->s_remote,
          SIZ(int), SQLT_INT,nctx->s_remote_ind,nctx->s_remote_len,
          nctx->s_remote_rcode);

    /* open third cursor and bind variables */

    for (i = 0; i < 10; i++)
    {
      j = i + 1;
      OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)&((nctx->curn3)[i]),
                          OCI_HTYPE_STMT, 0, (dvoid**)0));
      sprintf ((char *) stmbuf, SQLTXT3, j, j, j, j, j, j, j, j, j, j,
            j, j, j);
      OCIERROR(errhp,OCIStmtPrepare((nctx->curn3)[i], errhp, stmbuf,
                           strlen((char *)stmbuf),OCI_NTV_SYNTAX,
                           OCI_DEFAULT));
      OCIERROR(errhp,
            OCIAttrSet(nctx->curn3[i],OCI_HTYPE_STM,(dvoid*)&nctx->norow,0,
                OCI_ATTR_PREFETCH_ROWS,errhp));
      for (j = 0; j < NITEMS; j++)
      {
        sprintf (id, ":%d", j + 10);
```

```
    sprintf (sd, ":%d", j + 30);
    OCIBNDRA((nctx->curn3)[i],(nctx->id_bp)[i][j],errhp,id,ADR(newP->nol_i_id[j]),
            SIZ(int),SQLT_INT,
            &nctx->nol_i_id_ind[j],&nctx->nol_i_id_len[j],
            &nctx->nol_i_id_rcode[j]);
    OCIBNDRA((nctx->curn3)[i],(nctx->sd_bp)[i][j],errhp,sd,
            ADR(newP->nol_supply_w_id[j]),SIZ(int),SQLT_INT,
            &nctx->nol_supply_w_id_ind[j],&nctx->nol_supply_w_id_len[j],
            &nctx->nol_supply_w_id_rcode[j]);
    nctx->nol_i_id_ind[j] = NA;
    nctx->nol_supply_w_id_ind[j] = NA;
    nctx->nol_i_id_len[j] = sizeof(int);
    nctx->nol_supply_w_id_len[j] = sizeof(int);
  }

  OCIDFNRA((nctx->curn3)[i],(nctx->Dcons)[i],errhp,1,&(nctx->cons[0]),
          SIZ(nctx->cons[0]),SQLT_INT,
          nctx->cons_ind,nctx->cons_len, nctx->cons_rcode);
  OCIDFNRA((nctx->curn3)[i], (nctx->Ds_rowid)[i],errhp,2,
          nctx->s_rowid_ptr,
          sizeof(nctx->s_rowid_ptr[0]),
          SQLT_RDD,nctx->s_rowid_ind,nctx->s_rowid_len,
          nctx->s_rowid_rcode);
  OCIDFNRA((nctx->curn3)[i], (nctx->Di_price)[i],errhp,3,newP->i_price,SIZ(int),
          SQLT_INT, nctx->i_price_ind,nctx->i_price_len,nctx->i_price_rcode);

  OCIDFNRA((nctx->curn3)[i], (nctx->Di_name)[i],errhp,4,newP->i_name,
          SIZ(newP->i_name[0]),SQLT_STR, nctx->i_name_ind,nctx->i_name_len,
          nctx->i_name_rcode);
  OCIDFNRA((nctx->curn3)[i], (nctx->Di_data)[i],errhp,5,nctx->i_data,
          SIZ(nctx->i_data[0]),
          SQLT_STR,nctx->i_data_ind,nctx->i_data_len,nctx->i_data_rcode);
  OCIDFNRA((nctx->curn3)[i], (nctx->Ds_dist_info)[i],errhp,6,
          nctx->s_dist_info, SIZ(nctx->s_dist_info[0]),SQLT_STR,
          nctx->s_dist_info_ind, nctx->s_dist_info_len,
          nctx->s_dist_info_rcode);
  OCIDFNRA((nctx->curn3)[i],(nctx->Ds_data)[i],errhp,7,nctx->s_data,
          SIZ(nctx->s_data[0]),SQLT_STR,nctx->s_data_ind,
          nctx->s_data_len,nctx->s_data_rcode);
  OCIDFNRA((nctx->curn3)[i],(nctx->Ds_quantity)[i],errhp,8,newP->s_quantity,
          SIZ(int),SQLT_INT, nctx->s_quantity_ind,nctx->s_quantity_len,
          nctx->s_quantity_rcode);

 }

/* open fourth cursor */
OCIHandleAlloc(tpcenv, (dvoid **)(&nctx->curn4), OCI_HTYPE_STMT, 0,
            (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT4);
OCIStmtPrepare(nctx->curn4, errhp, stmbuf, strlen((char *)stmbuf),
            OCI_NTV_SYNTAX, OCI_DEFAULT);
/* bind variables */

OCIBNDRA(nctx->curn4, nctx->ol_o_id_bp,errhp,":ol_o_id",nctx->ol_o_id,
        SIZ(int),SQLT_INT, nctx->ol_o_id_ind,nctx->ol_o_id_len,
        nctx->ol_o_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_d_id_bp,errhp,":ol_d_id",nctx->ol_d_id,
        SIZ(int),SQLT_INT, nctx->ol_d_id_ind,nctx->ol_d_id_len,
        nctx->ol_d_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_w_id_bp,errhp,":ol_w_id",nctx->ol_w_id,
        SIZ(int),SQLT_INT, nctx->ol_w_id_ind,nctx->ol_w_id_len,
        nctx->ol_w_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_number_bp,errhp,":ol_number",nctx->ol_number,
        SIZ(int),SQLT_INT, nctx->ol_number_ind,nctx->ol_number_len,
        nctx->ol_number_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_i_id_bp,errhp,":ol_i_id",newP->nol_i_id,SIZ(int),
        SQLT_INT, nctx->nol_i_id_ind,nctx->nol_i_id_len,
        nctx->nol_i_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_supply_w_id_bp,errhp,":ol_supply_w_id",
        newP->nol_supply_w_id,SIZ(int),SQLT_INT,nctx->nol_supply_w_id_ind,
        nctx->nol_supply_w_id_len, nctx->nol_supply_w_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_quantity_bp,errhp,":ol_quantity",newP->nol_quantity,
        SIZ(int),SQLT_INT, nctx->nol_quantity_ind,nctx->nol_quantity_len,
        nctx->nol_quantity_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_amount_bp,errhp,":ol_amount",newP->nol_amount,
        SIZ(int),SQLT_INT, nctx->nol_amount_ind,nctx->nol_amount_len,
        nctx->nol_amount_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_dist_info_bp,errhp,":ol_dist_info",
        nctx->s_dist_info, SIZ(nctx->s_dist_info[0]),SQLT_AFC,
        nctx->ol_dist_info_ind, nctx->ol_dist_info_len,
        nctx->ol_dist_info_rcode);
OCIBNDRA(nctx->curn4, nctx->null_date_bp,errhp,":null_date",nctx->null_date,
        SIZ(nctx->null_date[0]), SQLT_DAT,nctx->null_date_ind,
        nctx->null_date_len, nctx->null_date_rc);

/* set up the null date Null date is 15-sep-11 */
for (i=0;i<NITEMS;i++)
{
  nctx->null_date[i][0] = 118;
  nctx->null_date[i][1] = 111;
  nctx->null_date[i][2] = 1;
  nctx->null_date[i][3] = 1;
  nctx->null_date[i][4] = 1;
  nctx->null_date[i][5] = 1;
  nctx->null_date[i][6] = 1;
 }

 return (0);

}



plnew (ora_cn_data_t *ora_SlotDataP)
{
  int i, j, k;
  int rpc, rpc3, rowoff, iters,rcount;
  ub4 flags;

  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  OCIError *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;
  global_newOrder_t *newP = ora_SlotDataP->globals;
  newctx *nctx = ora_SlotDataP->nctx;

#if defined(ISO1) || defined(ISO7)
  int reread;
  char sdate[30];

  sysdate (sdate);
  printf ("New Order started at: %s\n", sdate);
#endif

retry:

#ifdef ISO7
  reread = 1;
#endif

  newP->status = 0;                    /* number of invalid items */

  /* get number of order lines, and check if all are local */

  newP->o_ol_cnt = NITEMS;
  newP->o_all_local = 1;
  for (i = 0; i < NITEMS; i++) {
    if (newP->nol_i_id[i] == 0) {
      newP->o_ol_cnt = i;
      break;
    }
    if (newP->nol_supply_w_id[i] != newP->w_id) {
      nctx->s_remote[i] = 1;
      newP->o_all_local = 0;
    }
    else
      nctx->s_remote[i] = 0;
  }
  nctx->w_id_ind = TRUE;
  nctx->w_id_len = sizeof(newP->w_id);
  nctx->d_id_ind = TRUE;
  nctx->d_id_len = sizeof(newP->d_id);
  nctx->c_id_ind = TRUE;
  nctx->c_id_len = sizeof(newP->c_id);
  nctx->o_all_local_ind = TRUE;
  nctx->o_all_local_len = sizeof(newP->o_all_local);
  nctx->o_ol_cnt_ind = TRUE;
  nctx->o_ol_cnt_len = sizeof(newP->o_ol_cnt);
  nctx->w_tax_ind = TRUE;
  nctx->w_tax_len = 0;
  nctx->d_tax_ind = TRUE;
  nctx->d_tax_len = 0;
  nctx->o_id_ind = TRUE;
  nctx->o_id_len = sizeof(newP->o_id);
  nctx->c_discount_ind = TRUE;
  nctx->c_discount_len = 0;
  nctx->c_credit_ind = TRUE;
  nctx->c_credit_len = 0;
  nctx->c_last_ind = TRUE;
  nctx->c_last_len = 0;
  nctx->retries_ind = TRUE;
  nctx->retries_len = sizeof(newP->retries);
  nctx->cr_date_ind = TRUE;
  nctx->cr_date_len = sizeof(newP->cr_date);

  newP->execstatus = OCIStmtExecute(tpcsvc,nctx->curn1,errhp,1,0,0,0,OCI_DEFAULT);
  if(newP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    newP->errcode = OCIERROR(errhp,newP->execstatus);
    if(newP->errcode == NOT_SERIALIZABLE){
      newP->retries++;
      goto retry;
    } else if (newP->errcode == RECOVERR) {
      newP->retries++;
      goto retry;
    } else {
      return -1;
    }
```

```
  }
  /* initialization for array operations */

  for (i = 0; i < newP->o_ol_cnt; i++) {
    nctx->ol_w_id[i] = newP->w_id;
    nctx->ol_d_id[i] = newP->d_id;
    nctx->ol_number[i] = i + 1;
    nctx->null_date_ind[i]= TRUE;
    nctx->nol_i_id_ind[i] = TRUE;
    nctx->nol_supply_w_id_ind[i] = TRUE;
    nctx->nol_quantity_ind[i] = TRUE;
    nctx->nol_amount_ind[i] = TRUE;
    nctx->ol_w_id_ind[i] = TRUE;
    nctx->ol_d_id_ind[i] = TRUE;
    nctx->ol_o_id_ind[i] = TRUE;
    nctx->ol_number_ind[i] = TRUE;
    nctx->ol_dist_info_ind[i] = TRUE;
    nctx->s_remote_ind[i] = TRUE;
    nctx->s_quant_ind[i] = TRUE;
    nctx->cons_ind[i] = TRUE;
    nctx->s_rowid_ind[i] = TRUE;

    nctx->nol_i_id_len[i] = sizeof(int);
    nctx->nol_supply_w_id_len[i] = sizeof(int);
    nctx->nol_quantity_len[i] = sizeof(int);
    nctx->nol_amount_len[i] = sizeof(int);
    nctx->ol_w_id_len[i] = sizeof(int);
    nctx->ol_d_id_len[i] = sizeof(int);
    nctx->ol_o_id_len[i] = sizeof(int);
    nctx->ol_number_len[i] = sizeof(int);
    /* nctx->ol_dist_info_len[i] = sizeof(nctx->s_dist_info[0]);*/
    nctx->ol_dist_info_len[i] = nctx->s_dist_info_len[i];
    nctx->null_date_len[i]=sizeof(nctx->null_date[0]);
    nctx->s_remote_len[i] = sizeof(int);
    nctx->s_quant_len[i] = sizeof(int);
    nctx->s_rowid_len[i] = sizeof(nctx->s_rowid_ptr[0]);
    nctx->cons_len[i] = sizeof(int);
  }
  for (i = newP->o_ol_cnt; i < NITEMS; i++) {
    nctx->nol_i_id_ind[i] = NA;
    nctx->nol_supply_w_id_ind[i] = NA;
    nctx->nol_quantity_ind[i] = NA;
    nctx->nol_amount_ind[i] = NA;
    nctx->ol_w_id_ind[i] = NA;
    nctx->ol_d_id_ind[i] = NA;
    nctx->ol_o_id_ind[i] = NA;
    nctx->ol_number_ind[i] = NA;
    nctx->ol_dist_info_ind[i] = NA;
    nctx->null_date_ind[i]= NA;
    nctx->s_remote_ind[i] = NA;
    nctx->s_quant_ind[i] = NA;
    nctx->cons_ind[i] = NA;
    nctx->s_rowid_ind[i] = NA;

    nctx->nol_i_id_len[i] = 0;
    nctx->nol_supply_w_id_len[i] = 0;
    nctx->nol_quantity_len[i] = 0;
    nctx->nol_amount_len[i] = 0;
    nctx->ol_w_id_len[i] = 0;
    nctx->ol_d_id_len[i] = 0;
    nctx->ol_o_id_len[i] = 0;
    nctx->ol_number_len[i] = 0;
    nctx->ol_dist_info_len[i] = 0;
    nctx->null_date_len[i]= 0;
    nctx->s_remote_len[i] = 0;
    nctx->s_quant_len[i] = 0;
    nctx->s_rowid_len[i] = 0;
    nctx->cons_len[i] = 0;
  }

#ifdef OPS
  rpc = UpdStk ();
  if (rpc == -2)
    goto retry;
  else if (rpc == -1)
    return (-1);
#endif

#ifdef ISO7
iso7:
#endif

  rpc3 = SelItemStk (nctx, newP, tpcsvc, errhp);
  if (rpc3 == -2)
    goto retry;
  else if (rpc3 == -1)
    return (-1);

#ifdef ISO7
  sysdate (sdate);
  printf ("Item table read at: %s\n", sdate);
  for (i = 0; i < newP->o_ol_cnt; i++) {
    if (nctx->nol_i_id_ind[i] != NA)
      printf ("   i_id = %d, i_price = %d\n", newP->nol_i_id[i], newP->i_price[i]);
  }
    if (reread) {
      sleep (30);
      reread = 0;
      status = 0;
      goto iso7;
    }
#endif

    /* compute order line amounts, total amount and stock quantities*/

    newP->total_amount = 0.0;
    for (i = 0; i < newP->o_ol_cnt; i++) {
      nctx->ol_o_id[i] = newP->o_id;
      if (nctx->nol_i_id_ind[i] != NA) {
#ifndef OPS
        newP->s_quantity[i] -= newP->nol_quantity[i];
        if (newP->s_quantity[i] < 10)
          newP->s_quantity[i] += 91;
#endif
        newP->nol_amount[i] = (newP->nol_quantity[i] * newP->i_price[i]);
        newP->total_amount += newP->nol_amount[i];
        if (strstr (nctx->i_data[i], "ORIGINAL") &&
            strstr (nctx->s_data[i], "ORIGINAL"))
          newP->brand_gen[i] = 'B';
        else
          newP->brand_gen[i] = 'G';
      }

    }
    newP->total_amount *= ((float)(10000 - newP->c_discount)/10000) * (1.0 +
((float)(newP->d_tax)/10000) + ((float)(newP->w_tax)/10000));
    newP->total_amount = newP->total_amount/100;

#ifndef OPS
    rpc = UpdStk2 (nctx, newP, tpcsvc, errhp);
    if (rpc == -2)
      goto retry;
    else if (rpc == -1)
      return (-1);
#endif

    /* number of items selected != number of stock updated*/

    if (rpc3 != rpc) {
      fprintf (stderr, "Error in TPC-C server %d: %d rows of item read, ",
               newP->proc_no, rpc3);
      fprintf (stderr, "                but %d rows of stockupdate\n", rpc);
      /* rollback */
      OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      return (-1);
    }

    /* array insert into order line table */
#ifdef ISO1
    flags= OCI_DEFAULT;
#else
    flags= (newP->status ? OCI_DEFAULT : (OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS));
#endif
    if ((newP->o_ol_cnt - newP->status) > 0)
    {
      newP->execstatus = OCIStmtExecute(tpcsvc,nctx->curn4,errhp,newP->o_ol_cnt -
newP->status,
                        0,0,0,flags);
      if(newP->execstatus != OCI_SUCCESS) {
        OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
        newP->errcode = OCIERROR(errhp,newP->execstatus);
        if(newP->errcode == NOT_SERIALIZABLE){
          newP->retries++;
          goto retry;
        } else if (newP->errcode == RECOVERR) {
          newP->retries++;
          goto retry;
        } else {
          return -1;
        }
      }
      OCIAttrGet(nctx->curn4,OCI_HTYPE_STMT,&rcount,NULL,
                 OCI_ATTR_ROWCNT, errhp);
      if (rcount != (newP->o_ol_cnt - newP->status))
      {
       fprintf (stderr, "Error in TPC-C server %d: array insert failed\n",
                newP->proc_no);
       /* rollback */
       OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
       return (-1);
      }
    }

#ifdef ISO1
    sysdate (sdate);
    printf ("Sleep before commit/rollback at: %s\n", sdate);
    sleep (30);
    sysdate (sdate);
    printf ("Wake up after sleep at: %s\n", sdate);
#endif
```

```
  /* commit if no invalid item */

  if (newP->status) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
  }
#ifdef ISO1
  else {
    OCITransCommit(tpcsvc, errhp, OCI_DEFAULT);
  }
#endif

#if defined(ISO1) || defined(ISO7)
  sysdate (sdate);
  printf ("New Order completed at: %s\n", sdate);
#endif

  return (0);

}



void plnewdone (ora_cn_data_t *ora_SlotDataP)

{

  int i;
  newctx *nctx;
  global_newOrder_t *newP;

  if (nctx = ora_SlotDataP->nctx)
  {
    err_printf("free_handles> OCIHandleFree curn s\n");
    OCIHandleFree((dvoid *)nctx->curn1,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)nctx->curn2,OCI_HTYPE_STMT);
    for (i = 0; i < 10; i++)
      OCIHandleFree((dvoid *)(nctx->curn3)[i],OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)nctx->curn4,OCI_HTYPE_STMT);
    err_printf("free_handles> free nctx (0x%x)\n", nctx);
    free (nctx);
    ora_SlotDataP->nctx = NULL;
  }
  if (newP = ora_SlotDataP->globals) {
    err_printf("free_handles> newP: 0x%x\n", newP);
    free(newP);
    ora_SlotDataP->globals = NULL;
  }
}

/* the arrays are initialized based on a successful select from */
/* stock/item. We need to shift the values in the orderline array */
/* one position up to compensate when we have an invalid item */

shiftitemstock (i, j, nctx, newP)

int i, j;
newctx *nctx;
global_newOrder_t *newP;
{

  /* shift up the values for the stock table */
   nctx->s_remote[i] =  nctx->s_remote[j];

  /* shift up the order_line values */

  nctx->nol_i_id_ind[i]=nctx->nol_i_id_ind[j];
  newP->nol_i_id[i] = newP->nol_i_id[j];

  nctx->nol_quantity_ind[i] =  nctx->nol_quantity_ind[j];
  newP->nol_quantity[i] = newP->nol_quantity[j];

  nctx->nol_supply_w_id_ind[i] =  nctx->nol_supply_w_id_ind[j];
  newP->nol_supply_w_id[i] = newP->nol_supply_w_id[j];

}

#if 0

/* TODO - this routine is not ever called. So, no changes for now */

swapitemstock (i, j)

int i, j;

{

  int k;
  int tempi;
  int tempf;
  char tempstr[52];
  ub2 tempub2;
  sb2 tempsb2;
  OCIRowid  *tmprid;

  tempsb2 = nctx->cons_ind[i];
  nctx->cons_ind[i] = nctx->cons_ind[j];
  nctx->cons_ind[j] = tempsb2;
  tempub2 = nctx->cons_len[i];
  nctx->cons_len[i] = nctx->cons_len[j];
  nctx->cons_len[j] = tempub2;
  tempub2 = nctx->cons_rcode[i];
  nctx->cons_rcode[i] = nctx->cons_rcode[j];
  nctx->cons_rcode[j] = tempub2;
  tempi = nctx->cons[i];
  nctx->cons[i] = nctx->cons[j];
  nctx->cons[j] = tempi;

  tempsb2 = nctx->s_rowid_ind[i];
  nctx->s_rowid_ind[i] = nctx->s_rowid_ind[j];
  nctx->s_rowid_ind[j] = tempsb2;
  tempub2 = nctx->s_rowid_len[i];
  nctx->s_rowid_len[i] = nctx->s_rowid_len[j];
  nctx->s_rowid_len[j] = tempub2;
  tempub2 = nctx->s_rowid_rcode[i];
  nctx->s_rowid_rcode[i] = nctx->s_rowid_rcode[j];
  nctx->s_rowid_rcode[j] = tempub2;
  tmprid = nctx->s_rowid_ptr[i];
  nctx->s_rowid_ptr[i]= nctx->s_rowid_ptr[j];
  nctx->s_rowid_ptr[j]=tmprid;

  tempsb2 = nctx->i_price_ind[i];
  nctx->i_price_ind[i] = nctx->i_price_ind[j];
  nctx->i_price_ind[j] = tempsb2;
  tempub2 = nctx->i_price_len[i];
  nctx->i_price_len[i] = nctx->i_price_len[j];
  nctx->i_price_len[j] = tempub2;
  tempub2 = nctx->i_price_rcode[i];
  nctx->i_price_rcode[i] = nctx->i_price_rcode[j];
  nctx->i_price_rcode[j] = tempub2;
  tempf = i_price[i];
  i_price[i] = i_price[j];
  i_price[j] = tempf;

  tempsb2 = nctx->i_name_ind[i];
  nctx->i_name_ind[i] = nctx->i_name_ind[j];
  nctx->i_name_ind[j] = tempsb2;
  tempub2 = nctx->i_name_len[i];
  nctx->i_name_len[i] = nctx->i_name_len[j];
  nctx->i_name_len[j] = tempub2;
  tempub2 = nctx->i_name_rcode[i];
  nctx->i_name_rcode[i] = nctx->i_name_rcode[j];
  nctx->i_name_rcode[j] = tempub2;
  strncpy (tempstr, i_name[i], 25);
  strncpy (i_name[i], i_name[j], 25);
  strncpy (i_name[j], tempstr, 25);

  tempsb2 = nctx->i_data_ind[i];
  nctx->i_data_ind[i] = nctx->i_data_ind[j];
  nctx->i_data_ind[j] = tempsb2;
  tempub2 = nctx->i_data_len[i];
  nctx->i_data_len[i] = nctx->i_data_len[j];
  nctx->i_data_len[j] = tempub2;
  tempub2 = nctx->i_data_rcode[i];
  nctx->i_data_rcode[i] = nctx->i_data_rcode[j];
  nctx->i_data_rcode[j] = tempub2;
  strncpy (tempstr, nctx->i_data[i], 51);
  strncpy (nctx->i_data[i], nctx->i_data[j], 51);
  strncpy (nctx->i_data[j], tempstr, 51);

  tempsb2 = nctx->s_quantity_ind[i];
  nctx->s_quantity_ind[i] = nctx->s_quantity_ind[j];
  nctx->s_quantity_ind[j] = tempsb2;
  tempub2 = nctx->s_quantity_len[i];
  nctx->s_quantity_len[i] = nctx->s_quantity_len[j];
  nctx->s_quantity_len[j] = tempub2;
  tempub2 = nctx->s_quantity_rcode[i];
  nctx->s_quantity_rcode[i] = nctx->s_quantity_rcode[j];
  nctx->s_quantity_rcode[j] = tempub2;
  tempi = s_quantity[i];
  s_quantity[i] = s_quantity[j];
  s_quantity[j] = tempi;

  tempsb2 = nctx->s_dist_info_ind[i];
  nctx->s_dist_info_ind[i] = nctx->s_dist_info_ind[j];
  nctx->s_dist_info_ind[j] = tempsb2;
  tempub2 = nctx->s_dist_info_len[i];
  nctx->s_dist_info_len[i] = nctx->s_dist_info_len[j];
  nctx->s_dist_info_len[j] = tempub2;
  tempub2 = nctx->s_dist_info_rcode[i];
  nctx->s_dist_info_rcode[i] = nctx->s_dist_info_rcode[j];
  nctx->s_dist_info_rcode[j] = tempub2;
  strncpy (tempstr, nctx->s_dist_info[i], 25);
  strncpy (nctx->s_dist_info[i], nctx->s_dist_info[j], 25);
  strncpy (nctx->s_dist_info[j], tempstr, 25);

  tempsb2 = nctx->s_data_ind[i];
  nctx->s_data_ind[i] = nctx->s_data_ind[j];
  nctx->s_data_ind[j] = tempsb2;
  tempub2 = nctx->s_data_len[i];
```

```
  nctx->s_data_len[i] = nctx->s_data_len[j];
  nctx->s_data_len[j] = tempub2;
  tempub2 = nctx->s_data_rcode[i];
  nctx->s_data_rcode[i] = nctx->s_data_rcode[j];
  nctx->s_data_rcode[j] = tempub2;
  strncpy (tempstr, nctx->s_data[i], 51);
  strncpy (nctx->s_data[i], nctx->s_data[j], 51);
  strncpy (nctx->s_data[j], tempstr, 51);

}

#endif /* end of TODO */

static int SelItemStk (nctx, newP, tpcsvc, errhp)
newctx *nctx;
global_newOrder_t *newP;
OCISvcCtx *tpcsvc;
OCIError *errhp;

{

  int i, j, rpc3,rcount;

  /* array select from item and stock tables */

newP->execstatus=OCIStmtExecute(tpcsvc,(nctx->curn3)[newP->d_id-1],errhp,newP->o_ol
_cnt,
                        0,0,0,OCI_DEFAULT);
  if((newP->execstatus != OCI_SUCCESS) && (newP->execstatus != OCI_NO_DATA)){
    newP->errcode = OCIERROR(errhp,newP->execstatus);
    if(newP->errcode == NOT_SERIALIZABLE){
      newP->retries++;
      OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      return (-2);
    } else if (newP->errcode == RECOVERR) {
    /* In case of NO_DATA this should NOT return, but simply fall through*/
      OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      newP->retries++;
      return (-2);
    } else {
      OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      return (-1);
    }
  }
  /* mark invalid items */
  OCIAttrGet((nctx->curn3)[newP->d_id-1],OCI_HTYPE_STMT,&rcount,NULL,
           OCI_ATTR_ROWCNT, errhp);
  rpc3 = rcount;

  /* the result is in order, so we have to shift up to fill    */
  /* the slot for the line with the invalid item.              */
  /* If more than one item is wrong, this is not an simulated */
  /* error and we'll blow off */

  if ((newP->status = newP->o_ol_cnt - rcount) >1)
  {
    fprintf (stderr, "TPC-C server %d: more than 1 invalid item?\n", newP->proc_no);
    return (rpc3);
  }
  if (newP->status == 0) return (rpc3);

  /* find the invalid item, transfer the rowid information */

  for (i = 0; i < newP->o_ol_cnt; i++) {
    if (nctx->cons[i] != i) break;   /* this item is invalid */
  }


/*
          fprintf (stderr, "TPC-C server %d: reordering items and stocks\n",
                newP->proc_no);   */

  /* not the last item - shift up */

  for (j = i; j < newP->o_ol_cnt-1; j++)
  {
    shiftitemstock (j, j+1, nctx, newP);
  }
  /* zero the last item */
    i = newP->o_ol_cnt-1;
    nctx->nol_i_id_ind[i] = NA;
    nctx->nol_supply_w_id_ind[i] = NA;
    nctx->nol_quantity_ind[i] = NA;
    nctx->nol_amount_ind[i] = NA;
    nctx->ol_w_id_ind[i] = NA;
    nctx->ol_d_id_ind[i] = NA;
    nctx->ol_o_id_ind[i] = NA;
    nctx->null_date_ind[i]= NA;
    nctx->ol_number_ind[i] = NA;
    nctx->ol_dist_info_ind[i] = NA;
    nctx->s_remote_ind[i] = NA;
    nctx->s_quant_ind[i] = NA;

    nctx->nol_i_id_len[i] = 0;
    nctx->nol_supply_w_id_len[i] = 0;
    nctx->nol_quantity_len[i] = 0;
    nctx->nol_amount_len[i] = 0;
    nctx->ol_w_id_len[i] = 0;
    nctx->ol_d_id_len[i] = 0;
    nctx->ol_o_id_len[i] = 0;
    nctx->ol_number_len[i] = 0;
    nctx->ol_dist_info_len[i] = 0;
    nctx->null_date_ind[i] = 0;
    nctx->s_remote_len[i] = 0;
    nctx->s_quant_len[i] = 0;

  return (rpc3);

}

/* TODO - no changes yet for OPS code */


#ifdef OPS
UpdStk ()

{
  int rcount;
  /* array update of stock table */

  execstatus = OCIStmtExecute(tpcsvc,nctx->curn2,errhp,o_ol_cnt,
                        0,0,0,OCI_DEFAULT);
  if(execstatus != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    errcode = OCIERROR(errhp,execstatus);
    if(errcode == NOT_SERIALIZABLE){
      retries++;
      return (-2);
    } else if (errcode == RECOVERR) {
      retries++;
      return (-2);
    } else {
      return -1;
    }
  }
  OCIAttrGet(nctx->curn2,OCI_HTYPE_STMT,&rcount,NULL,
           OCI_ATTR_ROWCNT, errhp);
  if (rcount != (o_ol_cnt) ) {
     fprintf (stderr, "Error in TPC-C server %d: array update failed in UpdStk()\n",
           proc_no);
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     return (-1);
  }
  return (rcount);
}
#endif



#ifndef OPS
static int UpdStk2 (nctx, newP, tpcsvc, errhp)
newctx *nctx;
global_newOrder_t *newP;
OCISvcCtx *tpcsvc;
OCIError *errhp;

{

  int rpc, rowoff, iters,rcount;

  /* array update of stock table */

  newP->execstatus =
OCIStmtExecute(tpcsvc,nctx->curn2,errhp,newP->o_ol_cnt-newP->status,0,0,0,
                        OCI_DEFAULT);
  if(newP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    newP->errcode = OCIERROR(errhp,newP->execstatus);
    if(newP->errcode == NOT_SERIALIZABLE){
      newP->retries++;
      return (-2);
    } else if (newP->errcode == RECOVERR) {
      newP->retries++;
      return (-2);
    } else {
      return -1;
    }
  }
  OCIAttrGet(nctx->curn2,OCI_HTYPE_STMT,&rcount,NULL,OCI_ATTR_ROWCNT, errhp);
  rpc = rcount;

  if (rpc != (newP->o_ol_cnt - newP->status)) {
    fprintf (stderr, "Error in TPC-C server %d: array update failed\n",
          newP->proc_no);
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    return (-1);
  }

  return (rpc);
```

```
}
#endif
```

plnew.h

```
#ifndef TPCC_PLNEW_H
#define TPCC_PLNEW_H
#include <oci.h>

struct newctx {
  sb2 nol_i_id_ind[NITEMS];
  sb2 nol_supply_w_id_ind[NITEMS];
  sb2 nol_quantity_ind[NITEMS];
  sb2 nol_amount_ind[NITEMS];
  sb2 i_name_ind[NITEMS];
  sb2 s_quantity_ind[NITEMS];
  sb2 i_price_ind[NITEMS];
  sb2 ol_w_id_ind[NITEMS];
  sb2 ol_d_id_ind[NITEMS];
  sb2 ol_o_id_ind[NITEMS];
  sb2 ol_number_ind[NITEMS];
  sb2 cons_ind[NITEMS];
  sb2 s_rowid_ind[NITEMS];
  sb2 s_remote_ind[NITEMS];
  sb2 s_quant_ind[NITEMS];
  sb2 i_data_ind[NITEMS];
  sb2 s_data_ind[NITEMS];
  sb2 s_dist_info_ind[NITEMS];
  sb2 ol_dist_info_ind[NITEMS];
  sb2 null_date_ind[NITEMS];

  ub2 nol_i_id_len[NITEMS];
  ub2 nol_supply_w_id_len[NITEMS];
  ub2 nol_quantity_len[NITEMS];
  ub2 nol_amount_len[NITEMS];
  ub2 i_name_len[NITEMS];
  ub2 s_quantity_len[NITEMS];
  ub2 i_price_len[NITEMS];
  ub2 ol_w_id_len[NITEMS];
  ub2 ol_d_id_len[NITEMS];
  ub2 ol_o_id_len[NITEMS];
  ub2 ol_number_len[NITEMS];
  ub2 cons_len[NITEMS];
  ub2 s_rowid_len[NITEMS];
  ub2 s_remote_len[NITEMS];
  ub2 s_quant_len[NITEMS];
  ub2 i_data_len[NITEMS];
  ub2 s_data_len[NITEMS];
  ub2 s_dist_info_len[NITEMS];
  ub2 ol_dist_info_len[NITEMS];
  ub2 null_date_len[NITEMS];

  ub2 nol_i_id_rcode[NITEMS];
  ub2 nol_supply_w_id_rcode[NITEMS];
  ub2 nol_quantity_rcode[NITEMS];
  ub2 nol_amount_rcode[NITEMS];
  ub2 i_name_rcode[NITEMS];
  ub2 s_quantity_rcode[NITEMS];
  ub2 i_price_rcode[NITEMS];
  ub2 ol_w_id_rcode[NITEMS];
  ub2 ol_d_id_rcode[NITEMS];
  ub2 ol_o_id_rcode[NITEMS];
  ub2 ol_number_rcode[NITEMS];
  ub2 cons_rcode[NITEMS];
  ub2 s_rowid_rcode[NITEMS];
  ub2 s_remote_rcode[NITEMS];
  ub2 s_quant_rcode[NITEMS];
  ub2 i_data_rcode[NITEMS];
  ub2 s_data_rcode[NITEMS];
  ub2 s_dist_info_rcode[NITEMS];
  ub2 ol_dist_info_rcode[NITEMS];
  ub2 null_date_rc[NITEMS];



  int ol_w_id[NITEMS];
  int ol_d_id[NITEMS];
  int ol_o_id[NITEMS];
  int ol_number[NITEMS];
  int cons[NITEMS];

  OCIRowid  *s_rowid_ptr[NITEMS];

  int s_remote[NITEMS];
  char i_data[NITEMS][51];
  char s_data[NITEMS][51];
  char s_dist_info[NITEMS][25];
  unsigned char null_date[NITEMS][7]; /* base date for null date entry */
  OCIStmt *curn;
  OCIStmt *curn1;
  OCIStmt *curn2;
  OCIStmt *curn3[10];
  OCIStmt *curn4;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *c_id_bp;
  OCIBind *o_all_local_bp;
  OCIBind *o_all_cnt_bp;
  OCIBind *w_tax_bp;
  OCIBind *d_tax_bp;
  OCIBind *o_id_bp;
  OCIBind *c_discount_bp;
  OCIBind *c_credit_bp;
  OCIBind *c_last_bp;
  OCIBind *retries_bp;
  OCIBind *cr_date_bp;
  OCIBind *ol_i_id_bp;
  OCIBind *ol_supply_w_id_bp;
  OCIBind *s_quantity_bp;
  OCIBind *s_rowid_bp;
  OCIBind *ol_quantity_bp;
  OCIBind *s_remote_bp;
  OCIBind *id_bp[10][15];
  OCIBind *sd_bp[10][15];
  OCIDefine *Dcons[10];
  OCIDefine *Ds_rowid[10];
  OCIDefine *Di_price[10];
  OCIDefine *Di_data[10];
  OCIDefine *Ds_dist_info[10];
  OCIDefine *Ds_data[10];
  OCIDefine *Ds_quantity[10];
  OCIDefine *Di_name[10];
  OCIBind *ol_o_id_bp;
  OCIBind *ol_d_id_bp;
  OCIBind *ol_w_id_bp;
  OCIBind *ol_number_bp;
  OCIBind *ol_amount_bp;
  OCIBind *ol_dist_info_bp;
  OCIBind *null_date_bp;
  sb2 w_id_ind;
  ub2 w_id_len;
  ub2 w_id_rc;

  sb2 d_id_ind;
  ub2 d_id_len;
  ub2 d_id_rc;

  sb2 c_id_ind;
  ub2 c_id_len;
  ub2 c_id_rc;

  sb2 o_all_local_ind;
  ub2 o_all_local_len;
  ub2 o_all_local_rc;

  sb2 o_ol_cnt_ind;
  ub2 o_ol_cnt_len;
  ub2 o_ol_cnt_rc;

  sb2 w_tax_ind;
  ub2 w_tax_len;
  ub2 w_tax_rc;

  sb2 d_tax_ind;
  ub2 d_tax_len;
  ub2 d_tax_rc;

  sb2 o_id_ind;
  ub2 o_id_len;
  ub2 o_id_rc;

  sb2 c_discount_ind;
  ub2 c_discount_len;
  ub2 c_discount_rc;

  sb2 c_credit_ind;
  ub2 c_credit_len;
  ub2 c_credit_rc;

  sb2 c_last_ind;
  ub2 c_last_len;
  ub2 c_last_rc;

  sb2 retries_ind;
  ub2 retries_len;
  ub2 retries_rc;

  sb2 cr_date_ind;
  ub2 cr_date_len;
  ub2 cr_date_rc;


  int cs;
  int norow;
};
```

```
typedef struct newctx newctx;


/* struct to copy-in/out neworder vars */
struct global_newOrder_t {
  int w_id;
  int d_id;
  int c_id;
  int nol_i_id[15];
  int nol_supply_w_id[15];
  int nol_quantity[15];
  int retries;
  char o_entry_d[20];
  int o_id;
  int o_ol_cnt;
  char c_last[17];
  char c_credit[3];
  int c_discount;
  int w_tax;
  int d_tax;
  float total_amount;
  char i_name[15][25];
  int s_quantity[15];
  char brand_gen[15];
  int i_price[15];
  int nol_amount[15];
  int status;
  int o_all_local;
  int errcode;
  int execstatus;
  int proc_no;
  unsigned char cr_date[7];
};

typedef struct global_newOrder_t global_newOrder_t;

#endif  /* TPCC_PLNEW_H  */
```

plora.h

```
#ifndef TPCC_PLORA_H
#define TPCC_PLORA_H

#include <dce/pthread.h>
#include "tpcc.h"
#include "tpccpl.h"
#include "plnew.h"
#include "plpay.h"
#include "plord.h"
#include "plsto.h"
#include "pldel.h"
#include "tpcc_info.h"

#define MAX_ORA_CONNECTIONS 100
#define NEWO_TRANS (1)
#define PAYMENT_TRANS (2)
#define ORDER_STAT_TRANS (3)
#define DELIVERY_TRANS (4)
#define STOCK_TRANS (5)
#define DIST_NEWO_TRANS (6)
#define DIST_PAY_TRANS (7)
#define MAX_TRAN_TYPE (7)

#define NUM_STATES            4
#define SVR_STATE_NONE        0
#define SVR_STATE_SENT        1
#define SVR_STATE_REPLIED     2
#define SVR_STATE_ERR         3

typedef struct {
   int num;
   int errs;
   double RT;
} tran_info_t;
/*
 * total_tran_count_t
 *
 * structure that holds the total count of transaction of each type
 * as well as the reposne times.
 *
 */
typedef struct {
   tran_info_t tran[MAX_TRAN_TYPE + 1];
   double dvry_queue_time;  /*  The average time a
                            *  delivery request was in the queue
                            */
   int errors;
} total_tran_count_t;

typedef struct {
   int tran_type;
   union {
      struct newstruct no;
      struct paystruct pa;
      struct ordstruct os;
      struct delstruct dl;
      struct stostruct sl;
   } tran_info;
} tran_spec_info_t;

/* Oracle handles and rest of thread specific vars(thread slot data ) */

struct ora_cn_data_t {
  OCIEnv *tpcenv;
  OCIServer *tpcsrv;
  OCIError *errhp;
  OCISvcCtx *tpcsvc;
  OCISession *tpcusr;
  OCIStmt *curi;
  dvoid *xmem;

  global_newOrder_t *globals;
  global_payment_t *payP;
  global_order_t *ordP;
  global_delivery_t *delP;
  global_stock_t *stoP;
  newctx *nctx;
  payctx *pctx;
  ordctx *octx;
  stoctx *sctx;
  delctx *dctx;

  int calls;    /*  Number of times it was used */
  int errors;   /*  Total number of errors on this connection */
  int calls_last_err;   /*  Number of calls when the last error occured  */
  int consecutive_errs; /*  Number of consecutive errs  */
  int connect_time;     /*  Time (seconds) connections was created */

  /*  For debug  */
  int state;                  /*  State of the connection  */
  struct timeval tran_time;     /*  Time this tran started */
  int cur_tran_type;
  void *cur_tran_dataP;
  total_tran_count_t stat;
  int printed;
};

typedef struct ora_cn_data_t ora_cn_data_t;

extern int plnewinit (ora_cn_data_t *);
extern int plpayinit (ora_cn_data_t *);
extern int plordinit (ora_cn_data_t *);
extern int pldelinit (ora_cn_data_t *);
extern int plstoinit (ora_cn_data_t *);

extern int plnew (ora_cn_data_t *);
extern int plpay (ora_cn_data_t *);
extern int plord (ora_cn_data_t *);
extern int pldel (ora_cn_data_t *);
extern int plsto (ora_cn_data_t *);

extern void plnewdone (ora_cn_data_t *);
extern void plpaydone (ora_cn_data_t *);
extern void plorddone (ora_cn_data_t *);
extern void pldeldone (ora_cn_data_t *);
extern void plstodone (ora_cn_data_t *);



#endif /* TPCC_PLORA_H */
```

plord.c

```
#ifdef RCSID
static char *RCSid =
  "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RCS/plord.c,v 1.2 1998/01/23 15:08:13 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*+=================================================================+
 |      Copyright (c) 1995  Oracle Corp, Redwood Shores, CA        |
 |                OPEN SYSTEMS PERFORMANCE GROUP                   |
 |                   All Rights Reserved                           |
 +=================================================================+
 | FILENAME
 |    plord.c
 | DESCRIPTION
 |    OCI version (using PL/SQL anynomous block) of
 |    ORDER STATUS transaction in TPC-C benchmark.
 +=================================================================*/

#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"
```

```
#ifdef ISO8
#define SQLTXT "BEGIN aorderstatus.agetstatus (:w_id, :d_id, :c_id, :byln, \
  :c_last, :c_first, :c_middle, :c_balance, :o_id, :o_entry_d, :o_cr_id, \
  :o_ol_cnt, :ol_s_w_id, :ol_i_id, :ol_quantity, :ol_amount, :ol_d_d); END;"
#endif

#define SQLCUR0 "SELECT rowid FROM customer \
             WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last = :c_last \
             ORDER BY c_w_id, c_d_id, c_last, c_first"

#define SQLCUR1 "SELECT c_id, c_balance, c_first, c_middle, \
             o_id, o_entry_d, o_carrier_id, o_ol_cnt, c_last \
             FROM customer, orders \
             WHERE customer.rowid = :cust_rowid \
             AND   o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id = c_id \
             ORDER BY o_w_id, o_d_id, o_c_id, o_id DESC"


#define SQLCUR2 "SELECT c_balance, c_first, c_middle, c_last, \
             o_id, o_entry_d, o_carrier_id, o_ol_cnt \
             FROM customer, orders \
             WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id \
             AND o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id = c_id \
             ORDER BY o_w_id, o_d_id, o_c_id, o_id DESC"


#define SQLCUR3 "SELECT ol_i_id,ol_supply_w_id,ol_quantity,ol_amount,\
             ol_delivery_d\
             FROM order_line \
             WHERE ol_d_id = :d_id AND ol_w_id = :w_id AND ol_o_id = :o_id"


plordinit (ora_cn_data_t *ora_SlotDataP)

{

  int i;
  text stmbuf[SQL_BUF_SIZE];

  ordctx *octx;
  global_order_t *ordP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  OCIError *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;

  octx = (ordctx *) malloc (sizeof(ordctx));
  memset(octx,(char)0,sizeof(ordctx));
  ora_SlotDataP->octx = octx;

  ora_SlotDataP->ordP = (global_order_t *)malloc(sizeof(global_order_t));
  memset(ora_SlotDataP->ordP,(char)0,sizeof(global_order_t));
  ordP = ora_SlotDataP->ordP;

  octx->cs = 1;
  octx->norow = 0;
/* get the rowid handles */
  for(i=0;i<3000;i++) {
   OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&octx->c_rowid_ptr[i],
      OCI_DTYPE_ROWID,0,(dvoid**)0));
  }


#ifdef ISO8
  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid **)&octx->curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
#else
  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo1,OCI_HTYPE_STMT,0,(dvoid**)0));
  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo2,OCI_HTYPE_STMT,0,(dvoid**)0));
  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo3,OCI_HTYPE_STMT,0,(dvoid**)0));
#endif

#ifdef ISO8
  sprintf((char *) stmbuf, SQLTXT);
  OCIERROR(errhp,
    OCIStmtPrepare(octx->curo0,errhp,stmbuf,strlen((char*)stmbuf),
            OCI_NTV_SYNTAX,OCI_DEFAULT));
#else
/*  c_id = 0, use find customer by lastname. Get an array or rowid's back*/
  sprintf((char *) stmbuf, SQLCUR0);
  OCIERROR(errhp,
    OCIStmtPrepare(octx->curo0,errhp,stmbuf,strlen((char*)stmbuf),
            OCI_NTV_SYNTAX,OCI_DEFAULT));
  OCIERROR(errhp,
    OCIAttrSet(octx->curo0,OCI_HTYPE_STM,(dvoid*)&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));
/* get order/customer info back based on rowid */
  sprintf((char *) stmbuf, SQLCUR1);
  OCIERROR(errhp,
    OCIStmtPrepare(octx->curo1,errhp,stmbuf,strlen((char*)stmbuf),
            OCI_NTV_SYNTAX,OCI_DEFAULT));
  OCIERROR(errhp,
    OCIAttrSet(octx->curo1,OCI_HTYPE_STM,(dvoid*)&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));

/*  c_id == 0, use lastname to find customer */
  sprintf((char *) stmbuf, SQLCUR2);
  OCIERROR(errhp,
    OCIStmtPrepare(octx->curo2,errhp,stmbuf,strlen((char*)stmbuf),
            OCI_NTV_SYNTAX,OCI_DEFAULT));
  OCIERROR(errhp,
    OCIAttrSet(octx->curo2,OCI_HTYPE_STM,(dvoid*)&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));

  sprintf((char *) stmbuf, SQLCUR3);
  OCIERROR(errhp,
    OCIStmtPrepare(octx->curo3,errhp,stmbuf,strlen((char*)stmbuf),
            OCI_NTV_SYNTAX,OCI_DEFAULT));
  OCIERROR(errhp,
    OCIAttrSet(octx->curo3,OCI_HTYPE_STM,(dvoid*)&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));
#endif

  for (i = 0; i < NITEMS; i++) {
    octx->ol_supply_w_id_ind[i] = TRUE;
    octx->ol_i_id_ind[i] = TRUE;
    octx->ol_quantity_ind[i] = TRUE;
    octx->ol_amount_ind[i] = TRUE;
    octx->ol_delivery_d_ind[i] = TRUE;

    octx->ol_supply_w_id_len[i] = sizeof(int);
    octx->ol_i_id_len[i] = sizeof(int);
    octx->ol_quantity_len[i] = sizeof(int);
    octx->ol_amount_len[i] = sizeof(int);
    octx->ol_delivery_d_len[i] = sizeof(ordP->ol_d_base[0]);
  }
  octx->ol_supply_w_id_csize = NITEMS;
  octx->ol_i_id_csize = NITEMS;
  octx->ol_quantity_csize = NITEMS;
  octx->ol_amount_csize = NITEMS;
  octx->ol_delivery_d_csize = NITEMS;
  octx->ol_w_id_csize = NITEMS;
  octx->ol_o_id_csize = NITEMS;
  octx->ol_d_id_csize = NITEMS;
  octx->ol_w_id_ind = TRUE;
  octx->ol_d_id_ind = TRUE;
  octx->ol_o_id_ind = TRUE;
  octx->ol_w_id_len = sizeof(int);
  octx->ol_d_id_len = sizeof(int);
  octx->ol_o_id_len = sizeof(int);

  /* bind variables */
#ifdef ISO8

  OCIBND(octx->curo0,octx->w_id_bp0,errhp,":w_id",ADR(ordP->w_id),SIZ(ordP->w_id),
            SQLT_INT);
  OCIBND(octx->curo0, octx->d_id_bp0, errhp,":d_id",ADR(ordP->d_id),SIZ(ordP->d_id),
            SQLT_INT);
  OCIBND(octx->curo0, octx->c_id_bp, errhp,":c_id",ADR(ordP->c_id),SIZ(ordP->c_id),
            SQLT_INT);
  OCIBND(octx->curo0, octx->byln_bp, errhp,":byln",ADR(ordP->bylastname),
            SIZ(ordP->bylastname),SQLT_INT);
  OCIBND(octx->curo0, octx->c_last_bp, errhp,":c_last",ordP->c_last,
            SIZ(ordP->c_last),SQLT_STR);
  OCIBND(octx->curo0, octx->c_first_bp, errhp,":c_first",ordP->c_first,
            SIZ(ordP->c_first),SQLT_STR);
  OCIBND(octx->curo0, octx->c_middle_bp, errhp,":c_middle",ordP->c_middle,
            SIZ(ordP->c_middle),SQLT_STR);
  OCIBND(octx->curo0, octx->c_balance_bp, errhp,":c_balance",ADR(ordP->c_balance),
            SIZ(ordP->c_balance),SQLT_FLT);
  OCIBND(octx->curo0, octx->o_id_bp, errhp,":o_id",ADR(ordP->o_id),
            SIZ(ordP->o_id),SQLT_INT);
  OCIBND(octx->curo0, octx->o_entry_d_bp, errhp,":o_entry_d",
            ordP->o_entry_d_base,SIZ(ordP->o_entry_d_base),SQLT_DAT);
  OCIBND(octx->curo0, octx->o_cr_id_bp, errhp,":o_cr_id",ADR(ordP->o_carrier_id),
            SIZ(ordP->o_carrier_id), SQLT_INT);
  OCIBND(octx->curo0, octx->o_ol_cnt_bp, errhp,":o_ol_cnt",ADR(ordP->o_ol_cnt),
            SIZ(ordP->o_ol_cnt),SQLT_INT);
  OCIBNDRAA(octx->curo0, octx->ol_s_w_id_bp, errhp,":ol_s_w_id",
            ordP->ol_supply_w_id,SIZ(int),SQLT_INT,
            octx->ol_supply_w_id_ind,octx->ol_supply_w_id_len,
            octx->ol_supply_w_id_rcode,NITEMS,
            ADR(octx->ol_supply_w_id_csize));
  OCIBNDRAA(octx->curo0, octx->ol_i_id_bp, errhp,":ol_i_id",ordP->ol_i_id,
            SIZ(int),SQLT_INT,
            octx->ol_i_id_ind,octx->ol_i_id_len,octx->ol_i_id_rcode,NITEMS,
            ADR(octx->ol_i_id_csize));
  OCIBNDRAA(octx->curo0, octx->ol_quantity_bp, errhp,":ol_quantity",
            ordP->ol_quantity,SIZ(int),SQLT_INT,
            octx->ol_quantity_ind,octx->ol_quantity_len,
            octx->ol_quantity_rcode, NITEMS,ADR(octx->ol_quantity_csize));
```

```
  OCIBNDRAA(octx->curo0, octx->ol_amount_bp, errhp,":ol_amount",ordP->ol_amount,
            SIZ(float),SQLT_FLT,
            octx->ol_amount_ind,octx->ol_amount_len,octx->ol_amount_rcode,
            NITEMS,ADR(octx->ol_amount_csize));
  OCIBNDRAA(octx->curo0, octx->ol_d_d_bp, errhp,":ol_d_d",ordP->ol_delivery_d,
            SIZ(ordP->ol_delivery_d[0]),SQLT_STR,
            octx->ol_delivery_d_ind,octx->ol_delivery_d_len,
            octx->ol_delivery_d_rcode,NITEMS,
            ADR(octx->ol_delivery_d_csize));
#else

  /* c_id (customer id) is not known */

OCIBND(octx->curo0,octx->w_id_bp0,errhp,":w_id",ADR(ordP->w_id),SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->d_id_bp0,errhp,":d_id",ADR(ordP->d_id),SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->c_last_bp,errhp,":c_last",ordP->c_last,SIZ(ordP->c_last),
       SQLT_STR);
  OCIDFNRA(octx->curo0,octx->c_rowid_dp,errhp,1,octx->c_rowid_ptr,
        sizeof(octx->c_rowid_ptr[0]),SQLT_RDD,octx->c_rowid_ind,
        octx->c_rowid_len,octx->c_rowid_rcode);

  OCIBND(octx->curo1,octx->c_rowid_bp,errhp,":cust_rowid",
       &octx->c_rowid_ptr[octx->cust_idx],
       sizeof( octx->c_rowid_ptr[0]),SQLT_RDD);
  OCIDEF(octx->curo1,octx->c_id_dp,errhp,1,ADR(ordP->c_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->c_balance_dp1,errhp,2,ADR(ordP->c_balance),
       SIZ(double),SQLT_FLT);
  OCIDEF(octx->curo1,octx->c_first_dp1,errhp,3,ordP->c_first,SIZ(ordP->c_first),
       SQLT_STR);
  OCIDEF(octx->curo1,octx->c_middle_dp1,errhp,4,ordP->c_middle,
       SIZ(ordP->c_middle),SQLT_STR);
  OCIDEF(octx->curo1,octx->o_id_dp1,errhp,5,ADR(ordP->o_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_entry_d_dp1,errhp,6,
      ordP-> o_entry_d_base,SIZ(ordP->o_entry_d_base),SQLT_DAT);
  OCIDEF(octx->curo1,octx->o_cr_id_dp1,errhp,7,ADR(ordP->o_carrier_id),
       SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_ol_cnt_dp1,errhp,8,ADR(ordP->o_ol_cnt),
       SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->c_last_dp1,errhp,9,ordP->c_last,SIZ(ordP->c_last),
SQLT_STR);

/* Bind for third cursor , no-zero customer id */

OCIBND(octx->curo2,octx->w_id_bp2,errhp,":w_id",ADR(ordP->w_id),SIZ(int),SQLT_INT);
  OCIBND(octx->curo2,octx->d_id_bp2,errhp,":d_id",ADR(ordP->d_id),SIZ(int),SQLT_INT);
  OCIBND(octx->curo2,octx->c_id_bp,errhp,":c_id",ADR(ordP->c_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo2,octx->c_balance_dp2,errhp,1,ADR(ordP->c_balance),
       SIZ(double),SQLT_FLT);
  OCIDEF(octx->curo2,octx->c_first_dp2,errhp,2,ordP->c_first,SIZ(ordP->c_first),
       SQLT_STR);
  OCIDEF(octx->curo2,octx->c_middle_dp2,errhp,3,ordP->c_middle,
       SIZ(ordP->c_middle),SQLT_STR);
  OCIDEF(octx->curo2,octx->c_last_dp,errhp,4,ordP->c_last,SIZ(ordP->c_last),
SQLT_STR);
  OCIDEF(octx->curo2,octx->o_id_dp2,errhp,5,ADR(ordP->o_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo2,octx->o_entry_d_dp2,errhp,6,
       ordP->o_entry_d_base,SIZ(ordP->o_entry_d_base),SQLT_DAT);
  OCIDEF(octx->curo2, octx->o_cr_id_dp2,errhp,7,ADR(ordP->o_carrier_id),
       SIZ(int), SQLT_INT);
  OCIDEF(octx->curo2,octx->o_ol_cnt_dp2,errhp,8,ADR(ordP->o_ol_cnt),
       SIZ(int),SQLT_INT);

/* Bind for last cursor */


OCIBND(octx->curo3,octx->w_id_bp3,errhp,":w_id",ADR(ordP->w_id),SIZ(int),SQLT_INT);
  OCIBND(octx->curo3,octx->d_id_bp3,errhp,":d_id",ADR(ordP->d_id),SIZ(int),SQLT_INT);
  OCIBND(octx->curo3,octx->o_id_bp,errhp,":o_id",ADR(ordP->o_id),SIZ(int),SQLT_INT);

  OCIDFNRA(octx->curo3, octx->ol_i_id_dp, errhp, 1, ordP->ol_i_id,SIZ(int),SQLT_INT,
        octx->ol_i_id_ind,octx->ol_i_id_len, octx->ol_i_id_rcode);
  OCIDFNRA(octx->curo3,octx->ol_supply_w_id_dp,errhp,2,ordP->ol_supply_w_id,
        SIZ(int),SQLT_INT, octx->ol_supply_w_id_ind,
        octx->ol_supply_w_id_len, octx->ol_supply_w_id_rcode);
  OCIDFNRA(octx->curo3, octx->ol_quantity_dp,errhp,3, ordP->ol_quantity,SIZ(int),
        SQLT_INT, octx->ol_quantity_ind,octx->ol_quantity_len,
        octx->ol_quantity_rcode);
  OCIDFNRA(octx->curo3,octx->ol_amount_dp,errhp,4,ordP->ol_amount,SIZ(int),
        SQLT_INT,octx->ol_amount_ind, octx->ol_amount_len,
        octx->ol_amount_rcode);
  OCIDFNRA(octx->curo3,octx->ol_d_base_dp,errhp,5,ordP->ol_d_base,7, SQLT_DAT,
        octx->ol_delivery_d_ind,octx->ol_delivery_d_len,
        octx->ol_delivery_d_rcode);
#endif /* ISO8 */
  return (0);

}



plord (ora_cn_data_t *ora_SlotDataP)

{

  int i;
  int rcount;

  ordctx *octx = ora_SlotDataP->octx;
  global_order_t *ordP = ora_SlotDataP->ordP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  OCIError *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;

retry:

  for (i = 0; i < NITEMS; i++) {
    octx->ol_supply_w_id_ind[i] = TRUE;
    octx->ol_i_id_ind[i] = TRUE;
    octx->ol_quantity_ind[i] = TRUE;
    octx->ol_amount_ind[i] = TRUE;
    octx->ol_delivery_d_ind[i] = TRUE;
    octx->ol_supply_w_id_len[i] = sizeof(int);
    octx->ol_i_id_len[i] = sizeof(int);
    octx->ol_quantity_len[i] = sizeof(int);
    octx->ol_amount_len[i] = sizeof(int);
    octx->ol_delivery_d_len[i] = sizeof(ordP->ol_d_base[0]);
  }
  octx->ol_supply_w_id_csize = NITEMS;
  octx->ol_i_id_csize = NITEMS;
  octx->ol_quantity_csize = NITEMS;
  octx->ol_amount_csize = NITEMS;
  octx->ol_delivery_d_csize = NITEMS;
#ifdef ISO8
  OCIERROR(errhp,
    OCIStmtExecute(tpcsvc,octx->curo0,errhp,1,0,0,0,OCI_DEFAULT));
#else
  if(ordP->bylastname){

ordP->execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,3000,0,0,0,OCI_DEFAULT);
    if (ordP->execstatus != OCI_NO_DATA)   /* will get OCI_NO_DATA if <3000 found */
    {
       OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
       ordP->errcode = OCIERROR(errhp,ordP->execstatus);
       if(ordP->errcode == NOT_SERIALIZABLE){
          ordP->retries++;
          goto retry;
       } else if (ordP->errcode == RECOVERR) {
          ordP->retries++;
          goto retry;
       } else {
         return -1;
       }
    }
     /* get rowcount, find middle one */
    OCIAttrGet(octx->curo0,OCI_HTYPE_STM,&rcount,NULL,OCI_ATTR_ROWCNT,errhp);
    octx->cust_idx=(rcount+1)/2 ;
    ordP->execstatus = OCIStmtExecute(tpcsvc,octx->curo1,errhp,1,0,0,0,OCI_DEFAULT);
    if (ordP->execstatus != OCI_SUCCESS)
    {
       OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
       ordP->errcode = OCIERROR(errhp,ordP->execstatus);
       if(ordP->errcode == NOT_SERIALIZABLE){
          ordP->retries++;
          goto retry;
       } else if (ordP->errcode == RECOVERR) {
          ordP->retries++;
          goto retry;
       } else {
         return -1;
       }
    }

  } else {
    ordP->execstatus = OCIStmtExecute(tpcsvc,octx->curo2,errhp,1,0,0,0,OCI_DEFAULT);
    if (ordP->execstatus != OCI_SUCCESS)
    {
       OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
       ordP->errcode = OCIERROR(errhp,ordP->execstatus);
       if(ordP->errcode == NOT_SERIALIZABLE){
          ordP->retries++;
          goto retry;
       } else if (ordP->errcode == RECOVERR) {
          ordP->retries++;
          goto retry;
       } else {
         return -1;
       }
    }
  }
  octx->ol_w_id_ind = TRUE;
  octx->ol_d_id_ind = TRUE;
  octx->ol_o_id_ind = TRUE;
  octx->ol_w_id_len = sizeof(int);
  octx->ol_d_id_len = sizeof(int);
  octx->ol_o_id_len = sizeof(int);

  ordP->execstatus =   OCIStmtExecute(tpcsvc,octx->curo3,errhp,ordP->o_ol_cnt,0,0,0,
```

```
                OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
  if (ordP->execstatus != OCI_SUCCESS)
    {
       OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
       ordP->errcode = OCIERROR(errhp,ordP->execstatus);
       if(ordP->errcode == NOT_SERIALIZABLE){
          ordP->retries++;
          goto retry;
       } else if (ordP->errcode == RECOVERR) {
          ordP->retries++;
          goto retry;
       } else {
         return -1;
       }
    }
#ifdef NOTMORE
   OCIERROR(errhp,
      OCITransCommit(tpcsvc,errhp,OCI_DEFAULT));
#endif

   /* clean up and convert the delivery dates */
   for (i = 0; i < ordP->o_ol_cnt; i++) {
      if (octx->ol_delivery_d_ind[i] == -1) /* null date in field */
        strncpy(ordP->ol_delivery_d[i],"1-1-1811",10);
      else
        cvtdmy(ordP->ol_d_base[i],ordP->ol_delivery_d[i]);
   }

#endif
   return (0);

}



void plorddone (ora_cn_data_t *ora_SlotDataP)

{
  /*  TODO:  Should we free the cursor handles??  */

   if (ora_SlotDataP->octx) {
      free (ora_SlotDataP->octx);
      ora_SlotDataP->octx = NULL;
   }
   if (ora_SlotDataP->ordP) {
      free(ora_SlotDataP->ordP);
      ora_SlotDataP->ordP = NULL;
   }
}
```

plord.h

```
#ifndef TPCC_PLORD_H
#define TPCC_PLORD_H

#include <oci.h>

struct ordctx {
   sb2 c_rowid_ind[3000];
   sb2 ol_supply_w_id_ind[NITEMS];
   sb2 ol_i_id_ind[NITEMS];
   sb2 ol_quantity_ind[NITEMS];
   sb2 ol_amount_ind[NITEMS];
   sb2 ol_delivery_d_ind[NITEMS];
   sb2 ol_w_id_ind;
   sb2 ol_d_id_ind;
   sb2 ol_o_id_ind;
   sb2 c_id_ind;
   sb2 c_first_ind;
   sb2 c_middle_ind;
   sb2 c_balance_ind;
   sb2 c_last_ind;
   sb2 o_id_ind;
   sb2 o_entry_d_ind;
   sb2 o_carrier_id_ind;
   sb2 o_ol_cnt_ind;

   ub2 c_rowid_len[3000];
   ub2 ol_supply_w_id_len[NITEMS];
   ub2 ol_i_id_len[NITEMS];
   ub2 ol_quantity_len[NITEMS];
   ub2 ol_amount_len[NITEMS];
   ub2 ol_delivery_d_len[NITEMS];
   ub2 ol_w_id_len;
   ub2 ol_d_id_len;
   ub2 ol_o_id_len;

   ub2 c_rowid_rcode[3000];
   ub2 ol_supply_w_id_rcode[NITEMS];
   ub2 ol_i_id_rcode[NITEMS];
   ub2 ol_quantity_rcode[NITEMS];
   ub2 ol_amount_rcode[NITEMS];
   ub2 ol_delivery_d_rcode[NITEMS];
   ub2 ol_w_id_rcode;
   ub2 ol_d_id_rcode;
   ub2 ol_o_id_rcode;

   ub4 ol_supply_w_id_csize;
   ub4 ol_i_id_csize;
   ub4 ol_quantity_csize;
   ub4 ol_amount_csize;
   ub4 ol_delivery_d_csize;
   ub4 ol_w_id_csize;
   ub4 ol_d_id_csize;
   ub4 ol_o_id_csize;

   OCIStmt *curo0;
   OCIStmt *curo1;
   OCIStmt *curo2;
   OCIStmt *curo3;
   OCIBind *w_id_bp0;
   OCIBind *w_id_bp2;
   OCIBind *w_id_bp3;
   OCIBind *d_id_bp0;
   OCIBind *d_id_bp2;
   OCIBind *d_id_bp3;
   OCIBind *c_id_bp;
   OCIBind *byln_bp;
   OCIBind *c_last_bp;
   OCIBind *c_first_bp;
   OCIBind *c_middle_bp;
   OCIBind *c_balance_bp;
   OCIBind *o_id_bp;
   OCIBind *o_entry_d_bp;
   OCIBind *o_cr_id_bp;
   OCIBind *o_ol_cnt_bp;
   OCIBind *ol_s_w_id_bp;
   OCIBind *ol_i_id_bp;
   OCIBind *ol_quantity_bp;
   OCIBind *ol_amount_bp;
   OCIBind *ol_d_d_bp;
   OCIBind *c_rowid_bp;
   OCIDefine *c_rowid_dp;
   OCIDefine *c_last_dp;
   OCIDefine *c_last_dp1;
   OCIDefine *c_id_dp;
   OCIDefine *c_first_dp1;
   OCIDefine *c_first_dp2;
   OCIDefine *c_middle_dp1;
   OCIDefine *c_middle_dp2;
   OCIDefine *c_balance_dp1;
   OCIDefine *c_balance_dp2;
   OCIDefine *o_id_dp1;
   OCIDefine *o_id_dp2;
   OCIDefine *o_entry_d_dp1;
   OCIDefine *o_entry_d_dp2;
   OCIDefine *o_cr_id_dp1;
   OCIDefine *o_cr_id_dp2;
   OCIDefine *o_ol_cnt_dp1;
   OCIDefine *o_ol_cnt_dp2;
   OCIDefine *ol_d_d_dp;
   OCIDefine *ol_i_id_dp;
   OCIDefine *ol_supply_w_id_dp;
   OCIDefine *ol_quantity_dp;
   OCIDefine *ol_amount_dp;
   OCIDefine *ol_d_base_dp;

   OCIRowid *c_rowid_ptr[3000];
   int cs;
   int cust_idx;
   int norow;
};

typedef struct ordctx ordctx;

struct global_order_t {
  unsigned char ol_d_base[15][7];
  int w_id;
  int d_id;
  int c_id;
  char c_last[17];
  char c_first[17];
  char c_middle[3];
  double c_balance;
  int o_id;
  int o_carrier_id;
  int o_ol_cnt;
  int ol_supply_w_id[15];
  int ol_i_id[15];
  unsigned char o_entry_d_base[7];
  int ol_quantity[15];
  char ol_delivery_d[15][11];
  int ol_amount[15];
  int errcode;
  int execstatus;
  int retries;
  int bylastname;
  char o_entry_d[20];
```

```
};

typedef struct global_order_t global_order_t;


#endif  /* TPCC_PLORD_H  */
```

plpay.c

```
#ifdef RCSID
static char *RCSid =
   "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/plpay.c,v 1.2 1998/01/23 15:08:14 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*================================================================+
 |      Copyright (c) 1995  Oracle Corp, Redwood Shores, CA       |
 |                 OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                    All Rights Reserved                         |
 +================================================================+
 | FILENAME
 |    plpay.c
 | DESCRIPTION
 |    OCI version (using PL/SQL stored procedure) of
 |    PAYMENT transaction in TPC-C benchmark.
 +================================================================*/

#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"

#define SQLTXT_ZERO "BEGIN apayment.adopayment(:w_id,:d_id, :c_w_id, :c_d_id, \
   :c_id, 0, \
   :h_amount, :c_last, :w_street_1, :w_street_2, :w_city, :w_state, \
   :w_zip, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip, :c_first, \
   :c_middle, :c_street_1, :c_street_2, :c_city, :c_state, :c_zip, :c_phone, \
   :c_since, :c_credit, :c_credit_lim, :c_discount, :c_balance, :c_data, \
   :h_date, :retry, :cr_date); END;"

#define SQLTXT_NONZERO "BEGIN apayment.adopayment(:w_id,:d_id, :c_w_id, :c_d_id,
\
   :c_id, 1, \
   :h_amount, :c_last, :w_street_1, :w_street_2, :w_city, :w_state, \
   :w_zip, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip, :c_first, \
   :c_middle, :c_street_1, :c_street_2, :c_city, :c_state, :c_zip, :c_phone, \
   :c_since, :c_credit, :c_credit_lim, :c_discount, :c_balance, :c_data, \
   :h_date, :retry, :cr_date); END;"

#define SQLTXT_INIT "BEGIN pay.pay_init; END;"


plpayinit (ora_cn_data_t *ora_SlotDataP)

{
   char *ora_home = getenv("ORACLE_HOME");
   char sql_file_name[256];
   payctx *pctx;
   global_payment_t *payP;
   OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
   OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
   OCIError *errhp = ora_SlotDataP->errhp;
   OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
   OCISession *tpcusr = ora_SlotDataP->tpcusr;
   OCIStmt *curi = ora_SlotDataP->curi;

   text stmbuf[SQL_BUF_SIZE];

   if (!ora_home) {
      err_printf("Cannot find env variable ORACLE_HOME\n");
      exit(13);
   }

   pctx = (payctx *)malloc(sizeof(payctx));
   memset(pctx,(char)0,sizeof(payctx));
   ora_SlotDataP->pctx = pctx;

   ora_SlotDataP->payP = (global_payment_t*)malloc(sizeof(global_payment_t));
   memset(ora_SlotDataP->payP,(char)0,sizeof(global_payment_t));
   payP = ora_SlotDataP->payP;

/* cursor for init */
   OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curpi)),
         OCI_HTYPE_STMT,0,(dvoid**)0));

   OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curp0)),
         OCI_HTYPE_STMT,0,(dvoid**)0));
   OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curp1)),
         OCI_HTYPE_STMT,0,(dvoid**)0));

   /*  build the init statement  and execute it */

   sprintf ((char*)stmbuf, SQLTXT_INIT);
   OCIERROR(errhp,OCIStmtPrepare(pctx->curpi, errhp, stmbuf,
         strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
   OCIERROR(errhp,
     OCIStmtExecute(tpcsvc,pctx->curpi,errhp,1,0,0,0,OCI_DEFAULT));

   /* customer id !=  0, go by last name*/
#ifdef ATOMA

   sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/paynz_abort.sql",ora_home);
   sqlfile(sql_file_name,stmbuf);


/*   sqlfile("../../blocks/paynz_abort.sql",stmbuf);  */
   /* sprintf((char *)stmbuf, SQLTXT_NONZERO); */
#else
   sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/paynz.sql",ora_home);
   sqlfile(sql_file_name,stmbuf);

/*     sqlfile("../../blocks/paynz.sql",stmbuf);  */
#endif
   OCIERROR(errhp,OCIStmtPrepare(pctx->curp0, errhp, stmbuf,
         strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

   /* customer id ==  0, go by last name*/
#ifdef ATOMA
   sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/payz_abort.sql",ora_home);
   sqlfile(sql_file_name,stmbuf);

/*    sqlfile("../../blocks/payz_abort.sql",stmbuf); */
   /* sprintf((char *)stmbuf, SQLTXT_ZERO); */
#else
   sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/payz.sql",ora_home);
   sqlfile(sql_file_name,stmbuf);

/*    sqlfile("../../blocks/payz.sql",stmbuf); */
#endif
   OCIERROR(errhp,OCIStmtPrepare(pctx->curp1, errhp, stmbuf,
         strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

   pctx->w_id_ind = TRUE;
   pctx->w_id_len = SIZ(payP->w_id);
   pctx->d_id_ind = TRUE;
   pctx->d_id_len = SIZ(payP->d_id);
   pctx->c_w_id_ind = TRUE;
   pctx->c_w_id_len = SIZ(payP->c_w_id);
   pctx->c_d_id_ind = TRUE;
   pctx->c_d_id_len = SIZ(payP->c_d_id);
   pctx->c_id_ind = TRUE;
   pctx->c_id_len = 0;
   pctx->h_amount_len = SIZ(payP->h_amount);
   pctx->h_amount_ind = TRUE;
   pctx->c_last_ind = TRUE;
   pctx->c_last_len = 0;
   pctx->w_street_1_ind = TRUE;
   pctx->w_street_1_len = 0;
   pctx->w_street_2_ind = TRUE;
   pctx->w_street_2_len = 0;
   pctx->w_city_ind = TRUE;
   pctx->w_city_len = 0;
   pctx->w_state_ind = TRUE;
   pctx->w_state_len = 0;
   pctx->w_zip_ind = TRUE;
   pctx->w_zip_len = 0;
   pctx->d_street_1_ind = TRUE;
   pctx->d_street_1_len = 0;
   pctx->d_street_2_ind = TRUE;
   pctx->d_street_2_len = 0;
   pctx->d_city_ind = TRUE;
   pctx->d_city_len = 0;
   pctx->d_state_ind = TRUE;
   pctx->d_state_len = 0;
   pctx->d_zip_ind = TRUE;
   pctx->d_zip_len = 0;
   pctx->c_first_ind = TRUE;
   pctx->c_first_len = 0;
   pctx->c_middle_ind = TRUE;
   pctx->c_middle_len = 0;
   pctx->c_street_1_ind = TRUE;
   pctx->c_street_1_len = 0;
   pctx->c_street_2_ind = TRUE;
   pctx->c_street_2_len = 0;
   pctx->c_city_ind = TRUE;
   pctx->c_city_len = 0;
   pctx->c_state_ind = TRUE;
   pctx->c_state_len = 0;
   pctx->c_zip_ind = TRUE;
   pctx->c_zip_len = 0;
   pctx->c_phone_ind = TRUE;
   pctx->c_phone_len = 0;
   pctx->c_since_ind = TRUE;
   pctx->c_since_len = 0;
   pctx->c_credit_ind = TRUE;
   pctx->c_credit_len = 0;
   pctx->c_credit_lim_ind = TRUE;
   pctx->c_credit_lim_len = 0;
```

```
 pctx->c_discount_ind = TRUE;
 pctx->c_discount_len = 0;
 pctx->c_balance_ind = TRUE;
 pctx->c_balance_len = sizeof(double);
 pctx->c_data_ind = TRUE;
 pctx->c_data_len = 0;
 pctx->h_date_ind = TRUE;
 pctx->h_date_len = 0;
 pctx->retries_ind = TRUE;
 pctx->retries_len = 0;
 pctx->cr_date_ind = TRUE;
 pctx->cr_date_len = 7;



 /* bind variables */


  OCIBNDR(pctx->curp0, pctx->w_id_bp, errhp,":w_id",ADR(payP->w_id),SIZ(int),
       SQLT_INT, &pctx->w_id_ind, NULL, NULL);
  OCIBNDR(pctx->curp0, pctx->d_id_bp, errhp,":d_id",ADR(payP->d_id),SIZ(int),
       SQLT_INT, &pctx->d_id_ind, NULL, NULL);
  OCIBND(pctx->curp0, pctx->c_w_id_bp, errhp,":c_w_id",ADR(payP->c_w_id),SIZ(int),
       SQLT_INT);
  OCIBND(pctx->curp0, pctx->c_d_id_bp, errhp,":c_d_id",ADR(payP->c_d_id),SIZ(int),
       SQLT_INT);
  OCIBND(pctx->curp0, pctx->c_id_bp, errhp,":c_id",ADR(payP->c_id),SIZ(int),
       SQLT_INT);
  OCIBNDR(pctx->curp0, pctx->h_amount_bp, errhp,":h_amount",ADR(payP->h_amount),
       SIZ(int),SQLT_INT, &pctx->h_amount_ind, &pctx->h_amount_len,
       &pctx->h_amount_rc);
  OCIBNDR(pctx->curp0, pctx->c_last_bp, errhp,":c_last",payP->c_last,SIZ(payP->c_last),
       SQLT_STR, &pctx->c_last_ind, &pctx->c_last_len, &pctx->c_last_rc);
  OCIBNDR(pctx->curp0, pctx->w_street_1_bp, errhp,":w_street_1",payP->w_street_1,
       SIZ(payP->w_street_1),SQLT_STR, &pctx->w_street_1_ind,
       &pctx->w_street_1_len, &pctx->w_street_1_rc);
  OCIBNDR(pctx->curp0, pctx->w_street_2_bp, errhp,":w_street_2",payP->w_street_2,
       SIZ(payP->w_street_2),SQLT_STR, &pctx->w_street_2_ind,
       &pctx->w_street_2_len, &pctx->w_street_2_rc);
  OCIBNDR(pctx->curp0, pctx->w_city_bp,
errhp,":w_city",payP->w_city,SIZ(payP->w_city),
       SQLT_STR, &pctx->w_city_ind, &pctx->w_city_len, &pctx->w_city_rc);
  OCIBNDR(pctx->curp0, pctx->w_state_bp,
errhp,":w_state",payP->w_state,SIZ(payP->w_state),
       SQLT_STR, &pctx->w_state_ind, &pctx->w_state_len, &pctx->w_state_rc);
  OCIBNDR(pctx->curp0, pctx->w_zip_bp, errhp,":w_zip",payP->w_zip,SIZ(payP->w_zip),
       SQLT_STR, &pctx->w_zip_ind, &pctx->w_zip_len, &pctx->w_zip_rc);
  OCIBNDR(pctx->curp0, pctx->d_street_1_bp, errhp,":d_street_1",payP->d_street_1,
       SIZ(payP->d_street_1),SQLT_STR, &pctx->d_street_1_ind,
       &pctx->d_street_1_len, &pctx->d_street_1_rc);
  OCIBNDR(pctx->curp0, pctx->d_street_2_bp, errhp,":d_street_2",payP->d_street_2,
       SIZ(payP->d_street_2),SQLT_STR, &pctx->d_street_2_ind,
       &pctx->d_street_2_len, &pctx->d_street_2_rc);
  OCIBNDR(pctx->curp0, pctx->d_city_bp, errhp,":d_city",payP->d_city,SIZ(payP->d_city),
       SQLT_STR, &pctx->d_city_ind, &pctx->d_city_len, &pctx->d_city_rc);
  OCIBNDR(pctx->curp0, pctx->d_state_bp,
errhp,":d_state",payP->d_state,SIZ(payP->d_state),
       SQLT_STR, &pctx->d_state_ind, &pctx->d_state_len, &pctx->d_state_rc);
  OCIBNDR(pctx->curp0, pctx->d_zip_bp, errhp,":d_zip",payP->d_zip,SIZ(payP->d_zip),
       SQLT_STR, &pctx->d_zip_ind, &pctx->d_zip_len, &pctx->d_zip_rc);
  OCIBNDR(pctx->curp0, pctx->c_first_bp, errhp,":c_first",payP->c_first,SIZ(payP->c_first),
       SQLT_STR, &pctx->c_first_ind, &pctx->c_first_len, &pctx->c_first_rc);
  OCIBNDR(pctx->curp0, pctx->c_middle_bp, errhp,":c_middle",payP->c_middle,2,
       SQLT_AFC, &pctx->c_middle_ind, &pctx->c_middle_len,
       &pctx->c_middle_rc);
  OCIBNDR(pctx->curp0, pctx->c_street_1_bp, errhp,":c_street_1",payP->c_street_1,
       SIZ(payP->c_street_1),SQLT_STR, &pctx->c_street_1_ind,
       &pctx->c_street_1_len, &pctx->c_street_1_rc);
  OCIBNDR(pctx->curp0, pctx->c_street_2_bp, errhp,":c_street_2",payP->c_street_2,
       SIZ(payP->c_street_2),SQLT_STR, &pctx->c_street_2_ind,
       &pctx->c_street_2_len, &pctx->c_street_2_rc);
  OCIBNDR(pctx->curp0, pctx->c_city_bp, errhp,":c_city",payP->c_city,SIZ(payP->c_city),
       SQLT_STR, &pctx->c_city_ind, &pctx->c_city_len, &pctx->c_city_rc);
  OCIBNDR(pctx->curp0, pctx->c_state_bp,
errhp,":c_state",payP->c_state,SIZ(payP->c_state),
       SQLT_STR, &pctx->c_state_ind, &pctx->c_state_len, &pctx->c_state_rc);
  OCIBNDR(pctx->curp0, pctx->c_zip_bp, errhp,":c_zip",payP->c_zip,SIZ(payP->c_zip),
       SQLT_STR, &pctx->c_zip_ind, &pctx->c_zip_len, &pctx->c_zip_rc);
  OCIBNDR(pctx->curp0, pctx->c_phone_bp,
errhp,":c_phone",payP->c_phone,SIZ(payP->c_phone),         SQLT_STR,
&pctx->c_phone_ind, &pctx->c_phone_len, &pctx->c_phone_rc);
  OCIBNDR(pctx->curp0, pctx->c_since_bp, errhp,":c_since",payP->c_since,7,
       SQLT_DAT, &pctx->c_since_ind, &pctx->c_since_len, &pctx->c_since_rc);
  OCIBNDR(pctx->curp0, pctx->c_credit_bp, errhp,":c_credit",payP->c_credit,
       SIZ(payP->c_credit),SQLT_CHR, &pctx->c_credit_ind, &pctx->c_credit_len,
       &pctx->c_credit_rc);
  OCIBNDR(pctx->curp0, pctx->c_credit_lim_bp, errhp,":c_credit_lim",
       ADR(payP->c_credit_lim),SIZ(int),SQLT_INT, &pctx->c_credit_lim_ind,
       &pctx->c_credit_lim_len, &pctx->c_credit_lim_rc);
  OCIBNDR(pctx->curp0, pctx->c_discount_bp, errhp,":c_discount",
       ADR(payP->c_discount),SIZ(int),SQLT_INT, &pctx->c_discount_ind,
       &pctx->c_discount_len, &pctx->c_discount_rc);
  OCIBNDR(pctx->curp0, pctx->c_balance_bp, errhp,":c_balance",ADR(payP->c_balance),
       SIZ(double),SQLT_FLT, &pctx->c_balance_ind, &pctx->c_balance_len,
       &pctx->c_balance_rc);
```

```
  OCIBNDR(pctx->curp0, pctx->c_data_bp,
errhp,":c_data",payP->c_data,SIZ(payP->c_data),
       SQLT_STR, &pctx->c_data_ind, &pctx->c_data_len, &pctx->c_data_rc);

  OCIBNDR(pctx->curp0, pctx->h_date_bp,
errhp,":h_date",payP->h_date,SIZ(payP->h_date),
       SQLT_STR, &pctx->h_date_ind, &pctx->h_date_len, &pctx->h_date_rc);

  OCIBNDR(pctx->curp0, pctx->retries_bp, errhp,":retry",ADR(payP->retries),SIZ(int),
       SQLT_INT, &pctx->retries_ind, &pctx->retries_len, &pctx->retries_rc);
  OCIBNDR(pctx->curp0, pctx->cr_date_bp, errhp,":cr_date",ADR(payP->cr_date),
       SIZ(payP->cr_date),SQLT_DAT, &pctx->cr_date_ind, &pctx->cr_date_len,
       &pctx->cr_date_rc);

 /* ---- Binds for  the second cursor  */


  OCIBNDR(pctx->curp1, pctx->w_id_bp1, errhp,":w_id",ADR(payP->w_id),SIZ(int),
       SQLT_INT, &pctx->w_id_ind, &pctx->w_id_len, &pctx->w_id_rc);
  OCIBNDR(pctx->curp1, pctx->d_id_bp1, errhp,":d_id",ADR(payP->d_id),SIZ(int),
       SQLT_INT, &pctx->d_id_ind, &pctx->d_id_len, &pctx->d_id_rc);
  OCIBND(pctx->curp1, pctx->c_w_id_bp1, errhp,":c_w_id",ADR(payP->c_w_id),SIZ(int),
       SQLT_INT);
  OCIBND(pctx->curp1, pctx->c_d_id_bp1, errhp,":c_d_id",ADR(payP->c_d_id),SIZ(int),
       SQLT_INT);
  OCIBNDR(pctx->curp1, pctx->c_id_bp1, errhp,":c_id",ADR(payP->c_id),SIZ(int),
       SQLT_INT, &pctx->c_id_ind, &pctx->c_id_len, &pctx->c_id_rc);
  OCIBNDR(pctx->curp1, pctx->h_amount_bp1, errhp,":h_amount",ADR(payP->h_amount),
       SIZ(int),SQLT_INT, &pctx->h_amount_ind, &pctx->h_amount_len,
       &pctx->h_amount_rc);
  OCIBND(pctx->curp1, pctx->c_last_bp1, errhp,":c_last",payP->c_last,SIZ(payP->c_last),
       SQLT_STR);
  OCIBNDR(pctx->curp1, pctx->w_street_1_bp1, errhp,":w_street_1",payP->w_street_1,
       SIZ(payP->w_street_1),SQLT_STR, &pctx->w_street_1_ind,
       &pctx->w_street_1_len, &pctx->w_street_1_rc);
  OCIBNDR(pctx->curp1, pctx->w_street_2_bp1, errhp,":w_street_2",payP->w_street_2,
       SIZ(payP->w_street_2),SQLT_STR, &pctx->w_street_2_ind,
       &pctx->w_street_2_len, &pctx->w_street_2_rc);
  OCIBNDR(pctx->curp1, pctx->w_city_bp1,
errhp,":w_city",payP->w_city,SIZ(payP->w_city),
       SQLT_STR, &pctx->w_city_ind, &pctx->w_city_len, &pctx->w_city_rc);
  OCIBNDR(pctx->curp1, pctx->w_state_bp1,
errhp,":w_state",payP->w_state,SIZ(payP->w_state),
       SQLT_STR, &pctx->w_state_ind, &pctx->w_state_len, &pctx->w_state_rc);
  OCIBNDR(pctx->curp1, pctx->w_zip_bp1, errhp,":w_zip",payP->w_zip,SIZ(payP->w_zip),
       SQLT_STR, &pctx->w_zip_ind, &pctx->w_zip_len, &pctx->w_zip_rc);
  OCIBNDR(pctx->curp1, pctx->d_street_1_bp1, errhp,":d_street_1",payP->d_street_1,
       SIZ(payP->d_street_1),SQLT_STR, &pctx->d_street_1_ind,
       &pctx->d_street_1_len, &pctx->d_street_1_rc);
  OCIBNDR(pctx->curp1, pctx->d_street_2_bp1, errhp,":d_street_2",payP->d_street_2,
       SIZ(payP->d_street_2),SQLT_STR, &pctx->d_street_2_ind,
       &pctx->d_street_2_len, &pctx->d_street_2_rc);
  OCIBNDR(pctx->curp1, pctx->d_city_bp1,
errhp,":d_city",payP->d_city,SIZ(payP->d_city),
       SQLT_STR, &pctx->d_city_ind, &pctx->d_city_len, &pctx->d_city_rc);
  OCIBNDR(pctx->curp1, pctx->d_state_bp1, errhp,":d_state",payP->d_state,
       SIZ(payP->d_state), SQLT_STR, &pctx->d_state_ind, &pctx->d_state_len,
       &pctx->d_state_rc);
  OCIBNDR(pctx->curp1, pctx->d_zip_bp1, errhp,":d_zip",payP->d_zip,SIZ(payP->d_zip),
       SQLT_STR, &pctx->d_zip_ind, &pctx->d_zip_len, &pctx->d_zip_rc);
  OCIBNDR(pctx->curp1, pctx->c_first_bp1, errhp,":c_first",payP->c_first,
       SIZ(payP->c_first), SQLT_STR, &pctx->c_first_ind, &pctx->c_first_len,
       &pctx->c_first_rc);
  OCIBNDR(pctx->curp1, pctx->c_middle_bp1, errhp,":c_middle",payP->c_middle,2,
       SQLT_AFC, &pctx->c_middle_ind, &pctx->c_middle_len,
       &pctx->c_middle_rc);

  OCIBNDR(pctx->curp1, pctx->c_street_1_bp1, errhp,":c_street_1",payP->c_street_1,
       SIZ(payP->c_street_1),SQLT_STR, &pctx->c_street_1_ind,
       &pctx->c_street_1_len, &pctx->c_street_1_rc);
  OCIBNDR(pctx->curp1, pctx->c_street_2_bp1, errhp,":c_street_2",payP->c_street_2,
       SIZ(payP->c_street_2),SQLT_STR, &pctx->c_street_2_ind,
       &pctx->c_street_2_len, &pctx->c_street_2_rc);
  OCIBNDR(pctx->curp1, pctx->c_city_bp1, errhp,":c_city",payP->c_city,
       SIZ(payP->c_city),SQLT_STR,
       &pctx->c_city_ind, &pctx->c_city_len, &pctx->c_city_rc);
  OCIBNDR(pctx->curp1, pctx->c_state_bp1, errhp,":c_state",payP->c_state,
       SIZ(payP->c_state),SQLT_STR, &pctx->c_state_ind, &pctx->c_state_len,
       &pctx->c_state_rc);
  OCIBNDR(pctx->curp1, pctx->c_zip_bp1, errhp,":c_zip",payP->c_zip,SIZ(payP->c_zip),
       SQLT_STR, &pctx->c_zip_ind, &pctx->c_zip_len, &pctx->c_zip_rc);
  OCIBNDR(pctx->curp1, pctx->c_phone_bp1, errhp,":c_phone",payP->c_phone,
       SIZ(payP->c_phone), SQLT_STR, &pctx->c_phone_ind, &pctx->c_phone_len,
       &pctx->c_phone_rc);
  OCIBNDR(pctx->curp1, pctx->c_since_bp1, errhp,":c_since",payP->c_since,7,
       SQLT_DAT, &pctx->c_since_ind, &pctx->c_since_len, &pctx->c_since_rc);
  OCIBNDR(pctx->curp1, pctx->c_credit_bp1, errhp,":c_credit",payP->c_credit,
       SIZ(payP->c_credit),SQLT_CHR, &pctx->c_credit_ind, &pctx->c_credit_len,
       &pctx->c_credit_rc);
  OCIBNDR(pctx->curp1, pctx->c_credit_lim_bp1, errhp,":c_credit_lim",
       ADR(payP->c_credit_lim),SIZ(int),SQLT_INT, &pctx->c_credit_lim_ind,
       &pctx->c_credit_lim_len, &pctx->c_credit_lim_rc);
  OCIBNDR(pctx->curp1, pctx->c_discount_bp1, errhp,":c_discount",
       ADR(payP->c_discount),SIZ(int),SQLT_INT, &pctx->c_discount_ind,
       &pctx->c_discount_len, &pctx->c_discount_rc);
```

```
  OCIBNDR(pctx->curp1, pctx->c_balance_bp1,
errhp,":c_balance",ADR(payP->c_balance),
        SIZ(double),SQLT_FLT,&pctx->c_balance_ind, &pctx->c_balance_len,
        &pctx->c_balance_rc);
  OCIBNDR(pctx->curp1, pctx->c_data_bp1,
errhp,":c_data",payP->c_data,SIZ(payP->c_data),
        SQLT_STR, &pctx->c_data_ind, &pctx->c_data_len, &pctx->c_data_rc);

  OCIBNDR(pctx->curp1, pctx->h_date_bp1,
errhp,":h_date",payP->h_date,SIZ(payP->h_date),
        SQLT_STR, &pctx->h_date_ind, &pctx->h_date_len, &pctx->h_date_rc);

  OCIBNDR(pctx->curp1, pctx->retries_bp1, errhp,":retry",ADR(payP->retries),SIZ(int),
        SQLT_INT, &pctx->retries_ind, &pctx->retries_len, &pctx->retries_rc);
  OCIBNDR(pctx->curp1, pctx->cr_date_bp1, errhp,":cr_date",ADR(payP->cr_date),
        SIZ(payP->cr_date),SQLT_DAT,&pctx->cr_date_ind, &pctx->cr_date_len,
        &pctx->cr_date_rc);

  return (0);

}



plpay (ora_cn_data_t *ora_SlotDataP)
{
  payctx *pctx = ora_SlotDataP->pctx;
  global_payment_t *payP = ora_SlotDataP->payP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  OCIError *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;


retry:
  pctx->w_id_ind = TRUE;
  pctx->w_id_len = SIZ(payP->w_id);
  pctx->d_id_ind = TRUE;
  pctx->d_id_len = SIZ(payP->d_id);
  pctx->c_w_id_ind = TRUE;
  pctx->c_w_id_len = 0;
  pctx->c_d_id_ind = TRUE;
  pctx->c_d_id_len = 0;
  pctx->c_id_ind = TRUE;
  pctx->c_id_len = 0;
  pctx->h_amount_len = SIZ(payP->h_amount);
  pctx->h_amount_ind = TRUE;
  pctx->c_last_ind = TRUE;
  pctx->c_last_len = SIZ(payP->c_last);
  pctx->w_street_1_ind = TRUE;
  pctx->w_street_1_len = 0;
  pctx->w_street_2_ind = TRUE;
  pctx->w_street_2_len = 0;
  pctx->w_city_ind = TRUE;
  pctx->w_city_len = 0;
  pctx->w_state_ind = TRUE;
  pctx->w_state_len = 0;
  pctx->w_zip_ind = TRUE;
  pctx->w_zip_len = 0;
  pctx->d_street_1_ind = TRUE;
  pctx->d_street_1_len = 0;
  pctx->d_street_2_ind = TRUE;
  pctx->d_street_2_len = 0;
  pctx->d_city_ind = TRUE;
  pctx->d_city_len = 0;
  pctx->d_state_ind = TRUE;
  pctx->d_state_len = 0;
  pctx->d_zip_ind = TRUE;
  pctx->d_zip_len = 0;
  pctx->c_first_ind = TRUE;
  pctx->c_first_len = 0;
  pctx->c_middle_ind = TRUE;
  pctx->c_middle_len = 0;
  pctx->c_street_1_ind = TRUE;
  pctx->c_street_1_len = 0;
  pctx->c_street_2_ind = TRUE;
  pctx->c_street_2_len = 0;
  pctx->c_city_ind = TRUE;
  pctx->c_city_len = 0;
  pctx->c_state_ind = TRUE;
  pctx->c_state_len = 0;
  pctx->c_zip_ind = TRUE;
  pctx->c_zip_len = 0;
  pctx->c_phone_ind = TRUE;
  pctx->c_phone_len = 0;
  pctx->c_since_ind = TRUE;
  pctx->c_since_len = 0;
  pctx->c_credit_ind = TRUE;
  pctx->c_credit_len = 0;
  pctx->c_credit_lim_ind = TRUE;
  pctx->c_credit_lim_len = 0;
  pctx->c_discount_ind = TRUE;
  pctx->c_discount_len = 0;
  pctx->c_balance_ind = TRUE;
  pctx->c_balance_len = sizeof(double);
  pctx->c_data_ind = TRUE;
  pctx->c_data_len = 0;
  pctx->h_date_ind = TRUE;
  pctx->h_date_len = 0;
  pctx->retries_ind = TRUE;
  pctx->retries_len = 0;
  pctx->cr_date_ind = TRUE;
  pctx->cr_date_len = 7;

  if(payP->bylastname){

    payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp1,errhp,1,0,0,0,OCI_DEFAULT);
  } else {
    payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp0,errhp,1,0,0,0,OCI_DEFAULT);
  }
  if(payP->execstatus != OCI_SUCCESS) {
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     payP->errcode = OCIERROR(errhp,payP->execstatus);
     if(payP->errcode == NOT_SERIALIZABLE){
        payP->retries++;
        goto retry;
     } else if (payP->errcode == RECOVERR) {
        payP->retries++;
        goto retry;
     } else {
        return -1;
     }
  }
  return (0);

}



 void plpaydone(ora_cn_data_t *ora_SlotDataP)
 {
   /*  TODO:  Should we free the cursor handles??  */
   if(ora_SlotDataP->pctx) {
     free(ora_SlotDataP->pctx);
     ora_SlotDataP->pctx = NULL;
   }
   if (ora_SlotDataP->payP) {
     free(ora_SlotDataP->payP);
     ora_SlotDataP->payP = NULL;
   }
 }
```

plpay.h

```
 #ifndef TPCC_PLPAY_H
 #define TPCC_PLPAY_H
 #include <oci.h>

 struct payctx {
   OCIStmt *curpi;
   OCIStmt *curp0;
   OCIStmt *curp1;
   OCIBind *w_id_bp;
   OCIBind *w_id_bp1;
   sb2 w_id_ind;
   ub2 w_id_len;
   ub2 w_id_rc;

   OCIBind *d_id_bp;
   OCIBind *d_id_bp1;
   sb2 d_id_ind;
   ub2 d_id_len;
   ub2 d_id_rc;

   OCIBind *c_w_id_bp;
   OCIBind *c_w_id_bp1;
   sb2 c_w_id_ind;
   ub2 c_w_id_len;
   ub2 c_w_id_rc;

   OCIBind *c_d_id_bp;
   OCIBind *c_d_id_bp1;
   sb2 c_d_id_ind;
   ub2 c_d_id_len;
   ub2 c_d_id_rc;

   OCIBind *c_id_bp;
   OCIBind *c_id_bp1;
   sb2 c_id_ind;
   ub2 c_id_len;
   ub2 c_id_rc;

   OCIBind *h_amount_bp;
   OCIBind *h_amount_bp1;
   sb2 h_amount_ind;
   ub2 h_amount_len;
```

```
ub2 h_amount_rc;

OCIBind *c_last_bp;
OCIBind *c_last_bp1;
sb2 c_last_ind;
ub2 c_last_len;
ub2 c_last_rc;

OCIBind *w_street_1_bp;
OCIBind *w_street_1_bp1;
sb2 w_street_1_ind;
ub2 w_street_1_len;
ub2 w_street_1_rc;

OCIBind *w_street_2_bp;
OCIBind *w_street_2_bp1;
sb2 w_street_2_ind;
ub2 w_street_2_len;
ub2 w_street_2_rc;

OCIBind *w_city_bp;
OCIBind *w_city_bp1;
sb2 w_city_ind;
ub2 w_city_len;
ub2 w_city_rc;

OCIBind *w_state_bp;
OCIBind *w_state_bp1;
sb2 w_state_ind;
ub2 w_state_len;
ub2 w_state_rc;

OCIBind *w_zip_bp;
OCIBind *w_zip_bp1;
sb2 w_zip_ind;
ub2 w_zip_len;
ub2 w_zip_rc;

OCIBind *d_street_1_bp;
OCIBind *d_street_1_bp1;
sb2 d_street_1_ind;
ub2 d_street_1_len;
ub2 d_street_1_rc;

OCIBind *d_street_2_bp;
OCIBind *d_street_2_bp1;
sb2 d_street_2_ind;
ub2 d_street_2_len;
ub2 d_street_2_rc;

OCIBind *d_city_bp;
OCIBind *d_city_bp1;
sb2 d_city_ind;
ub2 d_city_len;
ub2 d_city_rc;

OCIBind *d_state_bp;
OCIBind *d_state_bp1;
sb2 d_state_ind;
ub2 d_state_len;
ub2 d_state_rc;

OCIBind *d_zip_bp;
OCIBind *d_zip_bp1;
sb2 d_zip_ind;
ub2 d_zip_len;
ub2 d_zip_rc;

OCIBind *c_first_bp;
OCIBind *c_first_bp1;
sb2 c_first_ind;
ub2 c_first_len;
ub2 c_first_rc;

OCIBind *c_middle_bp;
OCIBind *c_middle_bp1;
sb2 c_middle_ind;
ub2 c_middle_len;
ub2 c_middle_rc;

OCIBind *c_street_1_bp;
OCIBind *c_street_1_bp1;
sb2 c_street_1_ind;
ub2 c_street_1_len;
ub2 c_street_1_rc;

OCIBind *c_street_2_bp;
OCIBind *c_street_2_bp1;
sb2 c_street_2_ind;
ub2 c_street_2_len;
ub2 c_street_2_rc;

OCIBind *c_city_bp;
OCIBind *c_city_bp1;
sb2 c_city_ind;
ub2 c_city_len;
ub2 c_city_rc;

OCIBind *c_state_bp;
OCIBind *c_state_bp1;
sb2 c_state_ind;
ub2 c_state_len;
ub2 c_state_rc;

OCIBind *c_zip_bp;
OCIBind *c_zip_bp1;
sb2 c_zip_ind;
ub2 c_zip_len;
ub2 c_zip_rc;

OCIBind *c_phone_bp;
OCIBind *c_phone_bp1;
sb2 c_phone_ind;
ub2 c_phone_len;
ub2 c_phone_rc;

OCIBind *c_since_bp;
OCIBind *c_since_bp1;
sb2 c_since_ind;
ub2 c_since_len;
ub2 c_since_rc;

OCIBind *c_credit_bp;
OCIBind *c_credit_bp1;
sb2 c_credit_ind;
ub2 c_credit_len;
ub2 c_credit_rc;

OCIBind *c_credit_lim_bp;
OCIBind *c_credit_lim_bp1;
sb2 c_credit_lim_ind;
ub2 c_credit_lim_len;
ub2 c_credit_lim_rc;

OCIBind *c_discount_bp;
OCIBind *c_discount_bp1;
sb2 c_discount_ind;
ub2 c_discount_len;
ub2 c_discount_rc;

OCIBind *c_balance_bp;
OCIBind *c_balance_bp1;
sb2 c_balance_ind;
ub2 c_balance_len;
ub2 c_balance_rc;

OCIBind *c_data_bp;
OCIBind *c_data_bp1;
sb2 c_data_ind;
ub2 c_data_len;
ub2 c_data_rc;

OCIBind *h_date_bp;
OCIBind *h_date_bp1;
sb2 h_date_ind;
ub2 h_date_len;
ub2 h_date_rc;

OCIBind *retries_bp;
OCIBind *retries_bp1;
sb2 retries_ind;
ub2 retries_len;
ub2 retries_rc;

OCIBind *cr_date_bp;
OCIBind *cr_date_bp1;
sb2 cr_date_ind;
ub2 cr_date_len;
ub2 cr_date_rc;

OCIBind *byln_bp;
sb2 byln_ind;
ub2 byln_len;
ub2 byln_rc;
};

typedef struct payctx payctx;


struct global_payment_t{
  int w_id;
  int d_id;
  int c_id;
  char c_last[17];
  char c_first[17];
  char c_middle[3];
  double c_balance;
  int retries;
  int bylastname;
  unsigned char c_since[7];
```

```
 int execstatus;
 int errcode;
 int c_w_id;
 int c_d_id;
 int h_amount;
 char w_street_1[21];
 char w_street_2[21];
 char w_city[21];
 char w_state[3];
 char w_zip[10];
 char d_street_1[21];
 char d_street_2[21];
 char d_city[21];
 char d_state[3];
 char d_zip[10];
 char c_street_1[21];
 char c_street_2[21];
 char c_city[21];
 char c_state[3];
 char c_zip[10];
 char c_phone[17];
 char c_since_d[11];
 int c_discount;
 char c_credit[3];
 int c_credit_lim;
 char c_data[201];
 char h_date[20];
 unsigned char cr_date[7];
};

typedef struct global_payment_t global_payment_t;



#endif  /* TPCC_PLPAY_H  */
```

plsto.c

```
#ifdef RCSID
static char *RCSid =
  "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/plsto.c,v 1.2 1998/01/23 15:08:16 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*+=================================================================+
 |       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
 |                  OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                     All Rights Reserved                         |
 +=================================================================+
 | FILENAME
 |    plsto.c
 | DESCRIPTION
 |    OCI version of STOCK LEVEL transaction in TPC-C benchmark.
 +=================================================================*/

#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"

#define SQLTXT "SELECT count (DISTINCT s_i_id) \
          FROM order_line, stock, district \
          WHERE d_id = :d_id AND d_w_id = :w_id AND \
               d_id = ol_d_id AND d_w_id = ol_w_id AND \
               ol_i_id = s_i_id AND ol_w_id = s_w_id AND \
               s_quantity < :threshold AND \
               ol_o_id BETWEEN (d_next_o_id - 20) AND (d_next_o_id - 1)"


#define SQLTXTTEST "BEGIN stocklevel.getstocklevel (:w_id, :d_id, \
:threshold); END;"



plstoinit (ora_cn_data_t *ora_SlotDataP)

{

  stoctx *sctx;
  global_stock_t *stoP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  OCIError *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;

  text stmbuf[SQL_BUF_SIZE];

  sctx = (stoctx *)malloc(sizeof(stoctx));
  memset(sctx,(char)0,sizeof(stoctx));
  ora_SlotDataP->sctx = sctx;
  ora_SlotDataP->stoP = (global_stock_t *)malloc(sizeof(global_stock_t));
  memset(ora_SlotDataP->stoP,(char)0,sizeof(global_stock_t));
  stoP = ora_SlotDataP->stoP;

  sctx->norow=0;

  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&sctx->curs,OCI_HTYPE_STMT,0,(dvoid**)0));
  sprintf ((char *) stmbuf, SQLTXT);
  OCIERROR(errhp,OCIStmtPrepare(sctx->curs,errhp,stmbuf,strlen((char*)stmbuf),
           OCI_NTV_SYNTAX,OCI_DEFAULT));
  OCIERROR(errhp,
    OCIAttrSet(sctx->curs,OCI_HTYPE_STM,(dvoid*)&sctx->norow,0,
           OCI_ATTR_PREFETCH_ROWS,errhp));

  /* bind variables */

  OCIBND(sctx->curs,sctx->w_id_bp,errhp, ":w_id", ADR(stoP->w_id),sizeof(int),
       SQLT_INT);
  OCIBND(sctx->curs,sctx->d_id_bp,errhp, ":d_id", ADR(stoP->d_id),sizeof(int),
    SQLT_INT);
  OCIBND(sctx->curs,sctx->threshold_bp,errhp, ":threshold", ADR(stoP->threshold),
    sizeof(int),SQLT_INT);
  OCIDEFINE(sctx->curs,sctx->low_stock_bp,errhp, 1, ADR(stoP->low_stock),
    sizeof(int), SQLT_INT);

  return (0);

}



plsto (ora_cn_data_t *ora_SlotDataP)
{
  stoctx *sctx = ora_SlotDataP->sctx;
  global_stock_t *stoP = ora_SlotDataP->stoP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  OCIError *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;

retry:
  stoP->execstatus=
    OCIStmtExecute(tpcsvc,sctx->curs,errhp,1,0,0,0,
      OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
  if(stoP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    stoP->errcode = OCIERROR(errhp,stoP->execstatus);
    if(stoP->errcode == NOT_SERIALIZABLE){
      stoP->retries++;
      goto retry;
    } else if (stoP->errcode == RECOVERR) {
      stoP->retries++;
      goto retry;
    } else {
      return -1;
    }
  }
  return (0);
}



void plstodone (ora_cn_data_t *ora_SlotDataP)
{
  stoctx *sctx = ora_SlotDataP->sctx;
  if (sctx) {
    free(sctx);
    ora_SlotDataP->sctx = NULL;
  }
  if (ora_SlotDataP->stoP) {
    free(ora_SlotDataP->stoP);
    ora_SlotDataP->stoP = NULL;
  }
}
```

plsto.h

```
#ifndef TPCC_PLSTO_H
#define TPCC_PLSTO_H



struct stoctx {
  OCIStmt *curs;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *threshold_bp;
  OCIDefine *low_stock_bp;
```

```
  int norow;
};

typedef struct stoctx stoctx;

struct global_stock_t {
  int w_id;
  int d_id;
  int threshold;
  int retries;
  int low_stock;
  int errcode;
  int execstatus;
};

typedef struct global_stock_t global_stock_t;

#endif
```

tpcc.h

```
/*
 * $Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RCS/tpcc.h,v 1.2 1998/01/23 15:08:17 oz Exp $ Copyr (c) 1993 Oracle
 */
/*+=================================================================+
 |        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
 |                OPEN SYSTEMS PERFORMANCE GROUP                   |
 |                    All Rights Reserved                          |
 +=================================================================+
 | FILENAME
 |    tpcc.h
 | DESCRIPTION
 |    Include file for TPC-C benchmark programs.
 +=================================================================*/

#ifndef TPCC_H
#define TPCC_H


#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>
/*
#ifdef __STDC__
#include "ociapr.h"
#else
#include "ocikpr.h"
#endif
*/

typedef struct cda_def csrdef;
typedef struct cda_def ldadef;


/* TPC-C transaction functions */

extern int TPCinit ();
extern int TPCnew ();
extern int TPCpay ();
extern int TPCord ();
extern int TPCdel ();
extern int TPCsto ();
extern int TPCexit ();
extern int TPCdumpinit ();
extern int TPCdumpnew ();
extern int TPCdumppay ();
extern int TPCdumpord ();
extern int TPCdumpdel ();
extern int TPCdumpsto ();
extern int TPCdumpexit ();


/* Error codes */

#define RECOVERR -10
#define IRRECERR -20
#define NOERR    111
#define DEL_ERROR -666
#define DEL_DATE_LEN 7
#define NDISTS 10
#define NITEMS 15
#define SQL_BUF_SIZE 8192
#endif
```

tpcc_info.h

```
/*
 * $Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RCS/tpcc_info.h,v 1.2 1998/01/23 15:08:18 oz Exp $ Copyr (c) 1995 Oracle
 */
/*+=================================================================+
 |        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
 |                OPEN SYSTEMS PERFORMANCE GROUP                   |
 |                    All Rights Reserved                          |
 +=================================================================+
 | FILENAME
 |    tpcc_info.h
 | DESCRIPTION
 |    Include file for TPC-C benchmark programs.
 +=================================================================*/

#ifndef TPCC_INFO_H
#define TPCC_INFO_H

/* New order */

struct newinstruct {
   int w_id;
   int d_id;
   int c_id;
   int ol_i_id[15];
   int ol_supply_w_id[15];
   int ol_quantity[15];
};

struct newoutstruct {
   int terror;
   int o_id;
   int o_ol_cnt;
   char c_last[17];
   char c_credit[3];
   float c_discount;
   float w_tax;
   float d_tax;
   char o_entry_d[20];
   float total_amount;
   char i_name[15][25];
   int s_quantity[15];
   char brand_generic[15];
   float i_price[15];
   float ol_amount[15];
   char status[26];
   int retry;
};

struct newstruct {
   struct newinstruct newin;
   struct newoutstruct newout;
};


/* Payment */

struct payinstruct {
   int w_id;
   int d_id;
   int c_w_id;
   int c_d_id;
   int c_id;
   int bylastname;
   int h_amount;
   char c_last[17];
};

struct payoutstruct {
   int terror;
   char w_street_1[21];
   char w_street_2[21];
   char w_city[21];
   char w_state[3];
   char w_zip[10];
   char d_street_1[21];
   char d_street_2[21];
   char d_city[21];
   char d_state[3];
   char d_zip[10];
   int  c_id;
   char c_first[17];
   char c_middle[3];
   char c_last[17];
   char c_street_1[21];
   char c_street_2[21];
   char c_city[21];
   char c_state[3];
   char c_zip[10];
   char c_phone[17];
   char c_since[11];
```

```
  char c_credit[3];
  double c_credit_lim;
  float c_discount;
  double c_balance;
  char c_data[201];
  char h_date[20];
  int retry;
};

struct paystruct {
  struct payinstruct payin;
  struct payoutstruct payout;
};


/* Order status */

struct ordinstruct {
  int w_id;
  int d_id;
  int c_id;
  int bylastname;
  char c_last[17];
};

struct ordoutstruct {
  int terror;
  int c_id;
  char c_last[17];
  char c_first[17];
  char c_middle[3];
  double c_balance;
  int o_id;
  char o_entry_d[20];
  int o_carrier_id;
  int o_ol_cnt;
  int ol_supply_w_id[15];
  int ol_i_id[15];
  int ol_quantity[15];
  float ol_amount[15];
  char ol_delivery_d[15][11];
  int retry;
};

struct ordstruct {
  struct ordinstruct ordin;
  struct ordoutstruct ordout;
};


/* Delivery */

struct delinstruct {
  int w_id;
  int o_carrier_id;
  double qtime;
  int in_timing_int;
};

struct deloutstruct {
  int terror;
  int retry;
};

struct delstruct {
  struct delinstruct delin;
  struct deloutstruct delout;
};


/* Stock level */

struct stoinstruct {
  int w_id;
  int d_id;
  int threshold;
};

struct stooutstruct {
  int terror;
  int low_stock;
  int retry;
};

struct stostruct {
  struct stoinstruct stoin;
  struct stooutstruct stoout;
};

#endif
```

tpccpl.c

```
#ifdef RCSID
static char *RCSid =
  "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/tpccpl.c,v 1.3 1998/01/24 14:17:05 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*  For now: Preallocate all the connections that we will need  */
#if 1
#define PREALLOC_CN
#endif

/*+-----------------------------------------------------------------+
 |       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
 |                 OPEN SYSTEMS PERFORMANCE GROUP                  |
 |                    All Rights Reserved                          |
 +-----------------------------------------------------------------+
 | FILENAME
 |    tpccpl.c
 | DESCRIPTION
 |    TPC-C transactions in PL/SQL.
 +-----------------------------------------------------------------+*/

#ifdef ENCINA
#include <dce/pthread.h>
#define TRACE_WITHOUT_TPP 1
#include <utils/trace.h>
#endif
#include <stdio.h>
#include <time.h>
#include "tpcc.h"
#include "tpcc_info.h"
#include "tpccpl.h"
#include "plora.h"

#define SQLTXT "alter session set isolation_level = serializable"
#ifdef SQL_TRACE
#define SQLTXT1 "alter session set sql_trace = true"
#endif

void print_rt_avg(total_tran_count_t *curP, total_tran_count_t *prevP);
static void check_threads(total_tran_count_t *tran_ctP);

#define PRINT_AV(total, num, str)                   \
{                                                   \
   if ((num) > 0) {                                 \
      fprintf(stderr, " %s %.3f,",str, (double)(total)/(num));    \
   }                                                \
}

/*  ORA_RECONNECT_THRESHHOLD:
 *     Try reconnecting after this many consecutive errors.
 *  MIN_TIME_BETWEEN_RECONNECTS
 *     Min time in seconds to elapse between the last connection time before
 *     we are allowed to try and reconnect again.
 */
#define ORA_RECONNECT_THRESHHOLD       20
#define MIN_TIME_BETWEEN_RECONNECTS    20
#ifdef ENCINA
/* thread slot data that contains OCI handles and thread specific vars */
pthread_key_t thread_key;
int key_init = 0;
pthread_mutex_t key_lock;
pthread_mutex_t init_lock;
pthread_mutex_t dvry_log_lock;
#else
ora_cn_data_t *connectionP = NULL;
#endif
static char delivery_file_name[80];

FILE *lfp;
FILE *fopen ();
double gettime ();
int proc_no = 0;
char *TPC_uid, *TPC_pwd;

static void init_global_data(void);
#ifdef ENCINA
static ora_cn_data_t *get_cn(int tran, void *dataP);
static void done_with_cn(ora_cn_data_t *cnP, int tran, void *dataP);
#ifdef PREALLOC_CN
ora_cn_data_t *cn_array = NULL;
int num_connections = 0;
int cn_id = 0;
#endif
#else
#define get_cn(a,b) connectionP
#define done_with_cn(c,a,b) connectionP
#endif
static void init_cn_data(ora_cn_data_t *dataP);
static void clean_cn(void *ptr);

errrpt (lda, cur)

ldadef *lda;
```

```
csrdef *cur;

{

  text msg[2048];

  if (cur->rc) {
     oerhms (lda, cur->rc, msg, 2048);
     fprintf (stderr, "Error in TPC-C server %d: %s\n", proc_no, msg);
  }
  if ((cur->rc == DEADLOCK) || (cur->rc == SNAPSHOT_TOO_OLD))
     return (RECOVERR);
  else
     return (IRRECERR);

}


/* vmm313 void ocierror(fname, lineno, errhp, status)  */
int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
  text errbuf[512];
  ub4 buflen;
  sb4 errcode;

  switch (status) {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_SUCCESS_WITH_INFO\n");
  break;
  case OCI_NEED_DATA:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_NEED_DATA\n");
  break;
  case OCI_NO_DATA:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_NO_DATA\n");
    return IRRECERR;
  break;
  case OCI_ERROR:
    (void) OCIErrorGet (errhp, (ub4) 1,
                (text *) NULL, &errcode, errbuf,
                    (ub4) sizeof(errbuf), OCI_HTYPE_ERR);
    if (errcode != 8177)
    {
       (void) err_printf("Module %s Line %d\n", fname, lineno);
       (void) err_printf("Warning - %s\n", errbuf);
    }
    if ( errcode == SNAPSHOT_TOO_OLD ) {
      return RECOVERR;
    }
    if ( errcode == DEADLOCK ) {
      return RECOVERR;
    }
    return errcode;
/* vmm313    TPCexit(1); */
/* vmm313    exit(1); */
  break;
  case OCI_INVALID_HANDLE:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_INVALID_HANDLE\n");
    TPCexit(1);
    exit(-1);
  break;
  case OCI_STILL_EXECUTING:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_STILL_EXECUTE\n");
  break;
  case OCI_CONTINUE:
    (void) err_printf("Module %s Line %d\n", fname, lineno);
    (void) err_printf("Error - OCI_CONTINUE\n");
  break;
  default:
    break;
  }
  return RECOVERR;
}


FILE  *vopen(fnam,mode)
char  *fnam;
char  *mode;
{
FILE *fd;

#ifdef  DEBUG
  fprintf(stderr, "tkvuopen() fnam: %s,   mode:%s\n", fnam, mode);
#endif

    fd = fopen((char *)fnam,(char *)mode);
    if (!fd){
       fprintf(stderr," fopen on %s failed %d\n",fnam,fd);
       exit(-1);
    }
    return(fd);
}

int sqlfile(fnam,linebuf)
char   *fnam;
text   *linebuf;
{
FILE *fd;
int nulpt = 0;

#ifdef  DEBUG
  fprintf(stderr, "sqlfile() fnam: %s,  linebuf: %#x\n", fnam, linebuf);
#endif

    fd = vopen(fnam,"r");
    while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
    {
       nulpt = strlen((char *)linebuf);
    }
    return(nulpt);
}

/*
void
vgetdate(unsigned char *buf) {
  time_t tloc;
  char temp[5];
  int cen;

  if(time(&tloc) == (time_t)-1) {
    err_printf("Error getting date\n");
    exit(1);
  }
  cftime(temp,"%d",&tloc);
  buf[3] = (unsigned char)atoi(temp);
  cftime(temp,"%m",&tloc);
  buf[2] = (unsigned char)atoi(temp);
  cftime(temp,"%Y",&tloc);
  cen = atoi(temp);
  buf[0] = (unsigned char)((cen/100)+100);
  buf[1] = (unsigned char)((cen%100)+100);
  cftime(temp,"%H",&tloc);
  buf[4] = (unsigned char)(atoi(temp) + 1);
  cftime(temp,"%M",&tloc);
  buf[5] = (unsigned char)(atoi(temp) + 1);
  cftime(temp,"%S",&tloc);
  buf[6] = (unsigned char)(atoi(temp) + 1);
}
*/
void vgetdate (unsigned char *oradt)
{
 struct tm timebuf;
 struct tm *loctime = &timebuf;
 time_t  int_time;

 struct ORADATE {
 unsigned char  century;
 unsigned char  year;
 unsigned char  month;
 unsigned char  day;
 unsigned char  hour;
 unsigned char  minute;
 unsigned char  second;
 } Date;
int century;
int cnvrtOK;

/* assume convert is successful */
cnvrtOK = 1;

/* get the current date and time as an integer */
time( &int_time);

/* Convert the current date and time into local time */
localtime_r(&int_time, &timebuf);

century = (1900+loctime->tm_year) / 100;

Date.century = (unsigned char)(century + 100);
if (Date.century < 119 || Date.century > 120) cnvrtOK = 0;
Date.year  = (unsigned char)(loctime->tm_year+100);
if (Date.year < 100 || Date.year > 199) cnvrtOK = 0;
Date.month = (unsigned char)(loctime->tm_mon + 1);
if (Date.month < 1 || Date.month > 12) cnvrtOK = 0;
Date.day   = (unsigned char)loctime->tm_mday;
if (Date.day < 1 || Date.day > 31) cnvrtOK = 0;
Date.hour  = (unsigned char)(loctime->tm_hour + 1);
if (Date.hour < 1 || Date.hour > 24) cnvrtOK = 0;
Date.minute= (unsigned char)(loctime->tm_min + 1);
if (Date.minute < 1 || Date.minute > 60) cnvrtOK = 0;
```

```
Date.second= (unsigned char)(loctime->tm_sec + 1);
if (Date.second < 1 || Date.second > 60) cnvrtOK = 0;

if (cnvrtOK)
  memcpy(oradt,&Date,7);
  else
    *oradt = '\0';

 return;

}
void cvtdmy (unsigned char *oradt, char *outdate)
{

       struct ORADATE {
             unsigned char   century;
             unsigned char   year;
             unsigned char   month;
             unsigned char   day;
             unsigned char   hour;
             unsigned char   minute;
             unsigned char   second;
       } Date;


       int day,month,year;

       memcpy(&Date,oradt,7);

       year = (Date.century-100)*100 + Date.year-100;
       month = Date.month;
       day = Date.day;
       /* sprintf(outdate,"%02d-%02d-%4d\0",day,month,year);*/
       sprintf(outdate,"%02d-%02d-%4d",day,month,year);

       return;
}

void cvtdmyhms (unsigned char *oradt, char *outdate)
{

       struct ORADATE {
             unsigned char   century;
             unsigned char   year;
             unsigned char   month;
             unsigned char   day;
             unsigned char   hour;
             unsigned char   minute;
             unsigned char   second;
       } Date;


       int day,month,year;
       int hour,min,sec;

       memcpy(&Date,oradt,7);

       year = (Date.century-100)*100 + Date.year-100;
       month = Date.month;
       day = Date.day;
       hour = Date.hour - 1;
       min = Date.minute - 1;
       sec = Date.second - 1;

       /*sprintf(outdate,"%02d-%02d-%4d%02d:%02d:%02d\0";*/
       sprintf(outdate,"%02d-%02d-%4d%02d:%02d:%02d",
                day,month,year,hour,min,sec);

       return;
}

/*  Each server may have multiple connections to the DB.
 *  The OCI handles, which used to be global, are now grouped together
 *  In a data structure.  There is one such structure per DB connection.
 *
 *  There are two routines to deal with them:
 *      initOCIhandles: Initializes all the handles a connection needs
 *      freeOCIhandles: Frees all those handles.
 *  When the program is initialized it initializes an array of connections.
 *  Each thread is then assigned one of these connections and keeps reusing
 *  that connection.  The same connection may be shared by multiple threads.
 */
static void initOCIhandles(ora_cn_data_t *cn_dataP)
{
  text stmbuf[SQL_BUF_SIZE];
  OCIEnv *tpcenv;
  OCIServer *tpcsrv;
  OCIError *errhp;
  OCISvcCtx *tpcsvc;
  OCISession *tpcusr;
  OCIStmt *curi;
  dvoid *xmem;


  /* Initialize OCI handles in thread slot data.
   * This is called once per thread.
   * Specify that OCI library should not handle mutexing
   */
  OCIEnvInit(&tpcenv, OCI_ENV_NO_MUTEX, 0, (dvoid **)0);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid **)0);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid **)0);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid **)0);
  OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);

  OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SRVRCTXT, errhp);

  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)TPC_uid,
(ub4)strlen(TPC_uid),OCI_ATTR_USERNAME, errhp);
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)TPC_pwd,
(ub4)strlen(TPC_pwd), OCI_ATTR_PASSWORD, errhp);
  OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));
  OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_USERCTXT, errhp);

  /* run all transaction in serializable mode */
  OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);

  sprintf ((char *) stmbuf, SQLTXT);
  OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT);

  OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));

  OCIHandleFree(curi, OCI_HTYPE_STMT);
#ifdef SQL_TRACE
  /* Turn on the SQL_TRACE */
  OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, &xmem);
  sprintf ((char *) stmbuf, SQLTXT1);
  OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT);
  OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
  OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
#endif /* End SQL_TRACE */

  /*  Store the handles just initialized in the thread slot
   */
  cn_dataP->tpcenv = tpcenv;
  cn_dataP->tpcsrv = tpcsrv;
  cn_dataP->errhp = errhp;
  cn_dataP->tpcsvc = tpcsvc;
  cn_dataP->tpcusr = tpcusr;
  cn_dataP->curi = curi;
  cn_dataP->xmem = xmem;
}


static void freeOCIhandles(ora_cn_data_t *cn_dataP)
{

 OCIServer *tpcsrv;
 OCISession *tpcusr;
 OCIEnv *tpcenv;
 OCIError *errhp;
 OCISvcCtx *tpcsvc;

 if (tpcusr = cn_dataP->tpcusr) {
   err_printf("free_handles> OCIHandleFree tpcusr\n");
   OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
 }
 if (tpcsvc = cn_dataP->tpcsvc) {
   err_printf("free_handles> OCIHandleFree tpcsvc\n");
   OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
 }
 if (errhp = cn_dataP->errhp) {
   err_printf("free_handles> OCIHandleFree errhp\n");
   OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
 }
 if (tpcsrv = cn_dataP->tpcsrv) {
   err_printf("free_handles> OCIHandleFree tpcsrv\n");
   OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
 }
 if (tpcenv = cn_dataP->tpcenv) {
   err_printf("free_handles> OCIHandleFree tpcenv\n");
   OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);
 }
}

TPCexit ()
{
  if (lfp) {
    fclose (lfp);
    lfp = NULL;
  }
#ifndef ENCINA
  clean_cn((void *)connectionP);
  connectionP = NULL;
#endif
```

```
}

/*  clean_cn
 *
 *  Called to clean a connection.
 *  When using pthread this is registered during pthread_create
 *  and called automatically by pthread when the thread exits.
 */
static void clean_cn(void *ptr)
{

  /* free trans specific cursor handles first and later the ora handles */
  ora_cn_data_t *cn_dataP = (ora_cn_data_t *)ptr;

  if (cn_dataP != NULL) {
      err_printf("clean_cn, Freeing OCI handles\n");
      plnewdone(cn_dataP);
      plpaydone(cn_dataP);
      plorddone(cn_dataP);
      pldeldone(cn_dataP);
      plstodone(cn_dataP);

      freeOCIhandles(cn_dataP);

      err_printf("free_handles> free cn_dataP\n");
  }
}

static char *thread_state_to_str(int state)
{
   char *retval;
   switch(state) {
      case SVR_STATE_NONE: retval = "None"; break;
      case SVR_STATE_SENT: retval = "Sent"; break;
      case SVR_STATE_REPLIED: retval = "Replied"; break;
      case SVR_STATE_ERR: retval = "Err"; break;
      default: retval = "unknown"; break;
   }
   return retval;
}


void print_rt_avg(total_tran_count_t *curP, total_tran_count_t *prevP)
{
   int i;
   double avg_queued_time;
   static char *names[] = {"id", "no", "pa", "os", "dl", "sl", "dn", "dp"};
   err_printf("bg_thread RT avg: ");

   for (i=0; i<=MAX_TRAN_TYPE; i++) {
      int num_trans = curP->tran[i].num - prevP->tran[i].num;
      double rt_diff = curP->tran[i].RT - prevP->tran[i].RT;
      PRINT_AV(rt_diff, num_trans, names[i]);
      if (i == DELIVERY_TRANS && num_trans > 0) {
         rt_diff = curP->dvry_queue_time - prevP->dvry_queue_time;
         PRINT_AV(rt_diff, num_trans, "qd");
      }
   }
   fprintf(stderr, "\n");
}

void print_tran_info(tran_spec_info_t *tran_copy)
{
   int i;
   struct payinstruct *paP;
   struct newinstruct *noP;

   switch (tran_copy->tran_type) {
      case DIST_NEWO_TRANS:
      case NEWO_TRANS:
         noP = &(tran_copy->tran_info.no.newin);
         err_printf("bg_thread: TPCnew : w_id %d, d_id %d, c_id %d\n",
           noP->w_id,
           noP->d_id,
           noP->c_id);
         for (i=0; i<15; i++) {
            if (noP->ol_i_id[i] == 0) {
               break;
            }
            err_printf("bg_thread TPCnew : (%d)i_id %d sup_w_id %d Qty%d\n",
                  i,
                  noP->ol_i_id[i],
                  noP->ol_supply_w_id[i],
                  noP->ol_quantity[i]);
         }
           break;

      case DIST_PAY_TRANS:
      case PAYMENT_TRANS:
          paP = &(tran_copy->tran_info.pa.payin);
          err_printf("bg_thread: TPCpay: w_id %d, D_id %d, C_w_id %d, c_id %d,
bylastname %d, amount %.2f, c_last %s \n",
             paP->w_id,
             paP->d_id,
             paP->c_w_id,
             paP->c_id,
             paP->bylastname,
             paP->h_amount,
             paP->c_last);
             break;

      case ORDER_STAT_TRANS:
          err_printf("bg_thread: TPCord: w_id %d, d_id %d, c_id %d, bylastname %d, c_last
%s \n",
             tran_copy->tran_info.os.ordin.w_id,
             tran_copy->tran_info.os.ordin.d_id,
             tran_copy->tran_info.os.ordin.c_id,
             tran_copy->tran_info.os.ordin.bylastname,
             tran_copy->tran_info.os.ordin.c_last);
             break;

       case DELIVERY_TRANS:
          err_printf("bg_thread: TPCdel: w_id %d, o_carrier_id %d, %fqtime, %d
in_timing_int\n",
             tran_copy->tran_info.dl.delin.w_id,
             tran_copy->tran_info.dl.delin.o_carrier_id,
             tran_copy->tran_info.dl.delin.qtime,
             tran_copy->tran_info.dl.delin.in_timing_int);
             break;

       case STOCK_TRANS:
          err_printf("bg_thread: TPCsto: w_id %d, d_id %d, threashold %d\n",
             tran_copy->tran_info.sl.stoin.w_id,
             tran_copy->tran_info.sl.stoin.d_id,
             tran_copy->tran_info.sl.stoin.threshold);
             break;
       default:
            err_printf("bg_thread: bad tran\n");
             break;
       }
}

static void copy_tran_info(void *dataP, int type, tran_spec_info_t *outP)
{
   outP->tran_type = type;
   switch (type) {
      case NEWO_TRANS:
      case DIST_NEWO_TRANS:
         outP->tran_info.no = *(struct newstruct *)dataP;
         break;
      case PAYMENT_TRANS:
      case DIST_PAY_TRANS:
         outP->tran_info.pa = *(struct paystruct *)dataP;
         break;
      case ORDER_STAT_TRANS:
         outP->tran_info.os = *(struct ordstruct *)dataP;
         break;
      case STOCK_TRANS:
         outP->tran_info.sl = *(struct stostruct *)dataP;
         break;
      case DELIVERY_TRANS:
         outP->tran_info.dl = *(struct delstruct *)dataP;
         break;
      defaut:
         memset(&outP->tran_info, '\0', sizeof(outP->tran_info));
         break;
   }
}

static void check_threads(total_tran_count_t *tran_ctP)
{
   struct timezone tz;
   int num_per_state[NUM_STATES];
   int total_stuck = 0;
   static int init_printed = 0;
   int total_tran_err;
   tran_spec_info_t tran_copy;

   pthread_mutex_lock(&init_lock);

   if (cn_array && (num_connections > 0)) {
      int i,j;

      memset(num_per_state, '\0', sizeof(num_per_state));
      memset(tran_ctP, '\0', sizeof(*tran_ctP));
      for (i=0; i<num_connections; i++) {
         struct timeval cur_time;
         struct timeval *client_timeP;
         ora_cn_data_t *cnP = &cn_array[i];
         int time_diff;
         int delta = 60;
         total_tran_count_t *statP = &cnP->stat;

         for (j=0; j<=MAX_TRAN_TYPE; j++) {
            tran_ctP->tran[j].num += statP->tran[j].num;
            tran_ctP->tran[j].errs += statP->tran[j].errs;
            tran_ctP->tran[j].RT += statP->tran[j].RT;
            tran_ctP->errors += statP->tran[j].errs;
         }
         tran_ctP->dvry_queue_time += statP->dvry_queue_time;
```

```
        /*  Make a copy of the tran specifc data structure here
         *  Since we are not performing any locking or any other
         *  synchronization, we have to be careful: Copy the tran
         *  data before getting the current time.  If the tran
         *  has been stuck for a while, it is likely that
         *  the copy is good.
         */
        copy_tran_info(cnP->cur_tran_dataP, cnP->cur_tran_type, &tran_copy);
        gettimeofday(&cur_time,&tz);

        time_diff = cur_time.tv_sec - cnP->tran_time.tv_sec;
        if (time_diff > delta) {
           num_per_state[cnP->state] ++;
           total_stuck++;
           if (!cnP->printed) {
              err_printf("bg_thread: thread index %d state %s tran %d stuck for %d sec\n",
                      i,
                      thread_state_to_str(cnP->state),
                      cnP->cur_tran_type,
                      time_diff);
              /*  Print some tran specific info here off the copy  */
              if (cnP->state == SVR_STATE_SENT)
                 print_tran_info(&tran_copy);
              cnP->printed = 1;
           }
        } else if (cnP->printed) {
           err_printf("bg_thread: thread index %d state %s tran %d unstuck.\n",
                      i,
                      thread_state_to_str(cnP->state),
                      cnP->cur_tran_type);

           cnP->printed = 0;
        }
     }
     if (total_stuck > 0) {
        err_printf("bg_thread: Summary %d stuck: ",total_stuck);
        for (i=0; i<NUM_STATES; i++) {
           if (num_per_state[i] > 0) {
              fprintf(stderr, "%d %s, ",
                      num_per_state[i], thread_state_to_str(i));
           }
        }
        fprintf(stderr, "\n");
     }
     total_tran_err = 0;
     for (i=0; i<=MAX_TRAN_TYPE; i++)
        total_tran_err += tran_ctP->tran[i].errs;
     if (total_tran_err > 0) {
        err_printf("bg_thread: %d errs: %d no, %d pa, %d os, %d sl\n",
                total_tran_err,
                tran_ctP->tran[NEWO_TRANS].errs,
                tran_ctP->tran[PAYMENT_TRANS].errs,
                tran_ctP->tran[ORDER_STAT_TRANS].errs,
                tran_ctP->tran[STOCK_TRANS].errs);
     }
   }
   pthread_mutex_unlock(&init_lock);
}

/*
 *  time_diff_ms
 *     Return the difference in miliseconds between two times
 */
int time_diff_ms(t2,t1)
  struct timeval *t2, *t1;
{
   int t_diff;

   t_diff = (t2->tv_usec + 1000000 - t1->tv_usec + 500) / 1000 +
          (t2->tv_sec - t1->tv_sec - 1) * 1000;

   return(t_diff);
}

/*
 *  A background thread that keeps tabs on the state of all the
 *  threads of the server.  (For Debug)
 */
static void *bg_thread(void *argP)
{
   int i;
   int total_newo, total_tran_err;
   int total_trans;
   struct timeval cur_time;
   struct timezone tz;
   struct timeval time_reported[2];
   total_tran_count_t tran_ct, tran_reported[2];

   gettimeofday(&time_reported[0],&tz);
   time_reported[1] = time_reported[0];

   memset(&tran_reported[0], '\0', 2 * sizeof(tran_reported[0]));

   while (1) {
      double time_diff1, time_diff2;
      double tran_diff1, tran_diff2;
      double err_diff1, err_diff2;

      check_threads(&tran_ct);

      total_tran_err = tran_ct.errors;
      total_newo = tran_ct.tran[NEWO_TRANS].num;
      total_trans = 0;
      for (i=0; i<=MAX_TRAN_TYPE; i++) total_trans += tran_ct.tran[i].num;

      gettimeofday(&cur_time,&tz);
      time_diff1 = time_diff_ms(&cur_time,&time_reported[0]);
      tran_diff1 = total_newo - tran_reported[0].tran[NEWO_TRANS].num;
      err_diff1 = total_tran_err - tran_reported[0].errors;
      time_diff2 = time_diff_ms(&cur_time,&time_reported[1]);
      tran_diff2 = total_newo - tran_reported[1].tran[NEWO_TRANS].num;
      err_diff2 = total_tran_err - tran_reported[1].errors;
      if (total_trans != 0) {
         err_printf("bg_thread: TPM: %.0f (last %.0f sec), %.0f (last %.0f sec)\n",
               tran_diff1 / time_diff1 * 60000, time_diff1 / 1000.,
               tran_diff2 / time_diff2 * 60000, time_diff2 / 1000.);
         /* print av server response time for all transactions */
         print_rt_avg(&tran_ct, &tran_reported[1]);
      }
      if (err_diff2 != 0) {
         err_printf("bg_thread: errPM %.1f (last %.0f sec)\n",
                  err_diff2 / time_diff2 * 60000, time_diff2 / 1000.);
      }
      tran_reported[0] = tran_reported[1];
      tran_reported[1] = tran_ct;
      time_reported[0] = time_reported[1];
      time_reported[1] = cur_time;
      sleep(60);
   }
   pthread_mutex_lock(&init_lock);
   pthread_mutex_unlock(&init_lock);
}

void start_bg_thread( void )
{
   int rc;
   pthread_attr_t attr;
   pthread_t thread;

   err_printf("> start_bg_thread\n");
   if (rc = pthread_attr_create(&attr)) {
      err_printf("start_bg_debug_thread: pthread_attr_create failed: %d\n", rc);
      return;
   }
   UNCOND_EVENT("Creating thread for bg_thread");
   if ((rc = pthread_create(&thread,
                        attr,
                        bg_thread,
                        (pthread_addr_t)NULL)) != 0) {
      err_printf("start_bg_debug_thread: pthread_create failed: %d\n", rc);
      return;
   }
   UNCOND_EVENT("Detaching bg thread");
   if ( rc = pthread_detach(&thread) != 0) {
      err_printf("start_bd_debug_thread: pthread_detach failed %d\n", rc);
      return;
   }
   err_printf("< start_bg_thread\n");
}



/*
 *  init_global_data
 *
 *  Called once during initialization to initialize the (thread) global data:
 *     OCI Handles
 *     Transaction data structures and cursors
 */
static void init_global_data(void)
{
#ifdef ENCINA
  int status = 0;
  pthread_mutex_init(&dvry_log_lock, pthread_mutexattr_default);
  pthread_mutex_init(&key_lock, pthread_mutexattr_default);
  pthread_mutex_init(&init_lock, pthread_mutexattr_default);
  err_printf("init_global_data> grabbing pthread_mutex_lock\n");
  pthread_mutex_lock(&key_lock);
  if (!key_init) {
     if(status = pthread_keycreate(&thread_key, clean_cn)) {
        fprintf(stderr, "init_global_data : pthread_keycreate failed: %d\n", status);
        exit(20);
     }
     key_init = 1;
  }
  pthread_mutex_unlock(&key_lock);
#ifdef PREALLOC_CN
  {
    int i, env_val;
```

```
    char *env_str;
    extern int num_deferred_dvry_threads;
    extern int num_worker_threads;

    pthread_mutex_lock(&init_lock);
    if (num_worker_threads > 0 &&
        num_worker_threads < 200) {
        num_connections += num_worker_threads;
    }

    if (num_deferred_dvry_threads > 0 &&
        num_deferred_dvry_threads < 20) {
        num_connections += num_deferred_dvry_threads;
    }

    cn_array = (ora_cn_data_t *)calloc(num_connections, sizeof(ora_cn_data_t));
    err_printf("Preallocating %d connections tooracle\n", num_connections);
    if (cn_array == NULL) {
        err_printf("Failed to allocated %dentried for CN array\n",
                num_connections);
        exit(3);
    }
    for (i=0; i<num_connections; i++) {
        init_cn_data(&cn_array[i]);
    }
    pthread_mutex_unlock(&init_lock);
    start_bg_thread();
  }
#endif
#else
  connectionP = malloc(sizeof(*connectionP));
  memset(connectionP, (char)0, sizeof(*connectionP));
  init_cn_data(connectionP);
#endif
}

TPCinit (id, uid, pwd)
int id;
char *uid;
char *pwd;
{

    int i;
    extern char *tpcc_serverName;
    char *home_dir = getenv("HOME");

    err_printf("TPCinit id %d, uid %s pwd %s\n", id, uid, pwd);
    proc_no = id;
    sprintf (delivery_file_name,
            "%s/runs/deliveries/tpcc_%d.%s.del",
            home_dir ? home_dir : "/home/encina",
            proc_no, tpcc_serverName);
    lfp = NULL;  /*  The file will be opened on demand  */

    /* Using multithreaded Oracle clients, OCI is multithreaded mode. */
    OCIInitialize(OCI_THREADED,(dvoid *)0,0,0,0);

    TPC_uid = uid;
    TPC_pwd = pwd;

    init_global_data();

    return (0);
}

/*
 *  init_cn_data
 *      Called once for each thread to initialize the thread
 *      global data structure.
 */
static void init_cn_data(ora_cn_data_t *dataP)
{
    UNCOND_EVENT("init_cn_data: Initializing connection to DB.");
    initOCIhandles(dataP);

    UNCOND_EVENT("init_cn_data: plnewinit");
    plnewinit(dataP);
    UNCOND_EVENT("init_cn_data: plpayinit");
    plpayinit(dataP);
    UNCOND_EVENT("init_cn_data: plordinit");
    plordinit(dataP);
    UNCOND_EVENT("init_cn_data: pldelinit");
    pldelinit(dataP);
    UNCOND_EVENT("init_cn_data: plstoinit");
    plstoinit(dataP);
    UNCOND_EVENT("Initialized connection to DB.");
}

#ifdef ENCINA
/*
 * get_cn - Gets a connection to the DB.
 * Each thread is assigned a connection and keeps reusing it.
 *
 *  For debug: each connections contains some state about the
 *      thread which includes the time this call was made, the transaction
 *      being performed and some tran and response time stats.
 */
static ora_cn_data_t *get_cn( int tran, void *tran_dataP )
{
    ora_cn_data_t *dataP;
    struct timezone tz;
    struct timeval cur_time;

    /*  Get a connection structure.
     *  Each thread always uses the same connection.
     *  The first time the thread tries to talk to the DB it creates
     *  a connection, initializes it and stores it in a thread global
     *  data structure.
     */
    pthread_getspecific(thread_key, (pthread_addr_t *)&dataP);
    if (dataP == NULL) {  /* No connection assigned to this thread */
        gettimeofday(&cur_time,&tz);

        pthread_mutex_lock(&init_lock); /*  Initialize the connections one at a time */
        err_printf("get_cn> initializing thread slot\n");
#ifdef PREALLOC_CN
        if (cn_id >= num_connections) {
            err_printf("Too many threads, not enough connections\n");
            exit(3);
        }
        dataP = &cn_array[cn_id++];
#else
        dataP = (ora_cn_data_t *)malloc(sizeof(ora_cn_data_t));
        memset(dataP, (char)0, sizeof(*dataP));
        init_cn_data(dataP);
#endif
        dataP->connect_time = cur_time.tv_sec;
        dataP->calls = 0;
        err_printf("get_cn> initialized connection 0x%x\n", dataP);
        pthread_mutex_unlock(&init_lock);

        pthread_setspecific(thread_key, dataP);  /*  Store it  */

        err_printf("get_cn> initialized connection\n");
    }


    gettimeofday(&cur_time,&tz);

    /*  Keep track of how much time the thread is idle */
    if (dataP->state != SVR_STATE_NONE) {
        tran_info_t *statP = &dataP->stat.tran[0];
        double RT;
        RT = cur_time.tv_usec - dataP->tran_time.tv_usec;
        RT = RT / 1e6 + cur_time.tv_sec - dataP->tran_time.tv_sec;

        statP->RT += RT;
        statP->num++;
    }

    /*  Keep some state for debug  */
    dataP->state = SVR_STATE_SENT;
    dataP->cur_tran_type = tran;
    dataP->cur_tran_dataP = tran_dataP;
    dataP->tran_time = cur_time;
    dataP->calls++;
    return dataP;
}

/*
 * done_with_cn - Done with a conection - -keep stats -- FOR DEBUG ONLY --
 */
static void done_with_cn(ora_cn_data_t *dataP, int tran, void *tran_dataP )
{
    struct timezone tz;
    struct timeval cur_time;
    tran_info_t *statP = &dataP->stat.tran[tran];
    double RT;

    gettimeofday(&cur_time,&tz);

    RT = cur_time.tv_usec - dataP->tran_time.tv_usec;
    RT = RT / 1e6 + cur_time.tv_sec - dataP->tran_time.tv_sec;

    statP->RT += RT;
    statP->num++;
    dataP->tran_time = cur_time;
    dataP->state = SVR_STATE_REPLIED;
    if (tran == DELIVERY_TRANS && tran_dataP) {
        /*  This is a delivery transaction.
         *  keep track of the average time it spent on the queue
         */
        struct delstruct *str = (struct delstruct *)tran_dataP;
        double ct = cur_time.tv_usec;
        ct = ct / 1e6 + cur_time.tv_sec;
        dataP->stat.dvry_queue_time += (ct - str->delin.qtime);
    }

    if (RT > 45) {
        err_printf("STATS: Tran %d RT %.3f, cn: %d trans RT total %.3f avg %.3f\n",
```

```
            tran, RT, statP->num, statP->RT, statP->RT / statP->num);
  }
}

/*
 *  ora_cn_err
 *
 *  Called to keep track of errors with oracle connections and possibly
 *  reestablish a connection in case it is bad.
 */
static void ora_cn_err(ora_cn_data_t *dataP)
{
  struct timezone tz;
  struct timeval cur_time;
  gettimeofday(&cur_time,&tz);

  dataP->errors ++;
  if (dataP->calls-1 == dataP->calls_last_err)
    dataP->consecutive_errs ++;
  else
    dataP->consecutive_errs = 1;
  dataP->calls_last_err = dataP->calls;
  err_printf("ora_cn_err %d errors (%d consecutive) connected %d sec, %d calls, %d
cals_last_err\n",
          dataP->errors, dataP->consecutive_errs,
          cur_time.tv_sec - dataP->connect_time,
          dataP->calls, dataP->calls_last_err);
  if (dataP->consecutive_errs > ORA_RECONNECT_THRESHHOLD &&
    (cur_time.tv_sec - dataP->connect_time) > MIN_TIME_BETWEEN_RECONNECTS) {
    /*  This connection is not behaving, free it.
     *  The next time this thread needs a connection it will reconnect
     */
    err_printf("ora_cn_err: Giving up on the connection\n");
    clean_cn((void *)dataP);
    pthread_setspecific(thread_key, NULL);
  }
  dataP->state = SVR_STATE_ERR;
  dataP->tran_time = cur_time;
}
#endif

TPCnew (str)
struct newstruct *str;
{
 ora_cn_data_t *cn_dataP = get_cn(NEWO_TRANS, (void *)str);
 struct timeval cur_time;
 global_newOrder_t *newP = cn_dataP->globals;
 int i;

#if defined(ISO1) || defined(ISO7)
  int reread;
  char sdate[30];
#endif

  /*
   * copy from struct str to previous globals
   * and chnaged the globals to local vars */
  newP->w_id = str->newin.w_id;
  newP->d_id = str->newin.d_id;
  newP->c_id = str->newin.c_id;

  for (i = 0; i < 15; i++) {
    newP->nol_i_id[i] = str->newin.ol_i_id[i];
    newP->nol_supply_w_id[i] = str->newin.ol_supply_w_id[i];
    newP->nol_quantity[i] = str->newin.ol_quantity[i];
  }
  newP->retries = 0;
  vgetdate(newP->cr_date);

  if (str->newout.terror = plnew(cn_dataP)) {
    err_printf("plnew> returning, terror %d, retries %d\n",
            str->newout.terror, newP->retries);
    if (str->newout.terror != RECOVERR)
      str->newout.terror = IRRECERR;
    str->newout.retry = newP->retries;
    ora_cn_err(cn_dataP);
    return (-1);
  }

  /* fill in date for o_entry_d from time in beginning oftxn*/
  cvtdmyhms(newP->cr_date,newP->o_entry_d);

  str->newout.terror = NOERR;
  str->newout.o_id = newP->o_id;
  str->newout.o_ol_cnt = newP->o_ol_cnt;
  strncpy (str->newout.c_last, newP->c_last, 17);
  strncpy (str->newout.c_credit, newP->c_credit, 3);
  str->newout.c_discount = (float)(newP->c_discount)/10000;
  str->newout.w_tax = (float)(newP->w_tax)/10000;
  str->newout.d_tax = (float)(newP->d_tax)/10000;
  strncpy (str->newout.o_entry_d, newP->o_entry_d, 20);
  str->newout.total_amount = newP->total_amount;

  for (i = 0; i < newP->o_ol_cnt; i++) {
    strncpy (str->newout.i_name[i], newP->i_name[i], 25);
    str->newout.s_quantity[i] = newP->s_quantity[i];
    str->newout.brand_generic[i] = newP->brand_gen[i];
    str->newout.i_price[i] = (float)(newP->i_price[i])/100;
    str->newout.ol_amount[i] = (float)(newP->nol_amount[i])/100;
  }
  if (newP->status) {
    strcpy (str->newout.status, "Item number is not valid");
  } else {
    str->newout.status[0] = '\0';
  }
  str->newout.retry = newP->retries;
  done_with_cn(cn_dataP,
            newP->o_all_local ? NEWO_TRANS : DIST_NEWO_TRANS,
            (void *)str);
  return(0);
}



TPCpay (str)
struct paystruct *str;
{
  ora_cn_data_t *cn_dataP =
      get_cn(str->payin.w_id == str->payin.c_w_id ? PAYMENT_TRANS : DIST_PAY_TRANS,
          (void *)str);
  global_payment_t *payP = cn_dataP->payP;

  payP->w_id = str->payin.w_id;
  payP->d_id = str->payin.d_id;
  payP->c_w_id = str->payin.c_w_id;
  payP->c_d_id = str->payin.c_d_id;
  payP->h_amount = str->payin.h_amount;
  payP->bylastname = str->payin.bylastname;

  vgetdate(payP->cr_date);

  if (payP->bylastname) {
    payP->c_id = 0;
    strncpy (payP->c_last, str->payin.c_last, 17);
  }
  else {
    payP->c_id = str->payin.c_id;
    strcpy (payP->c_last, " ");
  }
  payP->retries = 0;

  if (str->payout.terror = plpay(cn_dataP)) {
    err_printf("plpay> returning, terror %d, retries %d\n",
            str->payout.terror, payP->retries);
    if (str->payout.terror != RECOVERR)
      str->payout.terror = IRRECERR;
    str->payout.retry = payP->retries;
    ora_cn_err(cn_dataP);
    return (-1);
  }

  cvtdmyhms(payP->cr_date,payP->h_date);
  cvtdmy(payP->c_since,payP->c_since_d);


  str->payout.terror = NOERR;
  strncpy (str->payout.w_street_1, payP->w_street_1, 21);
  strncpy (str->payout.w_street_2, payP->w_street_2, 21);
  strncpy (str->payout.w_city, payP->w_city, 21);
  strncpy (str->payout.w_state, payP->w_state, 3);
  strncpy (str->payout.w_zip, payP->w_zip, 10);
  strncpy (str->payout.d_street_1, payP->d_street_1, 21);
  strncpy (str->payout.d_street_2, payP->d_street_2, 21);
  strncpy (str->payout.d_city, payP->d_city, 21);
  strncpy (str->payout.d_state, payP->d_state, 3);
  strncpy (str->payout.d_zip, payP->d_zip, 10);
  str->payout.c_id = payP->c_id;
  strncpy (str->payout.c_first, payP->c_first, 17);
  strncpy (str->payout.c_middle, payP->c_middle, 3);
  strncpy (str->payout.c_last, payP->c_last, 17);
  strncpy (str->payout.c_street_1, payP->c_street_1, 21);
  strncpy (str->payout.c_street_2, payP->c_street_2, 21);
  strncpy (str->payout.c_city, payP->c_city, 21);
  strncpy (str->payout.c_state, payP->c_state, 3);
  strncpy (str->payout.c_zip, payP->c_zip, 10);
  strncpy (str->payout.c_phone, payP->c_phone, 17);
  strncpy (str->payout.c_since, payP->c_since_d, 11);
  strncpy (str->payout.c_credit, payP->c_credit, 3);
  str->payout.c_credit_lim = (float)(payP->c_credit_lim)/100;
  str->payout.c_discount = (float)(payP->c_discount)/10000;
  str->payout.c_balance = (float)(payP->c_balance)/100;
  strncpy (str->payout.c_data, payP->c_data, 201);
  strncpy (str->payout.h_date, payP->h_date, 20);
  str->payout.retry = payP->retries;
  done_with_cn(cn_dataP,
            str->payin.w_id == str->payin.c_w_id ? PAYMENT_TRANS : DIST_PAY_TRANS,
            (void *)str);
  return (0);

}
```

```
TPCord (str)
struct ordstruct *str;
{
  ora_cn_data_t *cn_dataP = get_cn(ORDER_STAT_TRANS, (void *)str);
  global_order_t *ordP = cn_dataP->ordP;
  int i;


  ordP->w_id = str->ordin.w_id;
  ordP->d_id = str->ordin.d_id;
  ordP->bylastname = str->ordin.bylastname;
  if (ordP->bylastname) {
    ordP->c_id = 0;
    strncpy (ordP->c_last, str->ordin.c_last, 17);
  }
  else {
    ordP->c_id = str->ordin.c_id;
    strcpy (ordP->c_last, " ");
  }
  ordP->retries = 0;

  if (str->ordout.terror = plord (cn_dataP)) {
    err_printf("plord> returning, terror %d, retries %d\n",
          str->ordout.terror, ordP->retries);
    if (str->ordout.terror != RECOVERR)
      str->ordout.terror = IRRECERR;
    str->ordout.retry = ordP->retries;
    ora_cn_err(cn_dataP);
    return (-1);
  }

  str->ordout.terror = NOERR;
  str->ordout.c_id = ordP->c_id;
  strncpy (str->ordout.c_last, ordP->c_last, 17);
  strncpy (str->ordout.c_first, ordP->c_first, 17);
  strncpy (str->ordout.c_middle, ordP->c_middle, 3);
  str->ordout.c_balance = ordP->c_balance/100;
  str->ordout.o_id = ordP->o_id;
  strncpy (str->ordout.o_entry_d, ordP->o_entry_d, 20);
  if ( ordP->o_carrier_id == 11 )
     str->ordout.o_carrier_id = 0;
  else
     str->ordout.o_carrier_id = ordP->o_carrier_id;
  str->ordout.o_ol_cnt = ordP->o_ol_cnt;
  for (i = 0; i < ordP->o_ol_cnt; i++) {
    ordP->ol_delivery_d[i][10] = '\0';
    if ( !strcmp(ordP->ol_delivery_d[i],"15-09-1911") )
       strncpy(ordP->ol_delivery_d[i],"NOT DELIVR",10);
    str->ordout.ol_supply_w_id[i] = ordP->ol_supply_w_id[i];
    str->ordout.ol_i_id[i] = ordP->ol_i_id[i];
    str->ordout.ol_quantity[i] = ordP->ol_quantity[i];
    str->ordout.ol_amount[i] = (float)(ordP->ol_amount[i])/100;
    strncpy (str->ordout.ol_delivery_d[i], ordP->ol_delivery_d[i], 11);
  }
  str->ordout.retry = ordP->retries;
  done_with_cn(cn_dataP, ORDER_STAT_TRANS, (void *)str);
  return (0);

}



TPCdel (str)
struct delstruct *str;
{
  ora_cn_data_t *cn_dataP = get_cn(DELIVERY_TRANS, (void *)str);
  global_delivery_t *delP = cn_dataP->delP;
  double tr_end, tr_begin;
  int i, skipped;
  struct timeval cur_time;
  static int tran_cntr=0;
  int pos, len;
  int queue_time, start_time, end_time;
  char stdout_buf[1024];

  /*  Open the delivery log file if needed  */
  if (lfp == NULL) {
    pthread_mutex_lock(&dvry_log_lock);
    if (lfp == NULL) {
      if ((lfp = fopen (delivery_file_name, "w")) == NULL) {
        fprintf (stderr, "Error in TPC-C server: Failed to open %s\n",
              delivery_file_name);
        pthread_mutex_unlock(&dvry_log_lock);
        done_with_cn(cn_dataP, DELIVERY_TRANS, (void *)str);
        return(-1);
      }
      err_printf("Opened delivery file %s\n", delivery_file_name);
    }
    pthread_mutex_unlock(&dvry_log_lock);
  }
    gettimeofday(&cur_time,NULL);
    tr_begin = (double)cur_time.tv_sec + 1.0e-6 * (double)cur_time.tv_usec;
    start_time = cur_time.tv_sec;

    delP->w_id = str->delin.w_id;
    delP->o_carrier_id = str->delin.o_carrier_id;
    delP->retries = 0;
    vgetdate(delP->cr_date);

    str->delout.terror = pldel(cn_dataP);

    gettimeofday(&cur_time,NULL);
    tr_end = (double)cur_time.tv_sec + 1.0e-6 * (double)cur_time.tv_usec;
    end_time = cur_time.tv_sec;

    /*  Make sure all the data pertaining to a single
     *  delivery record is written atomically
     */
    pthread_mutex_lock(&dvry_log_lock);

  #ifndef USE_ORACLE_DVRY_FORMAT
    queue_time = str->delin.qtime;
    pos = 0;
    ++tran_cntr;
  #ifdef USE_LONG_DVRY_FORMAT
    sprintf(&stdout_buf[pos], "----Transaction %d started----\n", tran_cntr);
    pos += strlen(&stdout_buf[pos]);
    sprintf(&stdout_buf[pos], "queued-time: %.6f %s",
          str->delin.qtime, ctime((time_t*)&queue_time));
    pos += strlen(&stdout_buf[pos]);
    sprintf(&stdout_buf[pos], "start-time: %.6f %s",
          tr_begin, ctime((time_t*)&start_time));
    pos += strlen(&stdout_buf[pos]);
    sprintf(&stdout_buf[pos], "W_ID: %d, CARRIER_ID: %d \n",
          str->delin.w_id, str->delin.o_carrier_id);
    pos += strlen(&stdout_buf[pos]);

    if (str->delout.terror == DEL_ERROR) {
      sprintf(&stdout_buf[pos], "Delivery transaction failed (DEL_ERROR)");
      pos += strlen(&stdout_buf[pos]);
    } else if (str->delout.terror != 0) {
      sprintf(&stdout_buf[pos], "Delivery transaction failed (%d)",
            str->delout.terror);
      pos += strlen(&stdout_buf[pos]);
    } else {
      skipped = 0;
      for (i = 0; i < 10; i++) {
        if (delP->del_o_id[i] <= 0) {
          skipped++;
          fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
                delP->w_id, i + 1);
          sprintf(&stdout_buf[pos], "D_ID %d has no neworders.\n", i+1);
          pos += strlen(&stdout_buf[pos]);
        } else {
          sprintf(&stdout_buf[pos], "D_ID: %d, O_ID: %d \n", i+1, delP->del_o_id[i]);
          pos += strlen(&stdout_buf[pos]);
        }
      }
      fprintf(lfp, "%send-time:%.6f %s\n", stdout_buf,
            tr_end, ctime((time_t*)&end_time));
    }
  #else
    pos += sprintf(&stdout_buf[pos], "--Tran %d Queue %.3f Start%.3f\n",
              tran_cntr, str->delin.qtime, tr_begin);
    pos += sprintf(&stdout_buf[pos], "W_ID: %d, CARRIER_ID: %d",
              str->delin.w_id, str->delin.o_carrier_id);

    if (str->delout.terror == DEL_ERROR) {
      pos += sprintf(&stdout_buf[pos],
                "\nDelivery transaction failed (DEL_ERROR)\n");
    } else if (str->delout.terror != 0) {
      pos += sprintf(&stdout_buf[pos], "Delivery transaction failed (%d)",
                str->delout.terror);
    } else {
      int skipped[10];
      int num_skipped = 0;
      for (i = 0; i < 10; i++) {
        if (delP->del_o_id[i] <= 0) {
          skipped[i] = 1;
          num_skipped ++;
        } else {
          skipped[i] = 0;
        }
        pos += sprintf(&stdout_buf[pos], " %d", delP->del_o_id[i]);
      }
      pos += sprintf(&stdout_buf[pos], "\n");
      if (num_skipped > 0) {
        for (i=0; i<10; i++) {
          if (skipped[i] == 1) {
            pos += sprintf(&stdout_buf[pos],
                      "D_ID %d has no neworders.\n", i+1);
          }
        }
      }
    }
```

```
  fprintf(lfp, "%send-time:%.3f\n", stdout_buf, tr_end);
#endif

  fflush (lfp);


#else
  fprintf(lfp, "%s%d%d %f %f %d %d",
        str->delout.terror == DEL_ERROR ? "DEL_ERROR: " :
        str->delout.terror != 0 ? "ERROR" : "",
        str->delin.in_timing_int,
        (tr_end - str->delin.qtime) <= DELRT ? 1 : 0,
         str->delin.qtime, tr_end, delP->w_id, delP->o_carrier_id);
  if (str->delout.terror != DEL_ERROR) {
    for (i = 0; i < 10; i++) {
      fprintf (lfp, " %d %d",i + 1, delP->del_o_id[i]);
      if (delP->del_o_id[i] <= 0) {
        fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
              delP->w_id, i + 1);
      }
    }
  }
  fprintf (lfp, " %d\n", delP->retries);
  fflush (lfp);

#endif
  pthread_mutex_unlock(&dvry_log_lock);

  str->delout.terror = NOERR;
  str->delout.retry = delP->retries;
  done_with_cn(cn_dataP, DELIVERY_TRANS, (void *)str);
  return (0);

}



TPCsto (str)
struct stostruct *str;
{
  ora_cn_data_t *cn_dataP = get_cn(STOCK_TRANS, (void *)str);
  global_stock_t *stoP = cn_dataP->stoP;

  stoP->w_id = str->stoin.w_id;
  stoP->d_id = str->stoin.d_id;
  stoP->threshold = str->stoin.threshold;
  stoP->retries = 0;

  if (str->stoout.terror = plsto (cn_dataP)) {
    err_printf("plsto> returning, terror %d, retries %d\n",
          str->stoout.terror, stoP->retries);

    if (str->stoout.terror != RECOVERR)
      str->stoout.terror = IRRECERR;
    str->stoout.retry = stoP->retries;
    ora_cn_err(cn_dataP);
    return (-1);
  }

  str->stoout.terror = NOERR;
  str->stoout.low_stock = stoP->low_stock;
  str->stoout.retry = stoP->retries;
  done_with_cn(cn_dataP, STOCK_TRANS, (void *)str);
  return (0);

}
```

tpccpl.h

```
/*
 * $Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RCS/tpccpl.h,v 1.2 1998/01/23 15:08:22 oz Exp $ Copyr (c) 1994 Oracle
 */
/*+=================================================================+
 |       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
 |                  OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                     All Rights Reserved                         |
 +=================================================================+
 | FILENAME
 |    tpccpl.h
 | DESCRIPTION
 |    Header file for TPC-C transactions in PL/SQL.
 +=================================================================*/

#ifndef TPCCPL_H
#define TPCCPL_H

#include <stdio.h>
#include "tpcc.h"

#define DELRT 80.0

extern errrpt ();
extern int ocierror(char *fname, int lineno,OCIError *errhp, sword status);
extern int sqlfile(char *fname, text *linebuf);

extern FILE *lfp;
extern FILE *fopen ();
extern int proc_no;

#ifndef DISCARD
# define DISCARD (void)
#endif

#ifndef sword
# define sword int
#endif

#define VER7          2

#define NA            -1     /* ANSI SQL NULL */
#define NLT           1      /* length for string null terminator */
#define DEADLOCK      60     /* ORA-00060: deadlock */
#define NO_DATA_FOUND  1403    /* ORA-01403: no data found */
#define NOT_SERIALIZABLE 8177  /* ORA-08177: transaction not serializable */
#define SNAPSHOT_TOO_OLD 1555  /* ORA-01555: snapshot too old */

#ifndef NULLP
# define NULLP (void *)NULL
#endif /* NULLP */

#define ADR(object) ((ub1 *)&(object))
#define SIZ(object) ((sword)sizeof(object))

typedef char date[24+NLT];
typedef char varchar2;

#define OCIERROR(errp,function)\
      ocierror(__FILE__,__LINE__,(errp),(function));

#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
     ocierror(__FILE__,__LINE__,(errp), \
          OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
     ocierror(__FILE__,__LINE__, (errp), \
          OCIBindByName((stmp),&(bndp), (errp), \
               (text *)(sqlvar), strlen((sqlvar)),\
               (progv), (progvl), (ftype),0,0,0,0,0,OCI_DEFAULT));

#define OCIBNDRA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
     ocierror(__FILE__,__LINE__,(errp), \
        OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
     ocierror(__FILE__,__LINE__,(errp), \
          OCIBindByName((stmp),&(bndp),(errp),(text*)(sqlvar),strlen((sqlvar)),\
                    (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));

#define OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,cbf_nodata,cbf_data) \
     ocierror(__FILE__,__LINE__,(errp), \
        OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
     ocierror(__FILE__,__LINE__,(errp), \
          OCIBindByName((stmp),&(bndp),(errp),(text*)(sqlvar), \
               strlen((sqlvar)),0,(progvl),(ftype), \
               indp,0,0,0,0,OCI_DATA_AT_EXEC));\
     ocierror(__FILE__,__LINE__,(errp), \
          OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)));



#define OCIBNDR(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
     ocierror(__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
     ocierror(__FILE__,__LINE__,(errp), \
       OCIBindByName((stmp),&(bndp),(errp),(text*)(sqlvar),strlen((sqlvar)),\
                    (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));

#define OCIBNDRAA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode,ms,cs) \
     ocierror(__FILE__,__LINE__, (errp), \
       OCIHandleAlloc((stmp),(dvoid**)(&(bndp)),OCI_HTYPE_BIND,0,(dvoid**)0));\
     ocierror(__FILE__,__LINE__,(errp),\
       OCIBindByName((stmp),&(bndp),(errp),(text*)(sqlvar),strlen((sqlvar)),\
    (progv),(progvl),(ftype),(indp),(alen),(arcode),(ms),(ub4*)(cs),OCI_DEFAULT));

#define OCIDEFINE(stmp,dfnp,errp,pos,progv,progvl,ftype)\
     OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),(ftype),\
               0,0,0,OCI_DEFAULT);


#define OCIDEF(stmp,dfnp,errp,pos,progv,progvl,ftype)\
     OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                    (dvoid**)0);\
     OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),\
                (ftype),NULL,NULL,NULL,OCI_DEFAULT);\


#define OCIDFNRA(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,alen,arcode) \
     OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                    (dvoid**)0);\
```

```
    OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
                   (progvl),(ftype),(indp),(alen),\
                   (arcode),OCI_DEFAULT);\
/*
    OCIDefineArrayOfStruct((dfnp),(errp),(progvl),\
                           sizeof((indp)[0]),\
                           sizeof((alen)[0]),\
                           sizeof((arcode)[0]));
*/
/*
#define OCIDFNRA(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,alen,arcode) \
    ocierror(__FILE__,__LINE__,(errp), \
        OCIHandleAlloc(tpcenv,&(dfnp),OCI_HTYPE_DEFINE,0,\
                   (dvoid**)0));\
    ocierror(__FILE__,__LINE__,(errp), \
        OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
                   (progvl),(ftype),(indp),(alen),\
                   (arcode),OCI_DEFAULT));\
    ocierror(__FILE__,__LINE__,(errp),\
        OCIDefineArrayOfStruct((dfnp),(errp),(progvl),\
                           sizeof((indp)[0]),\
                           sizeof((alen)[0]),\
                           sizeof((arcode)[0])));
*/

#define OBNDRV(lda,cursor,sqlvar,progv,progvl,ftype)\
  if (obndrv((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(progvl),(ftype),NA,\
    (sb2 *)0, (text *)0, NA, NA))\
    {errrpt(lda,cursor);return(-1);}\
  else\
    DISCARD 0

#define OBNDRA(lda,cursor,sqlvar,progv,progvl,ftype,indp,alen,arcode)\
  if (obndra((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(progvl),(ftype),NA,\
    (indp),(alen),(arcode),(ub4)0,(ub4*)0,(text*)0,NA,NA))\
    {errrpt(lda,cursor);return(-1);}\
  else\
    DISCARD 0

#define OBNDRAA(lda,cursor,sqlvar,progv,progvl,ftype,indp,alen,arcode,ms,cs)\
  if (obndra((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(progvl),(ftype),NA,\
    (indp),(alen),(arcode),(ub4)(ms),(ub4*)(cs),(text*)0,NA,NA))\
    {errrpt(lda,cursor);return(-1);}\
  else\
    DISCARD 0

#define ODEFIN(lda,cursor,pos,buf,bufl,ftype,scale,indp,fmt,fmtl,fmtt,rlen,rcode)\
  if (odefin((cursor),(pos),(ub1*)(buf),(bufl),(ftype),(scale),(indp),\
    (text*)(fmt),(fmtl),(fmtt),(rlen),(rcode)))\
    {errrpt(lda,cursor);return(-1);}\
  else\
    DISCARD 0

#define OEXFET(lda,cursor,nrows,cancel,exact)\
  if (oexfet((cursor),(nrows),(cancel),(exact)))\
    {if ((cursor)->rc == 1403) DISCARD 0; \
     else if (errrpt(lda,cursor)==RECOVERR) \
       {orol(lda);return(RECOVERR);}\
     else{orol(lda);return(-1);}}\
  else\
    DISCARD 0

#define OOPEN(lda,cursor)\
  if (oopen((cursor),(lda),(text*)0,NA,NA,(text*)0,NA))\
    {errrpt(lda,cursor);return(-1);}\
  else\
    DISCARD 0

#define OPARSE(lda,cursor,sqlstm,sqll,defflg,lngflg)\
  if (oparse((cursor),(sqlstm),(sb4)(sqll),(defflg),(ub4)(lngflg)))\
    {errrpt(lda,cursor);return(-1);}\
  else\
    DISCARD 0

#define OFEN(lda,cursor,nrows)\
  if (ofen((cursor),(nrows)))\
    {if (errrpt(lda,cursor)==RECOVERR) \
       {orol(lda);return(RECOVERR);}\
     else{orol(lda);return(-1);}}\
  else\
    DISCARD 0

#define OEXEC(lda,cursor)\
  if (oexec((cursor)))\
    {if (errrpt(lda,cursor)==RECOVERR) \
       {orol(lda);return(RECOVERR);}\
     else{orol(lda);return(-1);}}\
  else\
    DISCARD 0


#define OCOM(lda,cursor)\
  if (ocom((lda))) \
    {errrpt(lda,cursor);orol(lda);return(-1);}\
  else\
    DISCARD 0


#define OEXN(lda,cursor,iters,rowoff)\
  if (oexn((cursor),(iters),(rowoff)))\
    {if (errrpt(lda,cursor)==RECOVERR) \
       {orol(lda);return(RECOVERR);}\
     else{orol(lda);return(-1);}}\
  else\
    DISCARD 0

#endif



/*+=================================================================+
```

new.sql

```
 DECLARE   /* new order */
   not_serializable        EXCEPTION;
   PRAGMA EXCEPTION_INIT(not_serializable,-8177);
   deadlock                EXCEPTION;
   PRAGMA EXCEPTION_INIT(deadlock,-60);
   snapshot_too_old        EXCEPTION;
   PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
 BEGIN
   LOOP BEGIN
     SELECT c_discount, c_last, c_credit
      INTO :c_discount, :c_last, :c_credit
      FROM customer
      WHERE c_id = :c_id
       AND c_d_id = :d_id
       AND c_w_id = :w_id;

     UPDATE wh_dist SET d_next_o_id = d_next_o_id + 1, d_tax=d_tax+0
      WHERE  d_id = :d_id
       AND  w_id = :w_id
   RETURNING d_tax, d_next_o_id-1, w_tax
      INTO :d_tax, :o_id, :w_tax;

     INSERT INTO new_order  (no_o_id, no_d_id, no_w_id)
        VALUES (:o_id , :d_id, :w_id);
     INSERT INTO orders (o_id, o_w_id, o_d_id, o_c_id, o_carrier_id,
                 o_ol_cnt, o_all_local,o_entry_d)
        VALUES (:o_id, :w_id, :d_id, :c_id, 11,
              :o_ol_cnt, :o_all_local, :cr_date);
     EXIT;

     EXCEPTION
       WHEN not_serializable OR deadlock OR snapshot_too_old THEN
         ROLLBACK;
         :retry := :retry + 1;
     END;
   END LOOP;
 END;
```

pay.sql

```
CREATE OR REPLACE PACKAGE pay
AS
 TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
 row_id              rowidarray;
 cust_rowid            ROWID;
 dist_name             VARCHAR2(11);
 ware_name              VARCHAR2(11);
 c_num               BINARY_INTEGER;
 PROCEDURE pay_init;
END pay;
/
CREATE OR REPLACE PACKAGE BODY pay AS
 PROCEDURE pay_init IS
 BEGIN
   NULL;
 END pay_init;
END pay;
/

exit;
```

paynz.sql

```
DECLARE /* paynz */
--    cust_rowid            ROWID;
```

```
--     dist_name            VARCHAR2(11);
--     ware_name            VARCHAR2(11);
    not_serializable      EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock              EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old      EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  BEGIN
    LOOP BEGIN
       UPDATE  customer
         SET  c_balance = c_balance - :h_amount,
              c_ytd_payment = c_ytd_payment + :h_amount,
              c_payment_cnt = c_payment_cnt+1
       WHERE    c_id = :c_id AND c_d_id = :c_d_id AND
              c_w_id = :c_w_id
      RETURNING rowid, c_first, c_middle, c_last, c_street_1,
              c_street_2, c_city, c_state, c_zip, c_phone,
              c_since, c_credit, c_credit_lim,
              c_discount, c_balance
         INTO pay.cust_rowid,:c_first, :c_middle, :c_last,
:c_street_1,
              :c_street_2, :c_city, :c_state, :c_zip, :c_phone,
              :c_since, :c_credit, :c_credit_lim,
              :c_discount, :c_balance;
      IF SQL%NOTFOUNDTHEN
        raise NO_DATA_FOUND;
       END IF;

--       :c_data := ' ';

      IF :c_credit = 'BC' THEN
        UPDATE customer
                  SET c_data= substr ((to_char (:c_id) || ' ' ||
                        to_char (:c_d_id) || ' ' ||
                        to_char (:c_w_id) || ' ' ||
                        to_char (:d_id) || ' ' ||
                        to_char (:w_id) || ' ' ||
                        to_char (:h_amount, '9999.99') || ' |
')
                        || c_data, 1, 500)
         WHERE rowid = pay.cust_rowid
      RETURNING substr(c_data,1, 200)
         INTO :c_data;

      END IF;

        UPDATE district
          SET d_ytd = d_ytd + :h_amount
         WHERE d_id = :d_id
           AND d_w_id = :w_id
      RETURNING d_name, d_street_1, d_street_2, d_city,d_state, d_zip
         INTO pay.dist_name,:d_street_1,:d_street_2,:d_city,:d_state,
              :d_zip;
      IF SQL%NOTFOUNDTHEN
        raise NO_DATA_FOUND;
       END IF;

        UPDATE warehouse
          SET w_ytd = w_ytd + :h_amount
         WHERE w_id = :w_id
      RETURNING w_name, w_street_1, w_street_2, w_city, w_state, w_zip
         INTO pay.ware_name, :w_street_1, :w_street_2, :w_city,
:w_state,
              :w_zip;

      INSERT INTO history  (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                  h_amount, h_date, h_data)
      VALUES
        (:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
         :cr_date, pay.ware_name || '    ' || pay.dist_name);
      COMMIT;
      :h_date := to_char (:cr_date, 'DD-MM-YYYY.HH24:MI:SS');
      EXIT;

      EXCEPTION
        WHEN not_serializable OR deadlock OR snapshot_too_old THEN
          ROLLBACK;
          :retry := :retry + 1;
      END;

    END LOOP;
  END;
```

payz.sql

```
DECLARE /* payz */
--     TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
--     cust_rowid           ROWID;
--     dist_name            VARCHAR2(11);
--     ware_name            VARCHAR2(11);
--     c_num                BINARY_INTEGER;
--     row_id               rowidarray;
    not_serializable      EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock              EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old      EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
    CURSOR c_cur IS
      SELECT rowid
      FROM customer
      WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last = :c_last
      ORDER BY c_w_id, c_d_id, c_last, c_first;
  BEGIN
    LOOP BEGIN

      pay.c_num := 0;
      FOR c_id_rec IN c_cur LOOP
        pay.c_num := pay.c_num + 1;
        pay.row_id(pay.c_num) := c_id_rec.rowid;
      END LOOP;
      pay.cust_rowid := pay.row_id ((pay.c_num + 1) / 2);

      UPDATE customer
       SET c_balance = c_balance - :h_amount,
          c_ytd_payment = c_ytd_payment+:h_amount,
          c_payment_cnt = c_payment_cnt+1
      WHERE rowid = pay.cust_rowid
      RETURNING
            c_id, c_first, c_middle, c_last, c_street_1, c_street_2,
            c_city, c_state, c_zip, c_phone,
            c_since, c_credit, c_credit_lim,
            c_discount, c_balance
        INTO :c_id, :c_first, :c_middle, :c_last,
             :c_street_1, :c_street_2, :c_city, :c_state,
             :c_zip, :c_phone, :c_since, :c_credit,
             :c_credit_lim, :c_discount, :c_balance;

      :c_data := ' ';
      IF :c_credit = 'BC' THEN
        UPDATE customer
          SET c_data = substr ((to_char (:c_id) || ' ' ||
                        to_char (:c_d_id) || ' ' ||
                        to_char (:c_w_id) || ' ' ||
                        to_char (:d_id) || ' ' ||
                        to_char (:w_id) || ' ' ||
                        to_char (:h_amount/100, '9999.99') ||
' |
')
                        || c_data, 1, 500)
         WHERE rowid = pay.cust_rowid
         RETURNING substr(c_data,1, 200)
         INTO :c_data;

      END IF;

      UPDATE district
        SET d_ytd = d_ytd+:h_amount
        WHERE d_id = :d_id
          AND d_w_id = :w_id
        RETURNING  d_name, d_street_1, d_street_2, d_city,
                   d_state, d_zip
        INTO pay.dist_name, :d_street_1, :d_street_2, :d_city,
             :d_state, :d_zip;
      IF SQL%NOTFOUNDTHEN
        raise NO_DATA_FOUND;
       END IF;

      UPDATE warehouse
        SET w_ytd = w_ytd+:h_amount
        WHERE w_id = :w_id
        RETURNING w_name,
              w_street_1, w_street_2, w_city, w_state, w_zip
         INTO pay.ware_name,
             :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;

      INSERT INTO history (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                   h_amount, h_date, h_data)
       VALUES (:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
        :cr_date, pay.ware_name || '    ' || pay.dist_name);
      COMMIT;
      :h_date := to_char (:cr_date, 'DD-MM-YYYY.HH24:MI:SS');
      EXIT;

      EXCEPTION
        WHEN not_serializable OR deadlock OR snapshot_too_old THEN
          ROLLBACK;
          :retry := :retry + 1;
      END;

    END LOOP;
  END;
```

views.sql

```
create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
   from district d, warehouse w
   where w.w_id = d.d_w_id
/

create  or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
 select i.i_id, s_w_id, i.i_price, i.i_name, i.i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
  from stock s, item i
 where i.i_id = s.s_i_id
/

exit;
```

Appendix B: Tunable Parameters

B.1 Database Parameters

5s7a_common.ora

```
#
#===============================================================
==+
# FILENAME
#      5s7a_common.ora
# DESCRIPTION
#      Oracle parameter file for running TPC-C.
# MODIFIED
#      YPang created for 5 nodes BB cluster.
#
#===============================================================
==
#

control_files = /dev/rtpcc_cntl1, /dev/rtpcc_cntl2, /dev/rtpcc_cntl3
compatible            = 8.0.5
parallel_server                         = TRUE
disk_asynch_io        = TRUE

db_writer_processes=3

recovery_parallelism = 20
db_name               = tpcc
db_files                      = 3000
db_block_size         = 4096
dml_locks             = 2000
enqueue_resources     = 3500
log_archive_start     = FALSE
log_archive_buffer_size = 32
log_checkpoint_interval = 1000000000
log_checkpoints_to_alert = TRUE
log_simultaneous_copies = 14
gc_releasable_locks       = 100000
max_rollback_segments = 400
max_dump_file_size        = 3000
open_cursors          = 500
sessions              = 1000
transactions          = 1000
distributed_transactions = 0
transactions_per_rollback_segment = 1

discrete_transactions_enabled = FALSE
cursor_space_for_time   =                TRUE
replication_dependency_tracking=FALSE
log_small_entry_max_size =               80
processes             =       500
shared_pool_size =                       40000000
db_block_lru_latches =                   24

buffer_pool_recycle =  (buffers:100000,lru_latches:4)

spin_count = 8000
log_buffer            = 4194304

_db_block_write_batch   = 4096
db_block_checkpoint_batch = 2048

db_block_buffers  =     5000000
_db_block_hash_buckets = 1842667

db_block_max_dirty_target=0

recovery_parallelism  = 24

lm_procs = 3000
lm_ress = 2500000
lm_locks = 5000000
_lm_send_buffers=900000

gc_rollback_locks="0-1600=100EACH"

gc_files_to_locks="1,2656=4000" # SYSTEM
gc_files_to_locks="2-6=1EACH" # ROLL
gc_files_to_locks="8,10=24320EACH" # DIST

gc_files_to_locks="87-526=987!200EACH:" # CUST
gc_files_to_locks="47-86=1EACH:" # HIST
gc_files_to_locks="1914,1917-1919,1922,1924,1926,1928,1930,1932,1934,1936,\
1939,1941,1943,1945,1947,1949,1951,1953,2037,2042,2045,2047,2049=1EACH:"
#ICU1
gc_files_to_locks="1915-1916,1920-1921,1923,1925,1927,1929,1931,1933,1935,\
1937-1938,1940,1942,1944,1946,1948,1950,1952,1954-1983,2035,2038,2041\
,2043,2046,2048,2050-2053=1EACH:" # ICU2
gc_files_to_locks="541-576,1984-1997=1EACH:" # INOR
gc_files_to_locks="1713,1718,1722,1727,1732,1737,1742,1747,1751,1757,1762,\
1767,1772,1777,1783,1786,1791,1795,1800,1806,1810,1815,1820,1825,1829,1832,
\
1836,1841,1845,1850,1864-1883=1EACH:" # IOR1
gc_files_to_locks="1884-1913,1998-2033,2054-2067,2383-2402=1EACH:" # IOR2
gc_files_to_locks="2068-2083,2189-2281,2403=1EACH:" # IORL
gc_files_to_locks="2034,2036,2039-2040,2044,2294-2313=1EACH:" # ISTK
gc_files_to_locks="9=1:" # ITEM
gc_files_to_locks="11-46,527-540=1EACH:" # NORD

gc_files_to_locks="1465-1468,1470-1473,1475-1479,1481-1484,1486-1489,\
1491-1494,1496-1498,1500-1503,1505-1508,1510-1514,1516-1518,1520-1523,\
1525-1528,1530-1532,1534-1537,1539-1542,1544-1546,1548-1551,1553\
-1556,1558-1561,1563-1565,1567-1572,1574-1576,1578-1581,1583\
-1586,1588-1591,1593-1595,1597-1600,1602-1605,1607-1610,1612\
-1615,1617-1620,1622-1624,1626-1628,1630-1633,1635-1638,1640\
-1643,1645-1648,1650-1652,1654-1658,1660-1663,1665-1668,1670\
-1673,1675-1678,1680-1683,1685-1687,1689-1692,1694-1697,1699\
-1702,1704-1707,1709-1712,1714-1717,1719-1721,1723-1726,1728\
-1731,1733-1736,1738-1741,1743-1746,1748-1750,1752-1756,1758\
-1761,1763-1766,1768-1771,1773-1776,1778-1782,1784-1785,1787\
-1790,1792-1794,1796-1799,1801-1805,1807-1809,1811-1814,1816\
-1819,1821-1824,1826-1828,1830-1831,1833-1835,1837-1840,1842\
-1844,1846-1849,1851-1863,2282-2293,2314-2382=1EACH:" # ORDL

gc_files_to_locks="1469,1474,1480,1485,1490,1495,1499,1504,1509,1515,1519,\
1524,1529,1533,1538,1543,1547,1552,1557,1562,1566,1573,1577,1582,1587,1592,
\
1596,1601,1606,1611,1616,1621,1625,1629,1634,1639,1644,1649,1653,1659,1664,
\
1669,1674,1679,1684,1688,1693,1698,1703,1708=1EACH:" # ORDS

#gc_files_to_locks="577-1464=950EACH:2419-2529=5700EACH:" # STOC

gc_files_to_locks="577-585,587-602,604-619,622-637,639-654,656-667,669-692,\
694-710,712-727,729-744,746-779,781-796,798-831,833-848,850-878,880-883,\
885-904,906-921,923-956,958-973,975-1008,1010-1025,1027-1036,1038-1057,\
1059-1092,1094-1109,1111-1144,1146-1161,1163-1178,1180-1213,1215-1230,\
1232-1243,1245-1286,1288-1303,1305-1338,1340-1355,1357-1390,1392-1407,\
1409-1442,1444-1454,1456-1459,1461-1464=950EACH:\
2419-2420,2422-2424,2431-2432,2436-2444,2448-2449,2454-2459,2462,2465-2467
,\
2471-2529=5700EACH:" # STOC ABW KTP

gc_files_to_locks="7=11010" # WARE

gc_files_to_locks="586,638,655,668,728,780,879,922,974,1026,1244,1455,1460=\
3800EACH:" # STOC ABW KTP

gc_files_to_locks="2421,2425-2430,2433-2435,2445-2447,2450-2453,2460,\
2461,2463,2464,2468-2470=7600EACH:" # STOC ABW KTP

gc_files_to_locks="603,620-621,693,711,745,797,832,849,884,905,957,1009,1037
,\
1058,1093,1110,1145,1162,1179,1214,1231,1287,1304,1339,1356,1391,1408,1443\
=2850EACH:" # STOC ABW KTP
```

P_run_n1.ora

```
ifile=/home/oracle/app/oracle/product/8.0.5/bench/tpc/tpcc/scripts/5s7a/admin/5
s7a_common.ora
instance_number = 1
thread = 1
rollback_segments = (\
a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,\
a11,a12,a13,a14,a15,a16,a17,a18,a19,a20,\
a21,a22,a23,a24,a25,a26,a27,a28,a29,a30,\
a31,a32,a33,a34,a35,a36,a37,a38,a39,a40,\
a41,a42,a43,a44,a45,a46,a47,a48,a49,a50,\
a51,a52,a53,a54,a55,a56,a57,a58,a59,a60,\
a61,a62,a63,a64,a65,a66,a67,a68,a69,a70,\
a71,a72,a73,a74,a75,a76,a77,a78,a79,a80,\
a81,a82,a83,a84,a85,a86,a87,a88,a89,a90,\
a91,a92,a93,a94,a95,a96,a97,a98,a99,a100,\
a101,a102,a103,a104,a105,a106,a107,a108,a109,a110,\
a111,a112,a113,a114,a115,a116,a117,a118,a119,a120,\
a121,a122,a123,a124,a125,a126,a127,a128,a129,a130,\
a131,a132,a133,a134,a135,a136,a137,a138,a139,a140,\
```

a141,a142,a143,a144,a145,a146,a147,a148,a149,a150,
a151,a152,a153,a154,a155,a156,a157,a158,a159,a160,
a161,a162,a163,a164,a165,a166,a167,a168,a169,a170,
a171,a172,a173,a174,a175,a176,a177,a178,a179,a180,
a181,a182,a183,a184,a185,a186,a187,a188,a189,a190,
a191,a192,a193,a194,a195,a196,a197,a198,a199,a200,
a201,a202,a203,a204,a205,a206,a207,a208,a209,a210,
a211,a212,a213,a214,a215,a216,a217,a218,a219,a220,
a221,a222,a223,a224,a225,a226,a227,a228,a229,a230,
a231,a232,a233,a234,a235,a236,a237,a238,a239,a240,
a241,a242,a243,a244,a245,a246,a247,a248,a249,a250,
a251,a252,a253,a254,a255,a256,a257,a258,a259,a260,
a261,a262,a263,a264,a265,a266,a267,a268,a269,a270,
a271,a272,a273,a274,a275,a276,a277,a278,a279,a280,
a281,a282,a283,a284,a285,a286,a287,a288,a289,a290,
a291,a292,a293,a294,a295,a296,a297,a298,a299,a300,
a301,a302,a303,a304,a305,a306,a307,a308,a309,a310,
a311,a312,a313,a314,a315,a316,a317,a318,a319,a320,
a321,a322,a323,a324,a325,a326,a327,a328,a329,a330,
a331,a332,a333,a334,a335,a336,a337,a338,a339,a340,
a341,a342,a343,a344,a345,a346,a347,a348,a349,a350)

P_run_n2.ora

ifile=/home/oracle/app/oracle/product/8.0.5/bench/tpc/tpcc/scripts/5s7a/admin/5s7a_common.ora
instance_number = 2
thread = 2
rollback_segments = (\
b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,\
b11,b12,b13,b14,b15,b16,b17,b18,b19,b20,\
b21,b22,b23,b24,b25,b26,b27,b28,b29,b30,\
b31,b32,b33,b34,b35,b36,b37,b38,b39,b40,\
b41,b42,b43,b44,b45,b46,b47,b48,b49,b50,\
b51,b52,b53,b54,b55,b56,b57,b58,b59,b60,\
b61,b62,b63,b64,b65,b66,b67,b68,b69,b70,\
b71,b72,b73,b74,b75,b76,b77,b78,b79,b80,\
b81,b82,b83,b84,b85,b86,b87,b88,b89,b90,\
b91,b92,b93,b94,b95,b96,b97,b98,b99,b100,\
b101102,b103,b104,b105,b106,b107,b108,b109,b110,\
b111,b112,b113,b114,b115,b116,b117,b118,b119,b120,\
b121,b122,b123,b124,b125,b126,b127,b128,b129,b130,\
b131,b132,b133,b134,b135,b136,b137,b138,b139,b140,\
b141,b142,b143,b144,b145,b146,b147,b148,b149,b150,\
b151,b152,b153,b154,b155,b156,b157,b158,b159,b160,\
b161,b162,b163,b164,b165,b166,b167,b168,b169,b170,\
b171,b172,b173,b174,b175,b176,b177,b178,b179,b180,\
b181,b182,b183,b184,b185,b186,b187,b188,b189,b190,\
b191,b192,b193,b194,b195,b196,b197,b198,b199,b200,\
b201,b202,b203,b204,b205,b206,b207,b208,b209,b210,\
b211,b212,b213,b214,b215,b216,b217,b218,b219,b220,\
b221,b222,b223,b224,b225,b226,b227,b228,b229,b230,\
b231,b232,b233,b234,b235,b236,b237,b238,b239,b240,\
b241,b242,b243,b244,b245,b246,b247,b248,b249,b250,\
b251,b252,b253,b254,b255,b256,b257,b258,b259,b260,\
b261,b262,b263,b264,b265,b266,b267,b268,b269,b270,\
b271,b272,b273,b274,b275,b276,b277,b278,b279,b280,\
b281,b282,b283,b284,b285,b286,b287,b288,b289,b290,\
b291,b292,b293,b294,b295,b296,b297,b298,b299,b300,\
b301,b302,b303,b304,b305,b306,b307,b308,b309,b310,\
b311,b312,b313,b314,b315,b316,b317,b318,b319,b320,\
b321,b322,b323,b324,b325,b326,b327,b328,b329,b330,\
b331,b332,b333,b334,b335,b336,b337,b338,b339,b340,\
b341,b342,b343,b344,b345,b346,b347,b348,b349,b350)

P_run_n3.ora

ifile=/home/oracle/app/oracle/product/8.0.5/bench/tpc/tpcc/scripts/5s7a/admin/5s7a_common.ora
instance_number = 3
thread = 3
rollback_segments = (
c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,
c11,c12,c13,c14,c15,c16,c17,c18,c19,c20,
c21,c22,c23,c24,c25,c26,c27,c28,c29,c30,
c31,c32,c33,c34,c35,c36,c37,c38,c39,c40,
c41,c42,c43,c44,c45,c46,c47,c48,c49,c50,
c51,c52,c53,c54,c55,c56,c57,c58,c59,c60,
c61,c62,c63,c64,c65,c66,c67,c68,c69,c70,
c71,c72,c73,c74,c75,c76,c77,c78,c79,c80,
c81,c82,c83,c84,c85,c86,c87,c88,c89,c90,
c91,c92,c93,c94,c95,c96,c97,c98,c99,c100,
c101,c102,c103,c104,c105,c106,c107,c108,c109,c110,
c111,c112,c113,c114,c115,c116,c117,c118,c119,c120,
c121,c122,c123,c124,c125,c126,c127,c128,c129,c130,
c131,c132,c133,c134,c135,c136,c137,c138,c139,c140,
c141,c142,c143,c144,c145,c146,c147,c148,c149,c150,
c151,c152,c153,c154,c155,c156,c157,c158,c159,c160,
c161,c162,c163,c164,c165,c166,c167,c168,c169,c170,
c171,c172,c173,c174,c175,c176,c177,c178,c179,c180,
c181,c182,c183,c184,c185,c186,c187,c188,c189,c190,
c191,c192,c193,c194,c195,c196,c197,c198,c199,c200,
c201,c202,c203,c204,c205,c206,c207,c208,c209,c210,
c211,c212,c213,c214,c215,c216,c217,c218,c219,c220,
c221,c222,c223,c224,c225,c226,c227,c228,c229,c230,
c231,c232,c233,c234,c235,c236,c237,c238,c239,c240,
c241,c242,c243,c244,c245,c246,c247,c248,c249,c250,
c251,c252,c253,c254,c255,c256,c257,c258,c259,c260,
c261,c262,c263,c264,c265,c266,c267,c268,c269,c270,
c271,c272,c273,c274,c275,c276,c277,c278,c279,c280,
c281,c282,c283,c284,c285,c286,c287,c288,c289,c290,
c291,c292,c293,c294,c295,c296,c297,c298,c299,c300,
c301,c302,c303,c304,c305,c306,c307,c308,c309,c310,
c311,c312,c313,c314,c315,c316,c317,c318,c319,c320,
c321,c322,c323,c324,c325,c326,c327,c328,c329,c330,
c331,c332,c333,c334,c335,c336,c337,c338,c339,c340,
c341,c342,c343,c344,c345,c346,c347,c348,c349,c350)

p_run_n4.ora

ifile=/home/oracle/app/oracle/product/8.0.5/bench/tpc/tpcc/scripts/5s7a/admin/5s7a_common.ora
instance_number = 4
thread = 4
rollback_segments = (
d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,
d11,d12,d13,d14,d15,d16,d17,d18,d19,d20,
d21,d22,d23,d24,d25,d26,d27,d28,d29,d30,
d31,d32,d33,d34,d35,d36,d37,d38,d39,d40,
d41,d42,d43,d44,d45,d46,d47,d48,d49,d50,
d51,d52,d53,d54,d55,d56,d57,d58,d59,d60,
d61,d62,d63,d64,d65,d66,d67,d68,d69,d70,
d71,d72,d73,d74,d75,d76,d77,d78,d79,d80,
d81,d82,d83,d84,d85,d86,d87,d88,d89,d90,
d91,d92,d93,d94,d95,d96,d97,d98,d99,d100,
d101,d102,d103,d104,d105,d106,d107,d108,d109,d110,
d111,d112,d113,d114,d115,d116,d117,d118,d119,d120,
d121,d122,d123,d124,d125,d126,d127,d128,d129,d130,
d131,d132,d133,d134,d135,d136,d137,d138,d139,d140,
d141,d142,d143,d144,d145,d146,d147,d148,d149,d150,
d151,d152,d153,d154,d155,d156,d157,d158,d159,d160,
d161,d162,d163,d164,d165,d166,d167,d168,d169,d170,
d171,d172,d173,d174,d175,d176,d177,d178,d179,d180,
d181,d182,d183,d184,d185,d186,d187,d188,d189,d190,
d191,d192,d193,d194,d195,d196,d197,d198,d199,d200,
d201,d202,d203,d204,d205,d206,d207,d208,d209,d210,
d211,d212,d213,d214,d215,d216,d217,d218,d219,d220,
d221,d222,d223,d224,d225,d226,d227,d228,d229,d230,
d231,d232,d233,d234,d235,d236,d237,d238,d239,d240,
d241,d242,d243,d244,d245,d246,d247,d248,d249,d250,
d251,d252,d253,d254,d255,d256,d257,d258,d259,d260,
d261,d262,d263,d264,d265,d266,d267,d268,d269,d270,
d271,d272,d273,d274,d275,d276,d277,d278,d279,d280,
d281,d282,d283,d284,d285,d286,d287,d288,d289,d290,
d291,d292,d293,d294,d295,d296,d297,d298,d299,d300,
d301,d302,d303,d304,d305,d306,d307,d308,d309,d310,
d311,d312,d313,d314,d315,d316,d317,d318,d319,d320,
d321,d322,d323,d324,d325,d326,d327,d328,d329,d330,
d331,d332,d333,d334,d335,d336,d337,d338,d339,d340,
D341,d342,d343,d344,d345,d346,d347,d348,d349,d350)

P_run_n5.ora

ifile=/home/oracle/app/oracle/product/8.0.5/bench/tpc/tpcc/scripts/5s7a/admin/5s7a_common.ora
instance_number = 5
thread = 5
rollback_segments = (
e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,
e11,e12,e13,e14,e15,e16,e17,e18,e19,e20,
e21,e22,e23,e24,e25,e26,e27,e28,e29,e30,
e31,e32,e33,e34,e35,e36,e37,e38,e39,e40,
e41,e42,e43,e44,e45,e46,e47,e48,e49,e50,
e51,e52,e53,e54,e55,e56,e57,e58,e59,e60,
e61,e62,e63,e64,e65,e66,e67,e68,e69,e70,
e71,e72,e73,e74,e75,e76,e77,e78,e79,e80,
e81,e82,e83,e84,e85,e86,e87,e88,e89,e90,
e91,e92,e93,e94,e95,e96,e97,e98,e99,e100,
e101,e102,e103,e104,e105,e106,e107,e108,e109,e110,
e111,e112,e113,e114,e115,e116,e117,e118,e119,e120,
e121,e122,e123,e124,e125,e126,e127,e128,e129,e130,
e131,e132,e133,e134,e135,e136,e137,e138,e139,e140,
e141,e142,e143,e144,e145,e146,e147,e148,e149,e150,

```
e151,e152,e153,e154,e155,e156,e157,e158,e159,e160,\
e161,e162,e163,e164,e165,e166,e167,e168,e169,e170,\
e171,e172,e173,e174,e175,e176,e177,e178,e179,e180,\
e181,e182,e183,e184,e185,e186,e187,e188,e189,e190,\
e191,e192,e193,e194,e195,e196,e197,e198,e199,e200,\
e201,e202,e203,e204,e205,e206,e207,e208,e209,e210,\
e211,e212,e213,e214,e215,e216,e217,e218,e219,e220,\
e221,e222,e223,e224,e225,e226,e227,e228,e229,e230,\
e231,e232,e233,e234,e235,e236,e237,e238,e239,e240,\
e241,e242,e243,e244,e245,e246,e247,e248,e249,e250,\
e251,e252,e253,e254,e255,e256,e257,e258,e259,e260,\
e261,e262,e263,e264,e265,e266,e267,e268,e269,e270,\
e271,e272,e273,e274,e275,e276,e277,e278,e279,e280,\
e281,e282,e283,e284,e285,e286,e287,e288,e289,e290,\
e291,e292,e293,e294,e295,e296,e297,e298,e299,e300,\
e301,e302,e303,e304,e305,e306,e307,e308,e309,e310,\
e311,e312,e313,e314,e315,e316,e317,e318,e319,e320,\
e321,e322,e323,e324,e325,e326,e327,e328,e329,e330,\
e331,e332,e333,e334,e335,e336,e337,e338,e339,e340,\
e341,e342,e343,e344,e345,e346,e347,e348,e349,e350)
```

B.2 Transaction Monitor Parameters

tpccCommon.tcl

```
#
# ID: $Id: tpccCommon.tcl,v 1.14 1997/08/01 13:08:58 oz Exp $
#
# COMPONENT_NAME:  Encina Administration Test
#
# ORIGINS: Transarc Corp.
#
# (C) COPYRIGHT Transarc Corp. 1995
# All Rights Reserved
# Licensed Materials - Property of Transarc
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with Transarc Corp
#

# HISTORY
# $TALog: tpccCommon.tcl,v $
#


#-------------------------------------------------------------------------
# This tcl script provides procedures for creating and starting cells,
# node managers, sfs, rqs and mas servers. It is a general purpose
# script which uses enccp commands to accomplish the abvove.
#
# This script determines the cell name by checking whether the environment
# variable ENCINA_TPM_CELL is set and if so uses that as the CELL name
# otherwise, it builds a cell name using the user name and machine name
#

#-------------------------------------------------------------------------
# getenv - import an environment variable.  Error if it doesn't exist.
#-------------------------------------------------------------------------
proc getenv {variable} {
   global $variable
   global env
   set $variable $env($variable)
}

##  Default values used for the different scripts
set DEFAULT_TRACE "all=none"
set TPCC_BIN_DIR "/home/encina/bin"
getenv USER

set DCE_PWD e26spw

## The number PAs each server is configured to have
set MT_PAS 18
set MT_THREADS 1

set MT_DVRY_PAS 1
set DVRY_THREADS 4

set DB_NAME oratpcc

set NO_PAS  45
set PAY_PAS  15
set SL_PAS 5
set OS_PAS 3
set DVRY_PAS 5
set XA_PAS 0
set REMOTE_PAS 0
set DEFAULT_PAS 1

set QUEUE_PARAM "ENCINA_THREAD_POOL_QUEUE_LENGTH=1800
ENCINA_DEFAULT_THREAD_POOL_QUEUE_LENGTH=1800"
set UNAME [exec uname]
set ENCINA_BIN_PATH "/usr/lpp/encina/bin"

if { $UNAME == "AIX" } {
  set HOSTNAME [exec hostname -s]
} else {
  set HOSTNAME [exec hostname]
}

#set ADDITIONAL_ENV "RPC_SUPPORTED_PROTSEQS=ncadg_ip_udp
TRPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp"
#set ADDITIONAL_ENV "RPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp
TRPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp"
set ADDITIONAL_ENV ""

set ADDITIONAL_MAS_ENV "RPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp
TRPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp"
#set ADDITIONAL_MAS_ENV "RPC_SUPPORTED_PROTSEQS=ncadg_ip_udp
TRPC_SUPPORTED_PROTSEQS=ncadg_ip_udp"
# set ADDITIONAL_MAS_ENV ""

set UNSUPPORTED_NETIFS $env(RPC_UNSUPPORTED_NETIFS)
if {"${HOSTNAME}" == "perf" || "${HOSTNAME}" == "hydra1" ||
   "${HOSTNAME}" == "titan" } {
   set TPCC_BIN_DIR
"/afs/tr/usr/oz/kansas/test/build/benchmarks/tpcc/sp-tpcc"
   set UNSUPPORTED_NETIFS ""
   set ENCINA_BIN_PATH "/afs/tr/kansas/latest/internal/bin"
   set DEFAULT_PAS 2
   set NO_PAS 2
   set PAY_PAS 2
   set SL_PAS 2
   set OS_PAS 2
   set DVRY_PAS 2
   set XA_PAS 2
   set REMOTE_PAS 4
}
if {"${HOSTNAME}" == "frog"} {
   set TPCC_BIN_DIR
"/afs/tr/usr/oz/kansas/test/build/benchmarks/tpcc/sp-tpcc"
   set UNSUPPORTED_NETIFS ""
   set ENCINA_BIN_PATH "/afs/tr/kansas/latest/internal/bin"
   set DEFAULT_PAS 2
   set NO_PAS 2
   set PAY_PAS 2
   set SL_PAS 2
   set OS_PAS 2
   set DVRY_PAS 2
   set XA_PAS 2
   set REMOTE_PAS 4
}

proc showHelp {} {
   global NO_PAS SL_PAS OS_PAS DVRY_PAS PAY_PAS XA_PAS
REMOTE_PAS
   puts "There are a number of scripts to simplify the TPCC"
   puts "configuration.  The main ones are:"
   puts "  showMas -- shows the current status of all MAS"
   puts "  stopAllMas - Stops all the MASs"
   puts "  startAllMas - Start all the MASs"
   puts "  deleteAllMas - Delete all the MASs"
   puts "  showHelp -- Show this help message"
   puts " "
   puts "  doMas <op> <node> - perform the op on all the MAS of that node"
   puts "      e.g.  doMas start k66n28"
   puts " "
   puts "  Configuration"
   puts "     createCell"
   puts "     createNode"
   puts "     createMas <node> <db> - Create and start all the MASs needed"
   puts "          for the specified DB on the specified node"
   puts "     createMasDbg <node> <db> - Create and start all the "
   puts "          debug MASs needed"
   puts "          for the specified DB on the specified node"
   puts "     createTpcc <numNodes> - Create and start all the MASs needed"
   puts "          for the specified number of nodes.  The nodes are:"
   puts "          28->DB13, 27->DB05, 29->DB09, and 30->DB01"
   puts "          if <n> nodes are specified the first n nodes will be created"
   puts "    configMas <node> <DB> - Set the paCount on all the MASs."
   puts "    configMasDbg <node> <DB> - Set the paCount on all the "
   puts "         debug MASs."
   puts "      Defaults: NO $NO_PAS, PAY $PAY_PAS, OS $OS_PAS, SL
$SL_PAS, DVRY $DVRY_PAS, XA $XA_PAS, REMOTE $REMOTE_PAS"
   puts "  "
```

```
    puts "   createMas calls the following three functions:"
    puts "     dari_mas_all <node> <name> <db_spec> - Create all the local"
    puts "          servers for the specified node and db"
    puts "           e.g.  dari_mas_all k66n28 05A db:tpcc05"
    puts "     dari_xa <node> <name> <db_spec> <if> - create an XA server"
    puts "          on the specified node.  <if> should be -1"
    puts "          e.g. dari_xa k66n28 05A db:tpcc05 -1"
    puts "     dari_remote <node> <name> <db_spec> - create a REMOTE server"
    puts "          on the specified node.  "
    puts "          e.g. dari_remote k66n28 05A db:tpcc05"
    puts "  "
    puts "  Conventions: "
    puts "   - The names of the servers start with the name of the node"
    puts "   - The names contain the DB it is connect to"
    puts "   - Remote servers have -remote- in their names and XA ones"
    puts "     have -xa- in their name."
    puts "   - Local servers end with the name of the transaction they process."
    puts "   - The name of the nodes is comprised of the short name of the"
    puts "     machine."
    puts "  Example names:"
    puts "     k66n28-xa-05A"
    puts "     k66n28-05A-dvry"
    puts "  "
}

proc listMas {} {
   foreach mas [mas list] {
      puts "$mas"
   }
}

proc countThreads {mas} {
   set vars [mas show $mas -environment]
   foreach var $vars {
      if [regexp {([^=]*)=(.*)$} $var A B C] {
                  echo "Var $B  is  $C"
      }
   }
}

proc showMas {} {
    foreach mas [mas list] {
       mas show $mas -setArray masInfo
       set threads 0
       foreach el $masInfo(environment) {
          if [regexp {ENCINA_(APPL_|)TPOOL_SIZE=([0-9]+)} $el a name num] {
             set threads [expr $threads + $num]
          }
       }
       set dbase ""
       scan $masInfo(commandLineArgs) "%d MON_%s %d dvry=%d db:%s" flag
type debug dvry dbase
       set cn 0

       if [expr $flag & 8] {
          set cn [expr $cn + $dvry]
       }
       if [expr $flag & ~8] {
          set cn [expr $cn + $threads]
       }
       if {$masInfo(authorizationEnabled) == "true"} {
                   set Z "Auth: Enabled"
       } else {
                   set Z ""
       }
       puts "$mas: $masInfo(currentState), $masInfo(paCount) PAs, $threads thr,
$cn cn, db: $dbase $Z"
    }
}

proc displayAll {} {
    puts "\n\n\tECM\n\t---\n\n"
    puts [ecm show]
    puts "\n\n"
    foreach enm [enm list] {
                puts "\n\n\tNode: $enm\n\t----------\n\n"
                echo [enm show $enm]
    }
    displayAllMas
}

proc displayAllMas {} {
    foreach mas [mas list] {
                puts "\n\n\tApp Server $mas\n\t----------\n"
                echo [mas show $mas]
    }
    puts "\n"
}

#-------------------------------------------------------------------

# DO - echo and execute a command
#-------------------------------------------------------------------
# note: The command output is put in the result, but not echoed.
#
# limitation:  The command cannot be abbreviated or auto-loaded
#
proc DO {args} {
    puts "# $args"
    flush stdout
    uplevel $args
}

#-------------------------------------------------------------------
# nuke - remove any traces of a server, which may or may not exist or
#        be running from previous tests.  Volumes are not deleted.
#-------------------------------------------------------------------

proc nuke {objtype server} {
    catch {DO $objtype stop -gentleTimeout 0 $server}
    catch {DO $objtype erase $server}
    catch {DO $objtype delete -references $server}
    #
    # Now, here's the tricky part.  If we've created a server in the
    # current session, and we remove its principal, and we recreate the
    # same server, it fails.  alib.c has the following comment:
    #
    #             We do not delete the principal and keyfile, since otherwise if
    #             the same principal and keyfile is recreated (e.g., the server is
    #             again cold started), the node manager will reuse the old tickets
    #             and the new SFS will not honor them.  This results is the error
    #             RPC_CN_AUTH_VFY_CLIENT_REQ at the server.
    #
    # This really needs further investigation.  In the meantime, delete
    # it any time we don't have full access to it.  This means it will
    # be deleted any time it would cause a subsequent create operation
    # to fail, and not be deleted if we've created it ourselves.
    #
    global SHORT_CELL_NAME
    if {$objtype != "enm"} {
       set principal $SHORT_CELL_NAME/server/$server
    } else {
       set principal $SHORT_CELL_NAME/node/$server
    }

    set status [catch {acl check  /.:/sec/principal/$principal} access]
    if {$status != 0 && $access != "acl object not found"} {
                error $access
    }

    if {$status == 0 && $access != "rcDnfmaug"} {
                DO login cell_admin -password -dce-
                DO principal delete $principal
                DO logout
    }
}

#-------------------------------------------------------------------
# createVol - create a volume object for starting a server
#-------------------------------------------------------------------

proc createVol {volumeObject rawVolume} {
    global UNAME
    if { $UNAME == "AIX" } {
                DO lvol create ${volumeObject} -nativeVolumeName $rawVolume
    } else {
                DO pvol create ${volumeObject}_pvol -regions $rawVolume
                DO lvol create ${volumeObject} -physicalVolumes
${volumeObject}_pvol
    }
}

#-------------------------------------------------------------------
# deleteVol - delete a volume created by createVol
#-------------------------------------------------------------------

proc deleteVol {volumeObject} {
    global UNAME
    DO lvol delete ${volumeObject}
    if { $UNAME != "AIX" } {
                DO pvol delete ${volumeObject}_pvol
    }
}

#-------------------------------------------------------------------
# deleteAllVols - delete all volumes
#-------------------------------------------------------------------

proc deleteAllVols {} {
    foreach volumeObject [lvol list] {
                deleteVol $volumeObject
```

```
   }
}


proc deleteAllEnm { } {
   foreach node [enm list] {
      puts "Deleting node manager $node"
      deleteEnm $node
   }
}

#-------------------------------------------------------------------
# deleteAllMas - delete all mas servers
#-------------------------------------------------------------------

proc deleteAllMas {} {
   foreach masSvr [mas list] {
     puts "Deleting $masSvr"
     catch {mas stop $masSvr}
     catch {mas delete $masSvr}
   }
}

proc startAllMas {} {
   foreach masSvr [mas list] {
      DO mas start $masSvr
   }
}
proc stopAllMas {} {
   foreach masSvr [mas list] {
     DO catch " mas stop $masSvr "
   }
}

#-------------------------------------------------------------------
# deleteAllInterfaces - delete all interfaces
#-------------------------------------------------------------------

proc deleteAllInterfaces {} {
   foreach if [interface list] {
      DO catch {interface delete $if}
   }
}


proc setCell {name} {
  global ENCINA_TPM_CELL
  set ENCINA_TPM_CELL /.:/$cell
}

proc setBin {name} {
  global TPCC_BIN_DIR

  set TPCC_BIN_DIR /u/encina/encina_tpcc/$name
}

#-------------------------------------------------------------------
# get machine and user name
#-------------------------------------------------------------------

#-------------------------------------------------------------------
# set cell name
# First check if the environment variable ENCINA_TPM_CELL is defined
# and if it is use that as the cell name, else build a cell name based
# on the user name and machine
#-------------------------------------------------------------------

   set ET_ID "$USER-$HOSTNAME"

   set found 0
   foreach el [lsort [array names env]] {
               if {$el == "ENCINA_TPM_CELL"} {
                  set ENCINA_TPM_CELL $env(ENCINA_TPM_CELL)
                  set found 1
                  break
               }
   }
   regsub "/.:/" $ENCINA_TPM_CELL "" SHORT_CELL_NAME

   puts "Using Encina Cell $ENCINA_TPM_CELL"
   puts "Binaries found at:  $TPCC_BIN_DIR (TPCC_BIN_DIR)"
   puts "Current user: $USER"
   puts "type: showHelp for more help"

#-------------------------------------------------------------------
# createCell
#     procedure for creating and starting a cell
#     Takes the name of the logical volumes to be used for
#     the log and data
#-------------------------------------------------------------------

proc defineCell {} {
   global ENCINA_TPM_CELL
   global HOSTNAME UNSUPPORTED_NETIFS ADDITIONAL_ENV

   createVol ${HOSTNAME}_EcmLogVol  ecmlog
   createVol ${HOSTNAME}_EcmDataVol ecmdata

   puts "Creating cell $ENCINA_TPM_CELL"
   ecm create $ENCINA_TPM_CELL\
      -processPriority 30 \
      -logVolume ${HOSTNAME}_EcmLogVol \
      -environment "RPC_UNSUPPORTED_NETIFS=${UNSUPPORTED_NETIFS}
${ADDITIONAL_ENV}" \
      -dataVolumes ${HOSTNAME}_EcmDataVol
}

proc createCell {} {
   global ENCINA_TPM_CELL DCE_PWD
   global HOSTNAME UNSUPPORTED_NETIFS ADDITIONAL_ENV

   createVol ${HOSTNAME}_EcmLogVol  ecmlog
   createVol ${HOSTNAME}_EcmDataVol ecmdata

   puts "Creating cell $ENCINA_TPM_CELL"
   ecm create $ENCINA_TPM_CELL\
               -processPriority 30 \
               -logVolume ${HOSTNAME}_EcmLogVol \
      -environment "RPC_UNSUPPORTED_NETIFS=${UNSUPPORTED_NETIFS}
${ADDITIONAL_ENV}" \
      -dataVolumes ${HOSTNAME}_EcmDataVol

   puts "Starting cell $ENCINA_TPM_CELL"
   ecm start -adminPwd encina_admin -operatorPwd encina_operator \
      -dceAdminPrincipal cell_admin -dceAdminPwd $DCE_PWD -verbose

}

proc deleteEnm {name} {
   DO catch "enm stop ${name}"
   DO catch "enm erase ${name}"
   DO catch "enm delete ${name}"
   DO catch "lvol delete ${name}_EnmLogVol"
}

proc deleteNode {} {
   global HOSTNAME SHORT_CELL_NAME
   puts "Deleting node $HOSTNAME from cell ${SHORT_CELL_NAME}"
   deleteEnm $HOSTNAME
   puts "Removing /opt/encina*/${SHORT_CELL_NAME} directories for node
$HOSTNAME"
   exec rm -fr /opt/encinalocal/${SHORT_CELL_NAME}/node/${HOSTNAME}
   exec rm -fr /opt/encinamirror/${SHORT_CELL_NAME}/node/${HOSTNAME}
}
#-------------------------------------------------------------------
# proc createNode
#     procedure for creating and starting a node
#     Takes the logical volume to be used for the log
#-------------------------------------------------------------------

proc createNode {} {
   global ADDITIONAL_ENV
   global HOSTNAME ENCINA_BIN_PATH UNSUPPORTED_NETIFS
ENCINA_TPM_CELL

   deleteNode
   createVol ${HOSTNAME}_EnmLogVol  enmlog

   puts "Creating node ${HOSTNAME} "
   enm create ${HOSTNAME} -logVolume ${HOSTNAME}_EnmLogVol \
   -environment "TZ=CST6CDT
RPC_UNSUPPORTED_NETIFS=$UNSUPPORTED_NETIFS
ENCINA_TRPC_DS_PATH=/.:/tpccCell/trpc/ ${ADDITIONAL_ENV}" \
   -encinaBinariesPath $ENCINA_BIN_PATH
   puts "Starting node ${HOSTNAME} "
   enm start ${HOSTNAME} -myPwd encina_admin -cellManagerPrincipal
${ENCINA_TPM_CELL}/ecm -verbose
   enm modify ${HOSTNAME} -pingTimeout 25:00:00
   enm modify ${HOSTNAME} -pingInterval 20:00:00
}

proc go {node} {
   createCell
   createNode
   createServers  $node
}

#-------------------------------------------------------------------
# modifyAuthProt
#     procedure for modifying a server's authorization and protection
```

```
#    levels
#-------------------------------------------------------------------------

proc modifyAuthProt{serverType serverName authLevel protLevel} {
   global HOSTNAME

   $serverType modify ${HOSTNAME}-${serverName} \
                  -authorizationEnabled $authLevel \
                  -protectionLevel $protLevel
}


#-------------------------------------------------------------------------
# createInterfaces :
#    procedure for creating required interfaces
#-------------------------------------------------------------------------

proc createInterfaces {} {


   catch {interface create delivery}
   catch {interface create neworder}
   catch {interface create orderstatus}
   catch {interface create stocklevel}
   catch {interface create payment}
   catch {interface create tpccTrans}
}

#-------------------------------------------------------------------------
# create_and_start_mas :
#    procedure for creating and starting a mas server which
#    exports the interfaces tpmServer and tpmServer_admin
#    Takes the name of the mas server to be created
#-------------------------------------------------------------------------

proc mt_mas {node name db ifs threads} {
   global TPCC_BIN_DIR env DVRY_THREADS
   set schedule "MON_CONCURRENT_SHARED"
   if {$threads == 1} {
            set schedule "MON_SHARED"
   }
   if {$db == ""} {
      set cmdline "-no_db $ifs $schedule 1 dvry=$DVRY_THREADS db:null"
   } else {
      set cmdline "$ifs $schedule 1 dvry=$DVRY_THREADS db:$db"
   }
   set db_env "TWO_TASK=$db ORACLE_SID=$db
ORACLE_HOME=$env(ORACLE_HOME)"
   if {$threads == 1} {
      set mas_env ""
   } else {
      set e_tpool [expr $threads / 3]
      if {$e_tpool == 0} {
         set e_tpool 1
      }
      set a_tpool [expr $threads - $e_tpool]
      set mas_env "ENCINA_TPOOL_SIZE=$e_tpool
ENCINA_APPL_TPOOL_SIZE=$a_tpool"
      puts "mas env $mas_env"
   }
   set local_env "${db_env} ${mas_env}"
   set name [create_and_start_mas $node $name $cmdline $local_env]
   puts "Created and started $name"
   setTimeouts $name
   return $name
}

proc create_and_start_mas {node name cmdline masenv} {
   global SHORT_CELL_NAME env
   global ENCINA_TPM_CELL QUEUE_PARAM
   global HOSTNAME ADDITIONAL_MAS_ENV
   global TPCC_BIN_DIR DEFAULT_PAS
   global DEFAULT_TRACE UNSUPPORTED_NETIFS

   set mas_name ${node}-${name}
   puts "Creating MAS ${mas_name} on node $node"
   puts "    cmdline: '$cmdline'"
   puts "    Binary: ${TPCC_BIN_DIR}/serverMT_stats"
   mas create  ${mas_name} -node ${node} \
                  -executable ${TPCC_BIN_DIR}/serverMT_stats\
                  -commandLineArgs "$cmdline" \
                  -environment "TZ=CST6CDT
ENCINA_TRACE=${DEFAULT_TRACE}
RPC_UNSUPPORTED_NETIFS=$UNSUPPORTED_NETIFS
ENCINA_TRPC_DS_PATH=${ENCINA_TPM_CELL}/trpc/ $QUEUE_PARAM
${masenv} ${ADDITIONAL_MAS_ENV}" \
                  -interfaces {delivery tpccTrans} \
                  -paCount $DEFAULT_PAS -groupName staff

   # need this to do ema_attr stuff
   set server_principal $SHORT_CELL_NAME/server/$mas_name
#    puts "Adding principal $server_principal to encina_admin_group"
#    group add encina_admin_group -member $server_principal
   mas show $mas_name -setArray masInfo

   puts "Starting MAS ${mas_name}"
   mas start  ${mas_name} -myPwd encina_admin
   puts "Setting ACLs on all the interfaces"
   catch {exec acl_edit ${ENCINA_TPM_CELL}/ecm/interface/tpccTrans -m
user:encina_admin:x}
   catch {exec acl_edit ${ENCINA_TPM_CELL}/ecm/interface/delivery -m
user:encina_admin:x}
   return ${mas_name}
}


proc doMas {op node} {
   foreach mas [mas list] {
      if [ string match ${node}* $mas ] {
                  echo "mas $op $mas"
                  catch "mas $op $mas"
      } else {
#                   puts "Ignoring $mas"
               }
   }
   return
}
proc setTimeouts {mas} {
   mas modify $mas -authorizationEnabled 0
   puts "Setting timeouts for longRes on $mas"
   mas modify $mas -longTermResFirstUseTimeout 35:00:00
   mas modify $mas -longTermResPingTimeout 35:00:00
   mas modify $mas -longTermResPingInterval 30:00:00
   puts "Setting timeouts for ping on $mas"
   mas modify $mas -pingTimeout 25:00:00
   mas modify $mas -pingInitialTimeout 25:00:00
   mas modify $mas -pingInterval 20:00:00
   puts "Timeouts set"
}

proc createMas {node DB} {
   global XA_PAS
   createInterfaces
   dari_mas_all $node ${DB}-A $DB
   if {$XA_PAS > 0} {
      dari_xa $node ${DB}-A tpcc$DB 0xffff
      dari_remote $node ${DB}-A tpcc$DB
   }
   configMas  $node $DB
}

proc createMtMas {node {dataB "-check-"} } {
   global MT_PAS MT_DVRY_PAS DB_NAME
   global MT_THREADS DVRY_THREADS
   createInterfaces
   if { $dataB == "-check-" } { set dataB $DB_NAME }
   if { $dataB == "NULL" } { set dataB "" }

   set all_ifs 23

   if {$MT_DVRY_PAS > 0} {
      set name [mt_mas $node ${dataB}-dvr-MT $dataB 8 $DVRY_THREADS]
      mas modify $name -paCount $MT_DVRY_PAS
   } else {
      set all_ifs [expr $all_ifs + 8]
   }

   set name [mt_mas $node ${dataB}-all-MT $dataB $all_ifs $MT_THREADS]
   mas modify $name -paCount $MT_PAS
}

proc createTpcc {nodes {first 1}} {
   global client DB_NAME
   for {set i $first} {$i < $nodes + $first} {incr i 1} {
            createMtMas $client($i) $DB_NAME
   }
}

proc configAllMas {{udb "-check-"}} {
   global DB
   if { $udb == "NULL" } { set udb "" }
   foreach enm [enm list] {
      set dataB $udb
      if { $dataB == "-check-" } { set dataB $DB($enm) }
            createMtMas $enm $dataB
   }
}


proc createNullTpcc {nodes firstNode} {
```

```
   for {set i $firstNode} {$i < $nodes + $firstNode} {incr i 1} {
                set name "p16clnt"
                if {$i < 10} {
                   set name "${name}0"
                }
                set name "${name}$i"
                createMtMas $name ""
   }
}

# -environment "RPC_UNSUPPORTED_NETIFS=en0
ENCINA_TRPC_DS_PATH=/.:/tpccCell/trpc/"

return
```

B.3 AIX Parameters

RISC SYSTEM/6000 MODEL S7A

OS PARAMETERS

```
keylock       normal          State of system  keylock at boot time              False
maxbuf        200             Maximum number of pages in block I/O BUFFER CACHE
True
maxmbuf       0               Maximum Kbytes of real memory allowed for  MBUFS   True
maxuproc      40000           Maximum number of PROCESSES allowed per user
True
autorestart   false           Automatically REBOOT system after a crash        True
iostat        true            Continuously maintain DISK I/O history           True
realmem       33554432        Amount of usable physical memory in Kbytes       False
conslogin     enable          System Console  Login                            False
fwversion     IBM,19981110 (B) Firmware version and revision levels            False
maxpout       0               HIGH water mark for pending write I/Os per file  True
minpout       0               LOW water mark for pending write I/Os per file   True
fullcore      false           Enable full CORE dump                            True
pre430core    false           Use pre-430 style CORE dump                      True
rtasversion   1               Open Firmware RTAS version                       False
modelname     IBM,7017-S7A    Machine name                                     False
systemid      IBM,011092621   Hardware system identifier                       False
boottype      disk            N/A                                              False
SW_dist_intr  false           Enable SW distribution of interrupts
```

Appendix C: Database Setup

C.1 Database Creation Scripts

addfile.sh

```
#
# $Header: addfile.sh 7030100.1 96/05/02 10:30:04 plai Generic<base> $ Copyr (c) 1995 Oracle
#

#=====================================================================+
#       Copyright (c) 1996  OracleCorp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                 |
#                   All Rights Reserved                         |
#=====================================================================+
# FILENAME
#     addfile.sh
# DESCRIPTION
#     Add datafile to a tablespace.
# USAGE
#     addfile.sh <tablespace> <data file> <size>
#=====================================================================*/

FILE=`basename $2`

if [ -d ./outdir ]
then
  echo `date` > ./outdir/${FILE}.addf
fi

svrmgrl <<!
  connect internal
  alter tablespace $1 add datafile '$2' size $3 reuse;
  exit;
!

if [ -d ./outdir ]
then
  echo `date` >> ./outdir/${FILE}.addf
fi
fi
```

benchdb.sh

```
#
#=====================================================================+
#       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                 |
#                   All Rights Reserved                         |
#=====================================================================+
# FILENAME
#     benchdb.sh
# DESCRIPTION
#     Usage: benchdb.sh [options]
#          -n          do not create newtpcc database
#          -c          do not run catalog scripts
# MODIFIED
#     YPang for 5 nodes BB cluster.
#       Database size 11000 Warehouses.
#       Partitioned for 5 nodes.
#
#=====================================================================

if ( [ "${MULT}" = "" ] ); then
  echo You should only run this script from benchsetup.sh....
  exit
fi


while [ "$#" != "0" ]
do
  case $1 in
    -n) shift
       NO_CREATE="y"
       ;;
    -c) shift
       NO_CAT="y"
       ;;
     *) echo "Bad arg: $1"
       exit 1;
       ;;
  esac
done

#
# Create database if NO_CREATE unset
#

if [ "$NO_CREATE" = "" ]
then
svrmgrl <<!
 set echo on
 connect internal
 startup  pfile=$TPCC_ADMIN/p_create.ora  nomount
 create database tpcc controlfile reuse
    datafile '/dev/rroot' size 343M reuse
    logfile GROUP 1 ('/dev/rlog1i1a', '/dev/rlog1i1b') size 18230M reuse,
          GROUP 2 ('/dev/rlog2i1a', '/dev/rlog2i1b') size 18230M reuse
    maxinstances 5
    maxlogfiles 20
    maxdatafiles 3000;
 exit
!

# create threads 2-5 in parallel
crblog2.sh&
crblog3.sh&
crblog4.sh&
crblog5.sh&
wait
date

sqlplus system/manager @$TPCC_BUILD_SQL/pb_roll_cr.sql

#
#  Startup database with params file that includes new rollback segments
#

svrmgrl <<!
       connect internal
                shutdown
       startup pfile=$TPCC_ADMIN/p_build.ora
       connect system/manager

       create tablespace rol1 datafile '/dev/rrol1' size 255M reuse;
       create tablespace rol2 datafile '/dev/rrol2' size 255M reuse;
       create tablespace rol3 datafile '/dev/rrol3' size 255M reuse;
       create tablespace rol4 datafile '/dev/rrol4' size 255M reuse;
       create tablespace rol5 datafile '/dev/rrol5' size 255M reuse;

exit;
!

echo "----------------------------------------"
echo "benchdb.sh done at `date`"
```

benchsetup.sh

```
#
# $Header: benchsetup.sh 7030100.2 96/05/16 18:00:18  plai Generic<base> $ Copyr (c) 1995 Oracle
#
#
#=====================================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                 |
#                   All Rights Reserved                         |
#=====================================================================+
# FILENAME
#     benchsetup.sh
# DESCRIPTION
#     Usage: benchsetup.sh step_number
#          [step_number is a number between 0 and 14]
# MODIFIED
#     YPang for 5 nodes BB cluster.
#       Database size 11000 Warehouses.
#       Partitioned for 5 nodes.
#
#=====================================================================

set -x
echo $_

#OGMS_UP=`ps -fu ora | grep ogms | grep -v grep | wc -l`
#if [ ${OGMS_UP} -eq 0 ]; then
#   echo You must start GMS on all 5 nodes before running this script.
#   exit
#fi
#
#ORACLE_8=`echo ${ORACLE_HOME} | grep "8.0.5" | grep -v grep | wc -l`
#if [ ${ORACLE_8} -eq 0 ]; then
#   echo This script is intended for Oracle 8.0.5 only!
#   exit
#fi
#
#
#if ( [ "$1" = "" ] || [ $1 -lt 0 ] || [ $1 -gt 14 ] )
```

```
#then {
#   echo "usage: $0 step_number (-1 < step_number < 15)"
#   exit
#}
#fi
STEP=$1
echo "######## Starting step ${STEP}  `date`"


export BENCH_HOME=$ORACLE_HOME/bench/tpc
export BENCH_GEN=$ORACLE_HOME/bench/gen
export GEN_SQL=$BENCH_GEN/sql/12n
export TPCC_SOURCE=$BENCH_HOME/tpcc/source
export TPCC_LOADER=$BENCH_HOME/tpcc/loader
export TPCC_SQL=$BENCH_HOME/tpcc/sql
export TPCC_STORE=$BENCH_HOME/tpcc/blocks
export TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
export TPCC_UTILS=$TPCC_SCRIPTS/utils
export TPCC_ADMIN=$TPCC_SCRIPTS/5s7a/admin
export AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
export BUILD_SQL=sql
export OUTDIR=build_out
export PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export LDIR=/tmp/data

export MULT=11000

if echo "\c" | grep c >/dev/null 2>&1; then
   N='-n'
else
   C='\c'
fi
export N C


if [ ! -d $OUTDIR ]
then
  mkdir $OUTDIR
fi

#-------------------------------------------------------------
# Step 0: Do nothing (used to set environment)
#-------------------------------------------------------------
if [ ${STEP} -eq 0 ]; then
  echo "Setting bench environment on `hostname -s`...."
fi

#-------------------------------------------------------------
# Step 1: Create database.
#-------------------------------------------------------------
if [ ${STEP} -eq 1 ]; then
  benchdb.sh
fi

#-------------------------------------------------------------
# Step 2: Create private rollbacks and createdictionay views
#-------------------------------------------------------------
if [ ${STEP} -eq 2 ]; then
  switchlog.sh
  cr_private_roll.sh
  svrmgrl <<!
    set echo off;
    connect sys/change_on_install;
    @?/rdbms/admin/catalog;
    @?/rdbms/admin/catproc;
    @?/rdbms/admin/catparr;
    connect system/manager;
    @?/sqlplus/admin/pupbld;
    exit;
!
fi

#-------------------------------------------------------------
# Step 3: Create the tables
#-------------------------------------------------------------
if [ ${STEP} -eq 3 ]; then
 cr_all_ts.sh
 sqlplus system/manager @$BUILD_SQL/tpcc_tab
fi

#-------------------------------------------------------------
# Step 4: Load history, new-order, order, order-line tables
#-------------------------------------------------------------
if [ ${STEP} -eq 4 ]; then
  pldhist.sh
  pldnord.sh
  pld_ord.sh
  switchlog.sh
fi

#-------------------------------------------------------------
# Step 5: Create customer, stock tables as hashclusters
#-------------------------------------------------------------
if [ ${STEP} -eq 5 ]; then
  sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_tab2 &
  sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_tab3 &
  wait
  switchlog.sh
fi

#-------------------------------------------------------------
# Step 6: Load warehouse, district, item tables
#-------------------------------------------------------------
if [ ${STEP} -eq 6 ]; then
  echo Loading warehouse `date`
  tpccload -M $MULT -w
  echo Loading district `date`
  tpccload -M $MULT -d
  echo Loading items `date`
  tpccload -M $MULT -i
fi

#-------------------------------------------------------------
# Step 7: Load customer table in parallel using all nodes
#-------------------------------------------------------------

if [ ${STEP} -eq 7 ]; then

  pload_cust.sh

fi
#-------------------------------------------------------------
# Step 8: Load stock tables in parallel using all nodes
#-------------------------------------------------------------

if [ ${STEP} -eq 8 ]; then

  pload_stk.sh

fi

#-------------------------------------------------------------
#  Step 9: Deallocate unused space and get ready for index
#  index creation
#-------------------------------------------------------------

if [ ${STEP} -eq 9 ]; then

  switchlog.sh

sqlplus tpcc/tpcc <<!
  alter table history   deallocate unused;
  alter table orders    deallocate unused;
  alter table new_order  deallocate unused;
  alter table order_line deallocate unused;
  alter cluster scluster deallocate unused;
  alter cluster ccluster deallocate unused;
  exit;
!

sqlplus system/manager <<!
  alter user tpcc temporary tablespace temp;
  quit;
!

svrmgrl <<!
  connect internal
  alter tablespace temp
    default storage (initial 1998M next 1998M pctincrease 0);
  exit;
!

fi

#-------------------------------------------------------------
# Step 10: Create first set of indexes
#-------------------------------------------------------------

if [ ${STEP} -eq 10 ]; then

 sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_ix1

fi

#-------------------------------------------------------------
# Step 11: Create 2nd set of indexes & coalesce
#-------------------------------------------------------------

if [ ${STEP} -eq 11 ]; then

 sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_ix2

svrmgrl <<!
  connect internal
  alter tablespace temp
    default storage (initial 20K next 20K pctincrease 50);
  exit;
!

sqlplus tpcc/tpcc <<!
  alter index iorders     deallocate unused;
  alter index iorders2    deallocate unused;
  alter index inew_order  deallocate unused;
  alter index iorder_line deallocate unused;
  exit;
!

sqlplus system/manager <<!
```

```
    alter tablespace ior1_n1 coalesce;
    alter tablespace ior1_n2 coalesce;
    alter tablespace ior1_n3 coalesce;
    alter tablespace ior1_n4 coalesce;
    alter tablespace ior1_n5 coalesce;
    alter tablespace ior2_n1 coalesce;
    alter tablespace ior2_n2 coalesce;
    alter tablespace ior2_n3 coalesce;
    alter tablespace ior2_n4 coalesce;
    alter tablespace ior2_n5 coalesce;
    alter tablespace iorl_n1 coalesce;
    alter tablespace iorl_n2 coalesce;
    alter tablespace iorl_n3 coalesce;
    alter tablespace iorl_n4 coalesce;
    alter tablespace iorl_n5 coalesce;
  exit;
!
fi

#-------------------------------------------------------
# Step 12: Analyze tables and indexes
#-------------------------------------------------------

if [ ${STEP} -eq 12 ]; then

sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_ana_tab
sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_ana_idx

fi

#-------------------------------------------------------
# Step 13: Create tables for processing benchmark
#       and create stored procesures.
#-------------------------------------------------------

if [ ${STEP} -eq 13 ]; then

#
#  Create table for processing benchmark results
#

sqlplus sys/change_on_install @$GEN_SQL/orst_cre
sqlplus sys/change_on_install @$TPCC_SQL/c_stat
sqlplus sys/change_on_install @$GEN_SQL/pst_c

#
#  Create stored procedures
#

sqlplus tpcc/tpcc @$TPCC_STORE/pay
sqlplus tpcc/tpcc @$TPCC_STORE/views

#
#  Get some statistics
#

$TPCC_SCRIPTS/utils/ext_all.sh > ${OUTDIR}/ext_all.out 2>&1

$TPCC_SCRIPTS/utils/space_init.sh
$TPCC_SCRIPTS/utils/space_get.sh 30000 2592
$TPCC_SCRIPTS/utils/space_rpt.sh ${OUTDIR}/space.rpt

sqlplus system/manager <<!
  alter user tpcc temporary tablespace system;
  quit;
!

sqlplus sys/change_on_install <<!
  grant execute on dbms_lock to public;
  grant execute on dbms_pipe to public;
  grant select on v_\$parameter to public;
  quit;
!

sqlplus tpcc/tpcc @$AUDIT_SQL/plsql_mon


sqlplus tpcc/tpcc @$AUDIT_SQL/cre_tab

switchlog.sh

fi

#-------------------------------------------------------
# Step 14: Set next extents to smaller values
#-------------------------------------------------------

if [ ${STEP} -eq 14 ]; then

${TPCC_SCRIPTS}/5s7a/tools/set_next_extents

#dml.sh   # done in set_next_extents

svrmgrl <<!
  set echo off;
  connect sys/change_on_install;
  @?/rdbms/admin/catparr;
  exit;
!
fi

echo "######## Ended step ${STEP} `date`"
```

dml.sh

```
#
# $Header: dml.sh 7030100.1 96/05/02 10:22:52 plai Generic<base> $ Copyr (c) 1995 Oracle
#

#============================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#============================================================+
# FILENAME
#     dml.sh
# DESCRIPTION
#     Disable table locks for TPC-C tables.
# USAGE
#     dml.sh
#============================================================*/

sqlplus tpcc/tpcc <<!
  alter table warehouse  disable table lock;
  alter table district   disable table lock;
  alter table customer   disable table lock;
  alter table history    disable table lock;
  alter table item       disable table lock;
  alter table stock      disable table lock;
  alter table orders     disable table lock;
  alter table new_order  disable table lock;
  alter table order_line disable table lock;
  quit;
!
```

cr_all_ts.sh

```
#Creating tablespaces in parallel on all nodes.

    cr_wdi_ts.sh
    cr_hist_ts.sh
    cr_cust_ts.sh
    cr_icu1_ts.sh
    cr_icu2_ts.sh
    cr_nord_ts.sh
    cr_inor_ts.sh
    cr_ords_ts.sh
    cr_iord1_ts.sh
    cr_iord2_ts.sh
    cr_ordl_ts.sh
    cr_iorl_ts.sh
    cr_stoc_ts.sh
    cr_istk_ts.sh
    cr_temp_ts.sh
```

cr_cust_ts.sh

```
############################################################
#
# FILE:  cr_cust_ts.sh
#
#######################################

. ~/.profile

svrmgrl <<!
 set echo on
 connect system/manager
 create tablespace cust datafile '/dev/rcust001' size 771M reuse;
 exit;
!
   cr_n1_cust_ts.sh  &
   cr_n2_cust_ts.sh  &
   cr_n3_cust_ts.sh  &
   cr_n4_cust_ts.sh  &
   cr_n5_cust_ts.sh  &

   wait
```

Cr_hist_ts.sh

```
 set echo on
 connect system/manager
 create tablespace hist_n1_1 datafile   '/dev/rhist001'  size 1039M reuse;
 create tablespace hist_n2_1 datafile   '/dev/rhist009'  size 1039M reuse;
 create tablespace hist_n3_1 datafile   '/dev/rhist017'  size 1039M reuse;
 create tablespace hist_n4_1 datafile   '/dev/rhist025'  size 1039M reuse;
```

```
  create tablespace hist_n5_1 datafile   '/dev/rhist033'  size 1039M reuse;
  exit;
!
  addfile.sh hist_n1_1  /dev/rhist002 1039M &
  addfile.sh hist_n1_1  /dev/rhist003 1039M &
  addfile.sh hist_n1_1  /dev/rhist004 1039M &
  addfile.sh hist_n1_1  /dev/rhist005 1039M &
  addfile.sh hist_n1_1  /dev/rhist006 1039M &
  addfile.sh hist_n1_1  /dev/rhist007 1039M &
  addfile.sh hist_n1_1  /dev/rhist008 1039M &
  addfile.sh hist_n2_1  /dev/rhist010 1039M &
  addfile.sh hist_n2_1  /dev/rhist011 1039M &
  addfile.sh hist_n2_1  /dev/rhist012 1039M &
  addfile.sh hist_n2_1  /dev/rhist013 1039M &
  addfile.sh hist_n2_1  /dev/rhist014 1039M &
  addfile.sh hist_n2_1  /dev/rhist015 1039M &
  addfile.sh hist_n2_1  /dev/rhist016 1039M &
  addfile.sh hist_n3_1  /dev/rhist018 1039M &
  addfile.sh hist_n3_1  /dev/rhist019 1039M &
  addfile.sh hist_n3_1  /dev/rhist020 1039M &
  addfile.sh hist_n3_1  /dev/rhist021 1039M &
  addfile.sh hist_n3_1  /dev/rhist022 1039M &
  addfile.sh hist_n3_1  /dev/rhist023 1039M &
  addfile.sh hist_n3_1  /dev/rhist024 1039M &
  addfile.sh hist_n4_1  /dev/rhist026 1039M &
  addfile.sh hist_n4_1  /dev/rhist027 1039M &
  addfile.sh hist_n4_1  /dev/rhist028 1039M &
  addfile.sh hist_n4_1  /dev/rhist029 1039M &
  addfile.sh hist_n4_1  /dev/rhist030 1039M &
  addfile.sh hist_n4_1  /dev/rhist031 1039M &
  addfile.sh hist_n4_1  /dev/rhist032 1039M &
  addfile.sh hist_n5_1  /dev/rhist034 1039M &
  addfile.sh hist_n5_1  /dev/rhist035 1039M &
  addfile.sh hist_n5_1  /dev/rhist036 1039M &
  addfile.sh hist_n5_1  /dev/rhist037 1039M &
  addfile.sh hist_n5_1  /dev/rhist038 1039M &
  addfile.sh hist_n5_1  /dev/rhist039 1039M &
  addfile.sh hist_n5_1  /dev/rhist040 1039M &
wait
echo "ending $0: `date`"
```

cr_icu1_ts.sh

```
################################################################################
#
#   FILE:  cr_icu1_ts.sh
#
#         Createicus tablespace
#
################################################################################

. ~/.profile

svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace icu1 datafile '/dev/ricu1001' size 378M reuse;
  exit;
!

addfile.sh icu1  /dev/ricu1002 378M &
addfile.sh icu1  /dev/ricu1003 378M &
addfile.sh icu1  /dev/ricu1004 378M &
addfile.sh icu1  /dev/ricu1005 378M &
addfile.sh icu1  /dev/ricu1006 378M &
addfile.sh icu1  /dev/ricu1007 378M &
addfile.sh icu1  /dev/ricu1008 378M &
addfile.sh icu1  /dev/ricu1009 378M &
addfile.sh icu1  /dev/ricu1010 378M &
addfile.sh icu1  /dev/ricu1011 378M &
addfile.sh icu1  /dev/ricu1012 378M &
addfile.sh icu1  /dev/ricu1013 378M &
addfile.sh icu1  /dev/ricu1014 378M &
addfile.sh icu1  /dev/ricu1015 378M &
addfile.sh icu1  /dev/ricu1016 378M &
addfile.sh icu1  /dev/ricu1017 378M &
addfile.sh icu1  /dev/ricu1018 378M &
addfile.sh icu1  /dev/ricu1019 378M &
addfile.sh icu1  /dev/ricu1020 378M &
addfile.sh icu1  /dev/ricu1021 378M &
addfile.sh icu1  /dev/ricu1022 378M &
addfile.sh icu1  /dev/ricu1023 378M &
addfile.sh icu1  /dev/ricu1024 378M &
addfile.sh icu1  /dev/ricu1025 378M &

wait
```

Cr_icu2_ts.sh

```
set -x

date

. ~/.profile

svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace icu2 datafile '/dev/ricu2001' size 339M reuse;
  exit;
!

addfile.sh icu2  /dev/ricu2002 339M &
addfile.sh icu2  /dev/ricu2003 339M &
addfile.sh icu2  /dev/ricu2004 339M &
addfile.sh icu2  /dev/ricu2005 339M &
addfile.sh icu2  /dev/ricu2006 339M &
addfile.sh icu2  /dev/ricu2007 339M &
addfile.sh icu2  /dev/ricu2008 339M &
addfile.sh icu2  /dev/ricu2009 339M &
addfile.sh icu2  /dev/ricu2010 339M &
addfile.sh icu2  /dev/ricu2011 339M &
addfile.sh icu2  /dev/ricu2012 339M &
addfile.sh icu2  /dev/ricu2013 339M &
addfile.sh icu2  /dev/ricu2014 339M &
addfile.sh icu2  /dev/ricu2015 339M &
addfile.sh icu2  /dev/ricu2016 339M &
addfile.sh icu2  /dev/ricu2017 339M &
addfile.sh icu2  /dev/ricu2018 339M &
addfile.sh icu2  /dev/ricu2019 339M &
addfile.sh icu2  /dev/ricu2020 339M &
addfile.sh icu2  /dev/ricu2021 339M &
addfile.sh icu2  /dev/ricu2022 339M &
addfile.sh icu2  /dev/ricu2023 339M &
addfile.sh icu2  /dev/ricu2024 339M &
addfile.sh icu2  /dev/ricu2025 339M &
addfile.sh icu2  /dev/ricu2026 339M &
addfile.sh icu2  /dev/ricu2027 339M &
addfile.sh icu2  /dev/ricu2028 339M &
addfile.sh icu2  /dev/ricu2029 339M &
addfile.sh icu2  /dev/ricu2030 339M &
addfile.sh icu2  /dev/ricu2031 339M &
addfile.sh icu2  /dev/ricu2032 339M &
addfile.sh icu2  /dev/ricu2033 339M &
addfile.sh icu2  /dev/ricu2034 339M &
addfile.sh icu2  /dev/ricu2035 339M &
addfile.sh icu2  /dev/ricu2036 339M &
addfile.sh icu2  /dev/ricu2037 339M &
addfile.sh icu2  /dev/ricu2038 339M &
addfile.sh icu2  /dev/ricu2039 339M &
addfile.sh icu2  /dev/ricu2040 339M &
addfile.sh icu2  /dev/ricu2041 339M &
addfile.sh icu2  /dev/ricu2042 339M &
addfile.sh icu2  /dev/ricu2043 339M &
addfile.sh icu2  /dev/ricu2044 339M &
addfile.sh icu2  /dev/ricu2045 339M &
addfile.sh icu2  /dev/ricu2046 339M &
addfile.sh icu2  /dev/ricu2047 339M &
addfile.sh icu2  /dev/ricu2048 339M &
addfile.sh icu2  /dev/ricu2049 339M &
addfile.sh icu2  /dev/ricu2050 339M &
addfile.sh icu2  /dev/ricu2051 339M &
addfile.sh icu2  /dev/ricu2052 339M &
addfile.sh icu2  /dev/ricu2053 339M &
addfile.sh icu2  /dev/ricu2054 339M &
addfile.sh icu2  /dev/ricu2055 339M &
addfile.sh icu2  /dev/ricu2056 339M &
addfile.sh icu2  /dev/ricu2057 339M &
addfile.sh icu2  /dev/ricu2058 339M &
addfile.sh icu2  /dev/ricu2059 339M &
addfile.sh icu2  /dev/ricu2060 339M &

wait
date
echo "Done"
```

Cr_inor_ts.sh

```
. ~/.profile
echo "Starting $0: `date`"
svrmgrl <<!
  set echo on
  connect tpcc/tpcc
  drop index inew_order;
  connect system/manager;
  drop tablespace inor_n1_1 including contents;
  drop tablespace inor_n2_1 including contents;
  drop tablespace inor_n3_1 including contents;
  drop tablespace inor_n4_1 including contents;
  drop tablespace inor_n5_1 including contents;
  create tablespace inor_n1_1 datafile   '/dev/rinor001'  size 335M reuse;
  create tablespace inor_n2_1 datafile   '/dev/rinor011'  size 335M reuse;
  create tablespace inor_n3_1 datafile   '/dev/rinor021'  size 335M reuse;
  create tablespace inor_n4_1 datafile   '/dev/rinor031'  size 335M reuse;
  create tablespace inor_n5_1 datafile   '/dev/rinor041'  size 335M reuse;
  exit;
!
  addfile.sh inor_n1_1  /dev/rinor002 335M &
  addfile.sh inor_n1_1  /dev/rinor003 335M &
  addfile.sh inor_n1_1  /dev/rinor004 335M &
  addfile.sh inor_n1_1  /dev/rinor005 335M &
  addfile.sh inor_n1_1  /dev/rinor006 335M &
  addfile.sh inor_n1_1  /dev/rinor007 335M &
  addfile.sh inor_n1_1  /dev/rinor008 335M &
```

```
  addfile.sh inor_n1_1  /dev/rinor009 335M &
  addfile.sh inor_n1_1  /dev/rinor010 335M &
  addfile.sh inor_n2_1  /dev/rinor012 335M &
  addfile.sh inor_n2_1  /dev/rinor013 335M &
  addfile.sh inor_n2_1  /dev/rinor014 335M &
  addfile.sh inor_n2_1  /dev/rinor015 335M &
  addfile.sh inor_n2_1  /dev/rinor016 335M &
  addfile.sh inor_n2_1  /dev/rinor017 335M &
  addfile.sh inor_n2_1  /dev/rinor018 335M &
  addfile.sh inor_n2_1  /dev/rinor019 335M &
  addfile.sh inor_n2_1  /dev/rinor020 335M &
  addfile.sh inor_n3_1  /dev/rinor022 335M &
  addfile.sh inor_n3_1  /dev/rinor023 335M &
  addfile.sh inor_n3_1  /dev/rinor024 335M &
  addfile.sh inor_n3_1  /dev/rinor025 335M &
  addfile.sh inor_n3_1  /dev/rinor026 335M &
  addfile.sh inor_n3_1  /dev/rinor027 335M &
  addfile.sh inor_n3_1  /dev/rinor028 335M &
  addfile.sh inor_n3_1  /dev/rinor029 335M &
  addfile.sh inor_n3_1  /dev/rinor030 335M &
  addfile.sh inor_n4_1  /dev/rinor032 335M &
  addfile.sh inor_n4_1  /dev/rinor033 335M &
  addfile.sh inor_n4_1  /dev/rinor034 335M &
  addfile.sh inor_n4_1  /dev/rinor035 335M &
  addfile.sh inor_n4_1  /dev/rinor036 335M &
  addfile.sh inor_n4_1  /dev/rinor037 335M &
  addfile.sh inor_n4_1  /dev/rinor038 335M &
  addfile.sh inor_n4_1  /dev/rinor039 335M &
  addfile.sh inor_n4_1  /dev/rinor040 335M &
  addfile.sh inor_n5_1  /dev/rinor042 335M &
  addfile.sh inor_n5_1  /dev/rinor043 335M &
  addfile.sh inor_n5_1  /dev/rinor044 335M &
  addfile.sh inor_n5_1  /dev/rinor045 335M &
  addfile.sh inor_n5_1  /dev/rinor046 335M &
  addfile.sh inor_n5_1  /dev/rinor047 335M &
  addfile.sh inor_n5_1  /dev/rinor048 335M &
  addfile.sh inor_n5_1  /dev/rinor049 335M &
  addfile.sh inor_n5_1  /dev/rinor050 335M &
wait
echo "ending $0: `date`"
```

Cr_iord1_ts.sh

```
set -x
date
cr_n1_iord1_ts.sh &
cr_n2_iord1_ts.sh &
cr_n3_iord1_ts.sh &
cr_n4_iord1_ts.sh &
cr_n5_iord1_ts.sh &
wait
date
echo "Done"
```

Cr_iord2_ts.sh

```
set -x
date
cr_n1_iord2_ts.sh
cr_n2_iord2_ts.sh
cr_n3_iord2_ts.sh
cr_n4_iord2_ts.sh
cr_n5_iord2_ts.sh
#wait
date
echo "Done"
```

Cr_ordl_ts.sh

```
#############################################################
#
# FILE:  cr_iorl_ts.sh
#
#####################################

date
cr_n1_iorl_ts.sh  &
cr_n2_iorl_ts.sh  &
cr_n3_iorl_ts.sh  &
cr_n4_iorl_ts.sh  &
cr_n5_iorl_ts.sh  &
wait
date
exit
```

cr_istk_ts.sh

```
#############################################################################
#
#  FILE:  cr_istk_ts.sh
#
#        Createistk tablespace
#
#############################################################################

svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace istk datafile '/dev/ristk001'  size 1126M reuse;
  exit;
!
cr_n1_istk_ts.sh   &
cr_n2_istk_ts.sh   &
cr_n3_istk_ts.sh   &
cr_n4_istk_ts.sh   &
cr_n5_istk_ts.sh   &
addfile.sh istk  /dev/ristk021 1126M &
addfile.sh istk  /dev/ristk022 1126M &
addfile.sh istk  /dev/ristk023 1126M &
addfile.sh istk  /dev/ristk024 1126M &
addfile.sh istk  /dev/ristk025 1126M &

wait
```

cr_n1_cust_ts.sh

```
#############################################################################
#
#  FILE:  cr_n1_cust_ts.sh
#
#        Create cust tablespace on node1
#
#############################################################################

. ~/.profile

addfile.sh cust  /dev/rcust002 771M &
addfile.sh cust  /dev/rcust003 771M &
addfile.sh cust  /dev/rcust004 771M &
addfile.sh cust  /dev/rcust005 771M &
addfile.sh cust  /dev/rcust006 771M &
addfile.sh cust  /dev/rcust007 771M &
addfile.sh cust  /dev/rcust008 771M &
addfile.sh cust  /dev/rcust009 771M &
addfile.sh cust  /dev/rcust010 771M &
addfile.sh cust  /dev/rcust011 771M &
addfile.sh cust  /dev/rcust012 771M &
addfile.sh cust  /dev/rcust013 771M &
addfile.sh cust  /dev/rcust014 771M &
addfile.sh cust  /dev/rcust015 771M &
addfile.sh cust  /dev/rcust016 771M &
addfile.sh cust  /dev/rcust017 771M &
addfile.sh cust  /dev/rcust018 771M &
addfile.sh cust  /dev/rcust019 771M &
addfile.sh cust  /dev/rcust020 771M &
addfile.sh cust  /dev/rcust021 771M &
addfile.sh cust  /dev/rcust022 771M &

wait

addfile.sh cust  /dev/rcust023 771M &
addfile.sh cust  /dev/rcust024 771M &
addfile.sh cust  /dev/rcust025 771M &
addfile.sh cust  /dev/rcust026 771M &
addfile.sh cust  /dev/rcust027 771M &
addfile.sh cust  /dev/rcust028 771M &
addfile.sh cust  /dev/rcust029 771M &
addfile.sh cust  /dev/rcust030 771M &
addfile.sh cust  /dev/rcust031 771M &
addfile.sh cust  /dev/rcust032 771M &
addfile.sh cust  /dev/rcust033 771M &
addfile.sh cust  /dev/rcust034 771M &
addfile.sh cust  /dev/rcust035 771M &
addfile.sh cust  /dev/rcust036 771M &
addfile.sh cust  /dev/rcust037 771M &
addfile.sh cust  /dev/rcust038 771M &
addfile.sh cust  /dev/rcust039 771M &
addfile.sh cust  /dev/rcust040 771M &
addfile.sh cust  /dev/rcust041 771M &
addfile.sh cust  /dev/rcust042 771M &
addfile.sh cust  /dev/rcust043 771M &
addfile.sh cust  /dev/rcust044 771M &

wait

addfile.sh cust  /dev/rcust045 771M &
addfile.sh cust  /dev/rcust046 771M &
addfile.sh cust  /dev/rcust047 771M &
addfile.sh cust  /dev/rcust048 771M &
addfile.sh cust  /dev/rcust049 771M &
addfile.sh cust  /dev/rcust050 771M &
addfile.sh cust  /dev/rcust051 771M &
addfile.sh cust  /dev/rcust052 771M &
addfile.sh cust  /dev/rcust053 771M &
addfile.sh cust  /dev/rcust054 771M &
addfile.sh cust  /dev/rcust055 771M &
addfile.sh cust  /dev/rcust056 771M &
addfile.sh cust  /dev/rcust057 771M &
addfile.sh cust  /dev/rcust058 771M &
addfile.sh cust  /dev/rcust059 771M &
```

```
addfile.sh cust  /dev/rcust060 771M &
addfile.sh cust  /dev/rcust061 771M &
addfile.sh cust  /dev/rcust062 771M &
addfile.sh cust  /dev/rcust063 771M &
addfile.sh cust  /dev/rcust064 771M &
addfile.sh cust  /dev/rcust065 771M &
addfile.sh cust  /dev/rcust066 771M &

wait

addfile.sh cust  /dev/rcust067 771M &
addfile.sh cust  /dev/rcust068 771M &
addfile.sh cust  /dev/rcust069 771M &
addfile.sh cust  /dev/rcust070 771M &
addfile.sh cust  /dev/rcust071 771M &
addfile.sh cust  /dev/rcust072 771M &
addfile.sh cust  /dev/rcust073 771M &
addfile.sh cust  /dev/rcust074 771M &
addfile.sh cust  /dev/rcust075 771M &
addfile.sh cust  /dev/rcust076 771M &
addfile.sh cust  /dev/rcust077 771M &
addfile.sh cust  /dev/rcust078 771M &
addfile.sh cust  /dev/rcust079 771M &
addfile.sh cust  /dev/rcust080 771M &
addfile.sh cust  /dev/rcust081 771M &
addfile.sh cust  /dev/rcust082 771M &
addfile.sh cust  /dev/rcust083 771M &
addfile.sh cust  /dev/rcust084 771M &
addfile.sh cust  /dev/rcust085 771M &
addfile.sh cust  /dev/rcust086 771M &
addfile.sh cust  /dev/rcust087 771M &
addfile.sh cust  /dev/rcust088 771M &

wait
```

Cr_n1_iord1_ts.sh

```
################################################################################
#
#   FILE:  cr_n1_iord_ts.sh
#
#         Create ior1 and ior2 tablespaces on node1
#
################################################################################

. ~/.profile
set -x
svrmgrl <<!
       set echo on
       connect system/manager
       create tablespace ior1_n1 datafile '/dev/rior1001' size 607M reuse;
       exit;
!

addfile.sh ior1_n1  /dev/rior1002 607M &
addfile.sh ior1_n1  /dev/rior1003 607M &
addfile.sh ior1_n1  /dev/rior1004 607M &
addfile.sh ior1_n1  /dev/rior1005 607M &
addfile.sh ior1_n1  /dev/rior1006 607M &
addfile.sh ior1_n1  /dev/rior1007 607M &
addfile.sh ior1_n1  /dev/rior1008 607M &
addfile.sh ior1_n1  /dev/rior1009 607M &
addfile.sh ior1_n1  /dev/rior1010 607M &

wait
date
echo "Done: cr_n1_iord1_ts.sh"
```

cr_n1_iord_ts.sh

```
################################################################################
#
#   FILE:  cr_n1_iord_ts.sh
#
#         Create ior1 and ior2 tablespaces on node1
#
################################################################################

. ~/.profile
set -x

svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace ior2_n1 datafile    '/dev/rior2001'  size 511M reuse;
  exit;
!

addfile.sh ior2_n1  /dev/rior2002 511M &
addfile.sh ior2_n1  /dev/rior2003 511M &
addfile.sh ior2_n1  /dev/rior2004 511M &
addfile.sh ior2_n1  /dev/rior2005 511M &
addfile.sh ior2_n1  /dev/rior2006 511M &
addfile.sh ior2_n1  /dev/rior2007 511M &
addfile.sh ior2_n1  /dev/rior2008 511M &
addfile.sh ior2_n1  /dev/rior2009 511M &
addfile.sh ior2_n1  /dev/rior2010 511M &
addfile.sh ior2_n1  /dev/rior2011 511M &
addfile.sh ior2_n1  /dev/rior2012 511M &
addfile.sh ior2_n1  /dev/rior2013 511M &
addfile.sh ior2_n1  /dev/rior2014 511M &
addfile.sh ior2_n1  /dev/rior2015 511M &
addfile.sh ior2_n1  /dev/rior2016 511M &
addfile.sh ior2_n1  /dev/rior2017 511M &
addfile.sh ior2_n1  /dev/rior2018 511M &
addfile.sh ior2_n1  /dev/rior2019 511M &
addfile.sh ior2_n1  /dev/rior2020 511M &

wait
date
echo "Done: cr_n1_iord2_ts.sh"
```

cr_n1_iorl_ts.sh

```
################################################################################
#
#   FILE:  cr_n1_iorl_ts.sh
#
#         Create iorl tablespace  on node1
#
################################################################################

. ~/.profile

svrmgrl <<!
       set echo on
       connect system/manager

       create tablespace iorl_n1_1 datafile    '/dev/riorl001'  size 2127M reuse;
       create tablespace iorl_n1_2 datafile    '/dev/riorl012'  size 2127M reuse;

       exit;
!

addfile.sh iorl_n1_1  /dev/riorl002 2127M &
addfile.sh iorl_n1_1  /dev/riorl003 2127M &
addfile.sh iorl_n1_1  /dev/riorl004 2127M &
addfile.sh iorl_n1_1  /dev/riorl005 2127M &
addfile.sh iorl_n1_1  /dev/riorl006 2127M &
addfile.sh iorl_n1_1  /dev/riorl007 2127M &
addfile.sh iorl_n1_1  /dev/riorl008 2127M &
addfile.sh iorl_n1_1  /dev/riorl009 2127M &
addfile.sh iorl_n1_1  /dev/riorl010 2127M &
addfile.sh iorl_n1_1  /dev/riorl011 2127M &

wait

addfile.sh iorl_n1_2  /dev/riorl013 2127M &
addfile.sh iorl_n1_2  /dev/riorl014 2127M &
addfile.sh iorl_n1_2  /dev/riorl015 2127M &
addfile.sh iorl_n1_2  /dev/riorl016 2127M &
addfile.sh iorl_n1_2  /dev/riorl017 2127M &
addfile.sh iorl_n1_2  /dev/riorl018 2127M &
addfile.sh iorl_n1_2  /dev/riorl019 2127M &
addfile.sh iorl_n1_2  /dev/riorl020 2127M &
addfile.sh iorl_n1_2  /dev/riorl021 2127M &
addfile.sh iorl_n1_2  /dev/riorl022 2127M &

wait
```

cr_n1_iorl_ts.sh

```
################################################################################
#
#   FILE:  cr_n1_iorl_ts.sh
#
#         Create iorl tablespace  on node1
#
################################################################################

. ~/.profile

svrmgrl <<!
       set echo on
       connect system/manager

       create tablespace iorl_n1_1 datafile    '/dev/riorl001'  size 2127M reuse;
       create tablespace iorl_n1_2 datafile    '/dev/riorl012'  size 2127M reuse;

       exit;
!

addfile.sh iorl_n1_1  /dev/riorl002 2127M &
addfile.sh iorl_n1_1  /dev/riorl003 2127M &
addfile.sh iorl_n1_1  /dev/riorl004 2127M &
addfile.sh iorl_n1_1  /dev/riorl005 2127M &
addfile.sh iorl_n1_1  /dev/riorl006 2127M &
addfile.sh iorl_n1_1  /dev/riorl007 2127M &
addfile.sh iorl_n1_1  /dev/riorl008 2127M &
addfile.sh iorl_n1_1  /dev/riorl009 2127M &
addfile.sh iorl_n1_1  /dev/riorl010 2127M &
addfile.sh iorl_n1_1  /dev/riorl011 2127M &

wait
```

Cr_n1_iorl2_ts.sh

```
addfile.sh iorl_n1_2  /dev/riorl013 2127M &
addfile.sh iorl_n1_2  /dev/riorl014 2127M &
addfile.sh iorl_n1_2  /dev/riorl015 2127M &
addfile.sh iorl_n1_2  /dev/riorl016 2127M &
addfile.sh iorl_n1_2  /dev/riorl017 2127M &
addfile.sh iorl_n1_2  /dev/riorl018 2127M &
addfile.sh iorl_n1_2  /dev/riorl019 2127M &
addfile.sh iorl_n1_2  /dev/riorl020 2127M &
addfile.sh iorl_n1_2  /dev/riorl021 2127M &
addfile.sh iorl_n1_2  /dev/riorl022 2127M &

wait
```

cr_n1_iorl_ts.sh

```
#################################################################################
#
#   FILE: cr_n1_iorl_ts.sh
#
#         Create iorl tablespace  on node1
#
#################################################################################

. ~/.profile

svrmgrl <<!
        set echo on
        connect system/manager

        create tablespace iorl_n1_1 datafile    '/dev/riorl001'  size 2127M reuse;
        create tablespace iorl_n1_2 datafile    '/dev/riorl012'  size 2127M reuse;

        exit;
!

addfile.sh iorl_n1_1  /dev/riorl002 2127M &
addfile.sh iorl_n1_1  /dev/riorl003 2127M &
addfile.sh iorl_n1_1  /dev/riorl004 2127M &
addfile.sh iorl_n1_1  /dev/riorl005 2127M &
addfile.sh iorl_n1_1  /dev/riorl006 2127M &
addfile.sh iorl_n1_1  /dev/riorl007 2127M &
addfile.sh iorl_n1_1  /dev/riorl008 2127M &
addfile.sh iorl_n1_1  /dev/riorl009 2127M &
addfile.sh iorl_n1_1  /dev/riorl010 2127M &
addfile.sh iorl_n1_1  /dev/riorl011 2127M &

wait

addfile.sh iorl_n1_2  /dev/riorl013 2127M &
addfile.sh iorl_n1_2  /dev/riorl014 2127M &
addfile.sh iorl_n1_2  /dev/riorl015 2127M &
addfile.sh iorl_n1_2  /dev/riorl016 2127M &
addfile.sh iorl_n1_2  /dev/riorl017 2127M &
addfile.sh iorl_n1_2  /dev/riorl018 2127M &
addfile.sh iorl_n1_2  /dev/riorl019 2127M &
addfile.sh iorl_n1_2  /dev/riorl020 2127M &
addfile.sh iorl_n1_2  /dev/riorl021 2127M &
addfile.sh iorl_n1_2  /dev/riorl022 2127M &

wait
```

cr_n1_istk_ts.sh

```
#################################################################################
#
#   FILE: cr_n1_istk_ts.sh
#
#         Create istk tablespace on node1
#
#################################################################################

. ~/.profile

addfile.sh istk  /dev/ristk002 1126M &
addfile.sh istk  /dev/ristk003 1126M &
addfile.sh istk  /dev/ristk004 1126M &

wait
```

cr_n1_ordl_ts.sh

```
#################################################################################
#
#   FILE: cr_n1_ordl_ts.sh
#
#         Create ORDL tablespaces on node1
#
#################################################################################

. ~/.profile
set -x

svrmgrl <<!
        set echo on
        connect system/manager
        create tablespace ordl_n1 datafile '/dev/rordl001' size 1391M reuse;
        exit;
!

addfile.sh ordl_n1 /dev/rordl002 1391M &
addfile.sh ordl_n1 /dev/rordl003 1391M &
addfile.sh ordl_n1 /dev/rordl004 1391M &
addfile.sh ordl_n1 /dev/rordl005 1391M &
addfile.sh ordl_n1 /dev/rordl006 1391M &
addfile.sh ordl_n1 /dev/rordl007 1391M &
addfile.sh ordl_n1 /dev/rordl008 1391M &
addfile.sh ordl_n1 /dev/rordl009 1391M &
addfile.sh ordl_n1 /dev/rordl010 1391M &
addfile.sh ordl_n1 /dev/rordl011 1391M &
addfile.sh ordl_n1 /dev/rordl012 1391M &
addfile.sh ordl_n1 /dev/rordl013 1391M &
addfile.sh ordl_n1 /dev/rordl014 1391M &
addfile.sh ordl_n1 /dev/rordl015 1391M &
addfile.sh ordl_n1 /dev/rordl016 1391M &
addfile.sh ordl_n1 /dev/rordl017 1391M &
addfile.sh ordl_n1 /dev/rordl018 1391M &
addfile.sh ordl_n1 /dev/rordl019 1391M &
addfile.sh ordl_n1 /dev/rordl020 1391M &

#wait

addfile.sh ordl_n1 /dev/rordl021 1391M &
addfile.sh ordl_n1 /dev/rordl022 1391M &
addfile.sh ordl_n1 /dev/rordl023 1391M &
addfile.sh ordl_n1 /dev/rordl024 1391M &
addfile.sh ordl_n1 /dev/rordl025 1391M &
addfile.sh ordl_n1 /dev/rordl026 1391M &
addfile.sh ordl_n1 /dev/rordl027 1391M &
addfile.sh ordl_n1 /dev/rordl028 1391M &
addfile.sh ordl_n1 /dev/rordl029 1391M &
addfile.sh ordl_n1 /dev/rordl030 1391M &
addfile.sh ordl_n1 /dev/rordl031 1391M &
addfile.sh ordl_n1 /dev/rordl032 1391M &
addfile.sh ordl_n1 /dev/rordl033 1391M &
addfile.sh ordl_n1 /dev/rordl034 1391M &
addfile.sh ordl_n1 /dev/rordl035 1391M &
addfile.sh ordl_n1 /dev/rordl036 1391M &
addfile.sh ordl_n1 /dev/rordl037 1391M &
addfile.sh ordl_n1 /dev/rordl038 1391M &
addfile.sh ordl_n1 /dev/rordl039 1391M &
addfile.sh ordl_n1 /dev/rordl040 1391M &

wait

addfile.sh ordl_n1 /dev/rordl041 1391M &
addfile.sh ordl_n1 /dev/rordl042 1391M &
addfile.sh ordl_n1 /dev/rordl043 1391M &
addfile.sh ordl_n1 /dev/rordl044 1391M &
addfile.sh ordl_n1 /dev/rordl045 1391M &
addfile.sh ordl_n1 /dev/rordl046 1391M &
addfile.sh ordl_n1 /dev/rordl047 1391M &
addfile.sh ordl_n1 /dev/rordl048 1391M &
addfile.sh ordl_n1 /dev/rordl049 1391M &
addfile.sh ordl_n1 /dev/rordl050 1391M &
addfile.sh ordl_n1 /dev/rordl051 1391M &
addfile.sh ordl_n1 /dev/rordl052 1391M &
addfile.sh ordl_n1 /dev/rordl053 1391M &
addfile.sh ordl_n1 /dev/rordl054 1391M &
addfile.sh ordl_n1 /dev/rordl055 1391M &
addfile.sh ordl_n1 /dev/rordl056 1391M &
addfile.sh ordl_n1 /dev/rordl057 1391M &
addfile.sh ordl_n1 /dev/rordl058 1391M &
addfile.sh ordl_n1 /dev/rordl059 1391M &
addfile.sh ordl_n1 /dev/rordl060 1391M &

#wait

addfile.sh ordl_n1 /dev/rordl061 1391M &
addfile.sh ordl_n1 /dev/rordl062 1391M &
addfile.sh ordl_n1 /dev/rordl063 1391M &
addfile.sh ordl_n1 /dev/rordl064 1391M &
addfile.sh ordl_n1 /dev/rordl065 1391M &
addfile.sh ordl_n1 /dev/rordl066 1391M &
addfile.sh ordl_n1 /dev/rordl067 1391M &
addfile.sh ordl_n1 /dev/rordl068 1391M &
addfile.sh ordl_n1 /dev/rordl069 1391M &
addfile.sh ordl_n1 /dev/rordl070 1391M &
addfile.sh ordl_n1 /dev/rordl071 1391M &
addfile.sh ordl_n1 /dev/rordl072 1391M &
addfile.sh ordl_n1 /dev/rordl073 1391M &
addfile.sh ordl_n1 /dev/rordl074 1391M &
addfile.sh ordl_n1 /dev/rordl075 1391M &
addfile.sh ordl_n1 /dev/rordl076 1391M &
addfile.sh ordl_n1 /dev/rordl077 1391M &
addfile.sh ordl_n1 /dev/rordl078 1391M &
addfile.sh ordl_n1 /dev/rordl079 1391M &
addfile.sh ordl_n1 /dev/rordl080 1391M &

wait
echo "Done: cr_n1_ordl_ts.sh - `date`"
```

cr_n1_ords_ts.sh

```
#################################################################################
#
#   FILE: cr_n1_ords_ts.sh
#
```

```
#       Create ords tablespace on node1
#
################################################################################

. ~/.profile
set -x

svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ords_n1 datafile '/dev/rords001' size 607M reuse;
      exit;
!
addfile.sh ords_n1  /dev/rords002 607M &
addfile.sh ords_n1  /dev/rords003 607M &
addfile.sh ords_n1  /dev/rords004 607M &
addfile.sh ords_n1  /dev/rords005 607M &
addfile.sh ords_n1  /dev/rords006 607M &
addfile.sh ords_n1  /dev/rords007 607M &
addfile.sh ords_n1  /dev/rords008 607M &
addfile.sh ords_n1  /dev/rords009 607M &
addfile.sh ords_n1  /dev/rords010 607M &

wait
date
echo "Done: cr_n1_ords_ts.sh"
```

cr_n1_stoc_ts.sh

```
################################################################################
#
#   FILE:  cr_n1_stoc_ts.sh
#
#       Create stoc tablespace on node1
#
################################################################################

. ~/.profile

addfile.sh stoc /dev/rstoc002 371M &
addfile.sh stoc /dev/rstoc003 371M &
addfile.sh stoc /dev/rstoc004 371M &
addfile.sh stoc /dev/rstoc005 371M &
addfile.sh stoc /dev/rstoc006 371M &
addfile.sh stoc /dev/rstoc007 371M &
addfile.sh stoc /dev/rstoc008 371M &
addfile.sh stoc /dev/rstoc009 371M &
addfile.sh stoc /dev/rstoc010 371M &
addfile.sh stoc /dev/rstoc011 371M &
addfile.sh stoc /dev/rstoc012 371M &
addfile.sh stoc /dev/rstoc013 371M &
addfile.sh stoc /dev/rstoc014 371M &
addfile.sh stoc /dev/rstoc015 371M &
addfile.sh stoc /dev/rstoc016 371M &
addfile.sh stoc /dev/rstoc017 371M &
addfile.sh stoc /dev/rstoc018 371M &
addfile.sh stoc /dev/rstoc019 371M &
addfile.sh stoc /dev/rstoc020 371M &
addfile.sh stoc /dev/rstoc021 371M &
addfile.sh stoc /dev/rstoc022 371M &
addfile.sh stoc /dev/rstoc023 371M &
addfile.sh stoc /dev/rstoc024 371M &
addfile.sh stoc /dev/rstoc025 371M &
addfile.sh stoc /dev/rstoc026 371M &
addfile.sh stoc /dev/rstoc027 371M &
addfile.sh stoc /dev/rstoc028 371M &
addfile.sh stoc /dev/rstoc029 371M &
addfile.sh stoc /dev/rstoc030 371M &
addfile.sh stoc /dev/rstoc031 371M &
addfile.sh stoc /dev/rstoc032 371M &
addfile.sh stoc /dev/rstoc033 371M &
addfile.sh stoc /dev/rstoc034 371M &
addfile.sh stoc /dev/rstoc035 371M &
addfile.sh stoc /dev/rstoc036 371M &
addfile.sh stoc /dev/rstoc037 371M &
addfile.sh stoc /dev/rstoc038 371M &
addfile.sh stoc /dev/rstoc039 371M &
addfile.sh stoc /dev/rstoc040 371M &
addfile.sh stoc /dev/rstoc041 371M &
addfile.sh stoc /dev/rstoc042 371M &
addfile.sh stoc /dev/rstoc043 371M &
addfile.sh stoc /dev/rstoc044 371M &
addfile.sh stoc /dev/rstoc045 371M &
addfile.sh stoc /dev/rstoc046 371M &
addfile.sh stoc /dev/rstoc047 371M &
addfile.sh stoc /dev/rstoc048 371M &
addfile.sh stoc /dev/rstoc049 371M &
addfile.sh stoc /dev/rstoc050 371M &
addfile.sh stoc /dev/rstoc051 371M &
addfile.sh stoc /dev/rstoc052 371M &
addfile.sh stoc /dev/rstoc053 371M &
addfile.sh stoc /dev/rstoc054 371M &
addfile.sh stoc /dev/rstoc055 371M &
addfile.sh stoc /dev/rstoc056 371M &
addfile.sh stoc /dev/rstoc057 371M &
addfile.sh stoc /dev/rstoc058 371M &
addfile.sh stoc /dev/rstoc059 371M &
addfile.sh stoc /dev/rstoc060 371M &
addfile.sh stoc /dev/rstoc061 371M &
addfile.sh stoc /dev/rstoc062 371M &
addfile.sh stoc /dev/rstoc063 371M &
addfile.sh stoc /dev/rstoc064 371M &
addfile.sh stoc /dev/rstoc065 371M &
addfile.sh stoc /dev/rstoc066 371M &

addfile.sh stoc /dev/rstoc067 371M &
addfile.sh stoc /dev/rstoc068 371M &
addfile.sh stoc /dev/rstoc069 371M &
addfile.sh stoc /dev/rstoc070 371M &
addfile.sh stoc /dev/rstoc071a 371M &
addfile.sh stoc /dev/rstoc072 371M &
addfile.sh stoc /dev/rstoc073 371M &
addfile.sh stoc /dev/rstoc074 371M &
addfile.sh stoc /dev/rstoc075 371M &
addfile.sh stoc /dev/rstoc076 371M &
addfile.sh stoc /dev/rstoc077 371M &
addfile.sh stoc /dev/rstoc078 371M &
addfile.sh stoc /dev/rstoc079 371M &
addfile.sh stoc /dev/rstoc080 371M &
addfile.sh stoc /dev/rstoc081 371M &
addfile.sh stoc /dev/rstoc082 371M &
addfile.sh stoc /dev/rstoc083 371M &
addfile.sh stoc /dev/rstoc084 371M &
addfile.sh stoc /dev/rstoc085 371M &
addfile.sh stoc /dev/rstoc086 371M &
addfile.sh stoc /dev/rstoc087 371M &
addfile.sh stoc /dev/rstoc088 371M &

addfile.sh stoc /dev/rstoc089 371M &
addfile.sh stoc /dev/rstoc090 371M &
addfile.sh stoc /dev/rstoc091 371M &
addfile.sh stoc /dev/rstoc092 371M &
addfile.sh stoc /dev/rstoc093 371M &
addfile.sh stoc /dev/rstoc094 371M &
addfile.sh stoc /dev/rstoc095 371M &
addfile.sh stoc /dev/rstoc096 371M &
addfile.sh stoc /dev/rstoc097 371M &
addfile.sh stoc /dev/rstoc098 371M &
addfile.sh stoc /dev/rstoc099 371M &
addfile.sh stoc /dev/rstoc100 371M &
addfile.sh stoc /dev/rstoc101 371M &
addfile.sh stoc /dev/rstoc102 371M &
addfile.sh stoc /dev/rstoc103 371M &
addfile.sh stoc /dev/rstoc104 371M &
addfile.sh stoc /dev/rstoc105 371M &
addfile.sh stoc /dev/rstoc106 371M &
addfile.sh stoc /dev/rstoc107 371M &
addfile.sh stoc /dev/rstoc108 371M &
addfile.sh stoc /dev/rstoc109 371M &
addfile.sh stoc /dev/rstoc110 371M &

addfile.sh stoc /dev/rstoc111 371M &
addfile.sh stoc /dev/rstoc112 371M &
addfile.sh stoc /dev/rstoc113 371M &
addfile.sh stoc /dev/rstoc114 371M &
addfile.sh stoc /dev/rstoc115 371M &
addfile.sh stoc /dev/rstoc116 371M &
addfile.sh stoc /dev/rstoc117 371M &
addfile.sh stoc /dev/rstoc118 371M &
addfile.sh stoc /dev/rstoc119 371M &
addfile.sh stoc /dev/rstoc120 371M &
addfile.sh stoc /dev/rstoc121 371M &
addfile.sh stoc /dev/rstoc122 371M &
addfile.sh stoc /dev/rstoc123 371M &
addfile.sh stoc /dev/rstoc124 371M &
addfile.sh stoc /dev/rstoc125 371M &
addfile.sh stoc /dev/rstoc126 371M &
addfile.sh stoc /dev/rstoc127 371M &
addfile.sh stoc /dev/rstoc128 371M &
addfile.sh stoc /dev/rstoc129 371M &
addfile.sh stoc /dev/rstoc130 371M &
addfile.sh stoc /dev/rstoc131 371M &
addfile.sh stoc /dev/rstoc132 371M &

addfile.sh stoc /dev/rstoc133 371M &
addfile.sh stoc /dev/rstoc134 371M &
addfile.sh stoc /dev/rstoc135 371M &
addfile.sh stoc /dev/rstoc136 371M &
addfile.sh stoc /dev/rstoc137 371M &
addfile.sh stoc /dev/rstoc138 371M &
addfile.sh stoc /dev/rstoc139 371M &
addfile.sh stoc /dev/rstoc140 371M &
addfile.sh stoc /dev/rstoc141 371M &
addfile.sh stoc /dev/rstoc142 371M &
addfile.sh stoc /dev/rstoc143 371M &
addfile.sh stoc /dev/rstoc144 371M &
addfile.sh stoc /dev/rstoc145 371M &
addfile.sh stoc /dev/rstoc146 371M &
addfile.sh stoc /dev/rstoc147 371M &
addfile.sh stoc /dev/rstoc148 371M &
addfile.sh stoc /dev/rstoc149 371M &
addfile.sh stoc /dev/rstoc150 371M &
addfile.sh stoc /dev/rstoc151 371M &
addfile.sh stoc /dev/rstoc152 371M &
addfile.sh stoc /dev/rstoc153 371M &
addfile.sh stoc /dev/rstoc154 371M &

addfile.sh stoc /dev/rstoc155 371M &
```

```
addfile.sh stoc /dev/rstoc156 371M &
addfile.sh stoc /dev/rstoc157 371M &
addfile.sh stoc /dev/rstoc158 371M &
addfile.sh stoc /dev/rstoc159 371M &
addfile.sh stoc /dev/rstoc160 371M &
addfile.sh stoc /dev/rstoc161 371M &
addfile.sh stoc /dev/rstoc162 371M &
addfile.sh stoc /dev/rstoc163 371M &
addfile.sh stoc /dev/rstoc164 371M &
addfile.sh stoc /dev/rstoc165 371M &
addfile.sh stoc /dev/rstoc166 371M &
addfile.sh stoc /dev/rstoc167 371M &
addfile.sh stoc /dev/rstoc168 371M &
addfile.sh stoc /dev/rstoc169 371M &
addfile.sh stoc /dev/rstoc170 371M &
addfile.sh stoc /dev/rstoc171 371M &
addfile.sh stoc /dev/rstoc172 371M &
addfile.sh stoc /dev/rstoc173 371M &
addfile.sh stoc /dev/rstoc174 371M &
addfile.sh stoc /dev/rstoc175 371M &
addfile.sh stoc /dev/rstoc176 371M &

addfile.sh stoc /dev/rstoc177 371M &
addfile.sh stoc /dev/rstoc178 371M &
addfile.sh stoc /dev/rstoc179 371M &
addfile.sh stoc /dev/rstoc180 371M &
addfile.sh stoc /dev/rstoc181 371M &
addfile.sh stoc /dev/rstoc182 371M &
addfile.sh stoc /dev/rstoc183 371M &
addfile.sh stoc /dev/rstoc184 371M &
addfile.sh stoc /dev/rstoc185 371M &
addfile.sh stoc /dev/rstoc186 371M &
addfile.sh stoc /dev/rstoc187 371M &
addfile.sh stoc /dev/rstoc188 371M &
addfile.sh stoc /dev/rstoc189 371M &
addfile.sh stoc /dev/rstoc190 371M &
addfile.sh stoc /dev/rstoc191 371M &
addfile.sh stoc /dev/rstoc192 371M &
addfile.sh stoc /dev/rstoc193 371M &
addfile.sh stoc /dev/rstoc194 371M &
addfile.sh stoc /dev/rstoc195 371M &
addfile.sh stoc /dev/rstoc196 371M &
addfile.sh stoc /dev/rstoc197 371M &
addfile.sh stoc /dev/rstoc198 371M &

addfile.sh stoc /dev/rstoc199 371M &
addfile.sh stoc /dev/rstoc200 371M &
addfile.sh stoc /dev/rstoc201 371M &
addfile.sh stoc /dev/rstoc202 371M &
addfile.sh stoc /dev/rstoc203 371M &
addfile.sh stoc /dev/rstoc204 371M &
addfile.sh stoc /dev/rstoc205 371M &
addfile.sh stoc /dev/rstoc206 371M &
addfile.sh stoc /dev/rstoc207 371M &
addfile.sh stoc /dev/rstoc208 371M &
addfile.sh stoc /dev/rstoc209 371M &
addfile.sh stoc /dev/rstoc210 371M &
addfile.sh stoc /dev/rstoc211 371M &
addfile.sh stoc /dev/rstoc212 371M &
addfile.sh stoc /dev/rstoc213 371M &
addfile.sh stoc /dev/rstoc214 371M &
addfile.sh stoc /dev/rstoc215 371M &
addfile.sh stoc /dev/rstoc216 371M &
addfile.sh stoc /dev/rstoc217 371M &
addfile.sh stoc /dev/rstoc218 371M &
addfile.sh stoc /dev/rstoc219 371M &
addfile.sh stoc /dev/rstoc220 371M &
addfile.sh stoc /dev/rstoc221 371M &
addfile.sh stoc /dev/rstoc222 371M &

wait
```

cr_n1_temp_ts.sh

```
#################################################################################
#
#  FILE: cr_n1_temp_ts.sh
#
#       Create temp tablespace on node1
#
#################################################################################

. ~/.profile

addfile.sh temp  /dev/rtemp002 1999M &
addfile.sh temp  /dev/rtemp003 1999M &
addfile.sh temp  /dev/rtemp004 1999M &
addfile.sh temp  /dev/rtemp005 1999M &
addfile.sh temp  /dev/rtemp006 1999M &
addfile.sh temp  /dev/rtemp007 1999M &
addfile.sh temp  /dev/rtemp008 1999M &
addfile.sh temp  /dev/rtemp009 1999M &
addfile.sh temp  /dev/rtemp010 1999M &
addfile.sh temp  /dev/rtemp011 1999M &
addfile.sh temp  /dev/rtemp012 1999M &
addfile.sh temp  /dev/rtemp013 1999M &
addfile.sh temp  /dev/rtemp014 1999M &
addfile.sh temp  /dev/rtemp015 1999M &
addfile.sh temp  /dev/rtemp016 1999M &
addfile.sh temp  /dev/rtemp017 1999M &
addfile.sh temp  /dev/rtemp018 1999M &
addfile.sh temp  /dev/rtemp019 1999M &
addfile.sh temp  /dev/rtemp020 1999M &
addfile.sh temp  /dev/rtemp021 1999M &

wait
#################################################################################
#
#  FILE:  cr_n2_cust_ts.sh
#
#       Createcust tablespace on node2
#
#################################################################################

. ~/.profile


addfile.sh cust  /dev/rcust089  771M &
addfile.sh cust  /dev/rcust090  771M &
addfile.sh cust  /dev/rcust091  771M &
addfile.sh cust  /dev/rcust092  771M &
addfile.sh cust  /dev/rcust093  771M &
addfile.sh cust  /dev/rcust094  771M &
addfile.sh cust  /dev/rcust095  771M &
addfile.sh cust  /dev/rcust096  771M &
addfile.sh cust  /dev/rcust097  771M &
addfile.sh cust  /dev/rcust098  771M &
addfile.sh cust  /dev/rcust099  771M &
addfile.sh cust  /dev/rcust100  771M &
addfile.sh cust  /dev/rcust101  771M &
addfile.sh cust  /dev/rcust102  771M &
addfile.sh cust  /dev/rcust103  771M &
addfile.sh cust  /dev/rcust104  771M &
addfile.sh cust  /dev/rcust105  771M &
addfile.sh cust  /dev/rcust106  771M &
addfile.sh cust  /dev/rcust107  771M &
addfile.sh cust  /dev/rcust108  771M &
addfile.sh cust  /dev/rcust109  771M &
addfile.sh cust  /dev/rcust110  771M &

wait

addfile.sh cust  /dev/rcust111  771M &
addfile.sh cust  /dev/rcust112  771M &
addfile.sh cust  /dev/rcust113  771M &
addfile.sh cust  /dev/rcust114  771M &
addfile.sh cust  /dev/rcust115  771M &
addfile.sh cust  /dev/rcust116  771M &
addfile.sh cust  /dev/rcust117  771M &
addfile.sh cust  /dev/rcust118  771M &
addfile.sh cust  /dev/rcust119  771M &
addfile.sh cust  /dev/rcust120  771M &
addfile.sh cust  /dev/rcust121  771M &
addfile.sh cust  /dev/rcust122  771M &
addfile.sh cust  /dev/rcust123  771M &
addfile.sh cust  /dev/rcust124  771M &
addfile.sh cust  /dev/rcust125  771M &
addfile.sh cust  /dev/rcust126  771M &
addfile.sh cust  /dev/rcust127  771M &
addfile.sh cust  /dev/rcust128  771M &
addfile.sh cust  /dev/rcust129  771M &
addfile.sh cust  /dev/rcust130  771M &
addfile.sh cust  /dev/rcust131  771M &
addfile.sh cust  /dev/rcust132  771M &

wait

addfile.sh cust  /dev/rcust133  771M &
addfile.sh cust  /dev/rcust134  771M &
addfile.sh cust  /dev/rcust135  771M &
addfile.sh cust  /dev/rcust136  771M &
addfile.sh cust  /dev/rcust137  771M &
addfile.sh cust  /dev/rcust138  771M &
addfile.sh cust  /dev/rcust139  771M &
addfile.sh cust  /dev/rcust140  771M &
addfile.sh cust  /dev/rcust141  771M &
addfile.sh cust  /dev/rcust142  771M &
addfile.sh cust  /dev/rcust143  771M &
addfile.sh cust  /dev/rcust144  771M &
addfile.sh cust  /dev/rcust145  771M &
addfile.sh cust  /dev/rcust146  771M &
addfile.sh cust  /dev/rcust147  771M &
addfile.sh cust  /dev/rcust148  771M &
addfile.sh cust  /dev/rcust149  771M &
addfile.sh cust  /dev/rcust150  771M &
addfile.sh cust  /dev/rcust151  771M &
addfile.sh cust  /dev/rcust152  771M &
addfile.sh cust  /dev/rcust153  771M &
addfile.sh cust  /dev/rcust154  771M &

wait

addfile.sh cust  /dev/rcust155  771M &
addfile.sh cust  /dev/rcust156  771M &
addfile.sh cust  /dev/rcust157  771M &
addfile.sh cust  /dev/rcust158  771M &
addfile.sh cust  /dev/rcust159  771M &
addfile.sh cust  /dev/rcust160  771M &
```

```
addfile.sh cust  /dev/rcust161 771M &
addfile.sh cust  /dev/rcust162 771M &
addfile.sh cust  /dev/rcust163 771M &
addfile.sh cust  /dev/rcust164 771M &
addfile.sh cust  /dev/rcust165 771M &
addfile.sh cust  /dev/rcust166 771M &
addfile.sh cust  /dev/rcust167 771M &
addfile.sh cust  /dev/rcust168 771M &
addfile.sh cust  /dev/rcust169 771M &
addfile.sh cust  /dev/rcust170 771M &
addfile.sh cust  /dev/rcust171 771M &
addfile.sh cust  /dev/rcust172 771M &
addfile.sh cust  /dev/rcust173 771M &
addfile.sh cust  /dev/rcust174 771M &
addfile.sh cust  /dev/rcust175 771M &
addfile.sh cust  /dev/rcust176 771M &

wait

#############################################################################
#
#  FILE:  cr_n2_iord_ts.sh
#
#       Create ior1 and ior2 tablespaces on node2
#
#############################################################################

. ~/.profile
set -x
svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ior1_n2 datafile '/dev/rior1011' size 607M reuse;
      exit;
!

addfile.sh ior1_n2  /dev/rior1012 607M &
addfile.sh ior1_n2  /dev/rior1013 607M &
addfile.sh ior1_n2  /dev/rior1014 607M &
addfile.sh ior1_n2  /dev/rior1015 607M &
addfile.sh ior1_n2  /dev/rior1016 607M &
addfile.sh ior1_n2  /dev/rior1017 607M &
addfile.sh ior1_n2  /dev/rior1018 607M &
addfile.sh ior1_n2  /dev/rior1019 607M &
addfile.sh ior1_n2  /dev/rior1020 607M &

wait
date
echo "Done: cr_n2_iord1_ts.sh"


#############################################################################
#
#  FILE:  cr_n2_iord_ts.sh
#
#       Create ior1 and ior2 tablespaces on node2
#
#############################################################################

. ~/.profile
set -x

svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace ior2_n2 datafile   '/dev/rior2021'   size 511M reuse;
  exit;
!
addfile.sh ior2_n2  /dev/rior2022 511M &
addfile.sh ior2_n2  /dev/rior2023 511M &
addfile.sh ior2_n2  /dev/rior2024 511M &
addfile.sh ior2_n2  /dev/rior2025 511M &
addfile.sh ior2_n2  /dev/rior2026 511M &
addfile.sh ior2_n2  /dev/rior2027 511M &
addfile.sh ior2_n2  /dev/rior2028 511M &
addfile.sh ior2_n2  /dev/rior2029 511M &
addfile.sh ior2_n2  /dev/rior2030 511M &
addfile.sh ior2_n2  /dev/rior2031 511M &
addfile.sh ior2_n2  /dev/rior2032 511M &
addfile.sh ior2_n2  /dev/rior2033 511M &
addfile.sh ior2_n2  /dev/rior2034 511M &
addfile.sh ior2_n2  /dev/rior2035 511M &
addfile.sh ior2_n2  /dev/rior2036 511M &
addfile.sh ior2_n2  /dev/rior2037 511M &
addfile.sh ior2_n2  /dev/rior2038 511M &
addfile.sh ior2_n2  /dev/rior2039 511M &
addfile.sh ior2_n2  /dev/rior2040 511M &
wait
date
echo "Done: cr_n2_iord2_ts.sh"


#############################################################################
#
#  FILE:  cr_n2_iorl_ts.sh
#
#       Create iorl tablespace  on node2
#
#############################################################################

. ~/.profile

svrmgrl <<!
      set echo on
      connect system/manager

      create tablespace iorl_n2_1 datafile   '/dev/riorl023'   size 2127M reuse;
      create tablespace iorl_n2_2 datafile   '/dev/riorl034'   size 2127M reuse;

      exit;
!

addfile.sh iorl_n2_1  /dev/riorl024 2127M &
addfile.sh iorl_n2_1  /dev/riorl025 2127M &
addfile.sh iorl_n2_1  /dev/riorl026 2127M &
addfile.sh iorl_n2_1  /dev/riorl027 2127M &
addfile.sh iorl_n2_1  /dev/riorl028 2127M &
addfile.sh iorl_n2_1  /dev/riorl029 2127M &
addfile.sh iorl_n2_1  /dev/riorl030 2127M &
addfile.sh iorl_n2_1  /dev/riorl031 2127M &
addfile.sh iorl_n2_1  /dev/riorl032 2127M &
addfile.sh iorl_n2_1  /dev/riorl033 2127M &

wait

addfile.sh iorl_n2_2  /dev/riorl035 2127M &
addfile.sh iorl_n2_2  /dev/riorl036 2127M &
addfile.sh iorl_n2_2  /dev/riorl037 2127M &
addfile.sh iorl_n2_2  /dev/riorl038 2127M &
addfile.sh iorl_n2_2  /dev/riorl039 2127M &
addfile.sh iorl_n2_2  /dev/riorl040 2127M &
addfile.sh iorl_n2_2  /dev/riorl041 2127M &
addfile.sh iorl_n2_2  /dev/riorl042 2127M &
addfile.sh iorl_n2_2  /dev/riorl043 2127M &
addfile.sh iorl_n2_2  /dev/riorl044 2127M &

wait


#############################################################################
#
#  FILE:  cr_n2_istk_ts.sh
#
#       Create istk tablespace on node2
#
#############################################################################

. ~/.profile

addfile.sh istk  /dev/ristk005 1126M &
addfile.sh istk  /dev/ristk006 1126M &
addfile.sh istk  /dev/ristk007 1126M &
addfile.sh istk  /dev/ristk008 1126M &

wait

#############################################################################
#
#  FILE:  cr_n2_ordl_ts.sh
#
#       Create ORDL tablespaces on node2
#
#############################################################################

. ~/.profile
set -x

echo "starting Node 2 - `date`"
svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ordl_n2 datafile '/dev/rordl081' size 1391M reuse;
      exit;
!

addfile.sh ordl_n2 /dev/rordl082 1391M &
addfile.sh ordl_n2 /dev/rordl083 1391M &
addfile.sh ordl_n2 /dev/rordl084 1391M &
addfile.sh ordl_n2 /dev/rordl085 1391M &
addfile.sh ordl_n2 /dev/rordl086 1391M &
addfile.sh ordl_n2 /dev/rordl087 1391M &
addfile.sh ordl_n2 /dev/rordl088 1391M &
addfile.sh ordl_n2 /dev/rordl089 1391M &
addfile.sh ordl_n2 /dev/rordl090 1391M &
addfile.sh ordl_n2 /dev/rordl091 1391M &
addfile.sh ordl_n2 /dev/rordl092 1391M &
addfile.sh ordl_n2 /dev/rordl093 1391M &
addfile.sh ordl_n2 /dev/rordl094 1391M &
addfile.sh ordl_n2 /dev/rordl095 1391M &
addfile.sh ordl_n2 /dev/rordl096 1391M &
addfile.sh ordl_n2 /dev/rordl097 1391M &
addfile.sh ordl_n2 /dev/rordl098 1391M &
addfile.sh ordl_n2 /dev/rordl099 1391M &
addfile.sh ordl_n2 /dev/rordl100 1391M &

#wait

addfile.sh ordl_n2 /dev/rordl101 1391M &
```

```
addfile.sh ordl_n2 /dev/rordl102 1391M &
addfile.sh ordl_n2 /dev/rordl103 1391M &
addfile.sh ordl_n2 /dev/rordl104 1391M &
addfile.sh ordl_n2 /dev/rordl105 1391M &
addfile.sh ordl_n2 /dev/rordl106 1391M &
addfile.sh ordl_n2 /dev/rordl107 1391M &
addfile.sh ordl_n2 /dev/rordl108 1391M &
addfile.sh ordl_n2 /dev/rordl109 1391M &
addfile.sh ordl_n2 /dev/rordl110 1391M &
addfile.sh ordl_n2 /dev/rordl111 1391M &
addfile.sh ordl_n2 /dev/rordl112 1391M &
addfile.sh ordl_n2 /dev/rordl113 1391M &
addfile.sh ordl_n2 /dev/rordl114 1391M &
addfile.sh ordl_n2 /dev/rordl115 1391M &
addfile.sh ordl_n2 /dev/rordl116 1391M &
addfile.sh ordl_n2 /dev/rordl117 1391M &
addfile.sh ordl_n2 /dev/rordl118 1391M &
addfile.sh ordl_n2 /dev/rordl119 1391M &
addfile.sh ordl_n2 /dev/rordl120 1391M &

wait

addfile.sh ordl_n2 /dev/rordl121 1391M &
addfile.sh ordl_n2 /dev/rordl122 1391M &
addfile.sh ordl_n2 /dev/rordl123 1391M &
addfile.sh ordl_n2 /dev/rordl124 1391M &
addfile.sh ordl_n2 /dev/rordl125 1391M &
addfile.sh ordl_n2 /dev/rordl126 1391M &
addfile.sh ordl_n2 /dev/rordl127 1391M &
addfile.sh ordl_n2 /dev/rordl128 1391M &
addfile.sh ordl_n2 /dev/rordl129 1391M &
addfile.sh ordl_n2 /dev/rordl130 1391M &
addfile.sh ordl_n2 /dev/rordl131 1391M &
addfile.sh ordl_n2 /dev/rordl132 1391M &
addfile.sh ordl_n2 /dev/rordl133 1391M &
addfile.sh ordl_n2 /dev/rordl134 1391M &
addfile.sh ordl_n2 /dev/rordl135 1391M &
addfile.sh ordl_n2 /dev/rordl136 1391M &
addfile.sh ordl_n2 /dev/rordl137 1391M &
addfile.sh ordl_n2 /dev/rordl138 1391M &
addfile.sh ordl_n2 /dev/rordl139 1391M &
addfile.sh ordl_n2 /dev/rordl140 1391M &

#wait

addfile.sh ordl_n2 /dev/rordl141 1391M &
addfile.sh ordl_n2 /dev/rordl142 1391M &
addfile.sh ordl_n2 /dev/rordl143 1391M &
addfile.sh ordl_n2 /dev/rordl144 1391M &
addfile.sh ordl_n2 /dev/rordl145 1391M &
addfile.sh ordl_n2 /dev/rordl146 1391M &
addfile.sh ordl_n2 /dev/rordl147 1391M &
addfile.sh ordl_n2 /dev/rordl148 1391M &
addfile.sh ordl_n2 /dev/rordl149 1391M &
addfile.sh ordl_n2 /dev/rordl150 1391M &
addfile.sh ordl_n2 /dev/rordl151 1391M &
addfile.sh ordl_n2 /dev/rordl152 1391M &
addfile.sh ordl_n2 /dev/rordl153 1391M &
addfile.sh ordl_n2 /dev/rordl154 1391M &
addfile.sh ordl_n2 /dev/rordl155 1391M &
addfile.sh ordl_n2 /dev/rordl156 1391M &
addfile.sh ordl_n2 /dev/rordl157 1391M &
addfile.sh ordl_n2 /dev/rordl158 1391M &
addfile.sh ordl_n2 /dev/rordl159 1391M &
addfile.sh ordl_n2 /dev/rordl160 1391M &

wait

echo "Done: cr_n2_ordl_ts.sh - `date`"

################################################################################
#
#   FILE:  cr_n2_ords_ts.sh
#
#         Createords tablespace on node2
#
################################################################################

. ~/.profile
set -x

svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ords_n2 datafile '/dev/rords011' size 607M reuse;
      exit;
!
addfile.sh ords_n2  /dev/rords012 607M &
addfile.sh ords_n2  /dev/rords013 607M &
addfile.sh ords_n2  /dev/rords014 607M &
addfile.sh ords_n2  /dev/rords015 607M &
addfile.sh ords_n2  /dev/rords016 607M &
addfile.sh ords_n2  /dev/rords017 607M &
addfile.sh ords_n2  /dev/rords018 607M &
addfile.sh ords_n2  /dev/rords019 607M &
addfile.sh ords_n2  /dev/rords020 607M &

wait
date
echo "Done cr_n2_ords_ts"

################################################################################
#
#   FILE:  cr_n1_stoc_ts.sh
#
#         Createstoc tablespace on node1
#
################################################################################

. ~/.profile

addfile.sh stoc /dev/rstoc002 371M &
addfile.sh stoc /dev/rstoc003 371M &
addfile.sh stoc /dev/rstoc004 371M &
addfile.sh stoc /dev/rstoc005 371M &
addfile.sh stoc /dev/rstoc006 371M &
addfile.sh stoc /dev/rstoc007 371M &
addfile.sh stoc /dev/rstoc008 371M &
addfile.sh stoc /dev/rstoc009 371M &
addfile.sh stoc /dev/rstoc010 371M &
addfile.sh stoc /dev/rstoc011 371M &
addfile.sh stoc /dev/rstoc012 371M &
addfile.sh stoc /dev/rstoc013 371M &
addfile.sh stoc /dev/rstoc014 371M &
addfile.sh stoc /dev/rstoc015 371M &
addfile.sh stoc /dev/rstoc016 371M &
addfile.sh stoc /dev/rstoc017 371M &
addfile.sh stoc /dev/rstoc018 371M &
addfile.sh stoc /dev/rstoc019 371M &
addfile.sh stoc /dev/rstoc020 371M &
addfile.sh stoc /dev/rstoc021 371M &
addfile.sh stoc /dev/rstoc022 371M &
addfile.sh stoc /dev/rstoc023 371M &
addfile.sh stoc /dev/rstoc024 371M &
addfile.sh stoc /dev/rstoc025 371M &
addfile.sh stoc /dev/rstoc026 371M &
addfile.sh stoc /dev/rstoc027 371M &
addfile.sh stoc /dev/rstoc028 371M &
addfile.sh stoc /dev/rstoc029 371M &
addfile.sh stoc /dev/rstoc030 371M &
addfile.sh stoc /dev/rstoc031 371M &
addfile.sh stoc /dev/rstoc032 371M &
addfile.sh stoc /dev/rstoc033 371M &
addfile.sh stoc /dev/rstoc034 371M &
addfile.sh stoc /dev/rstoc035 371M &
addfile.sh stoc /dev/rstoc036 371M &
addfile.sh stoc /dev/rstoc037 371M &
addfile.sh stoc /dev/rstoc038 371M &
addfile.sh stoc /dev/rstoc039 371M &
addfile.sh stoc /dev/rstoc040 371M &
addfile.sh stoc /dev/rstoc041 371M &
addfile.sh stoc /dev/rstoc042 371M &
addfile.sh stoc /dev/rstoc043 371M &
addfile.sh stoc /dev/rstoc044 371M &
addfile.sh stoc /dev/rstoc045 371M &
addfile.sh stoc /dev/rstoc046 371M &
addfile.sh stoc /dev/rstoc047 371M &
addfile.sh stoc /dev/rstoc048 371M &
addfile.sh stoc /dev/rstoc049 371M &
addfile.sh stoc /dev/rstoc050 371M &
addfile.sh stoc /dev/rstoc051 371M &
addfile.sh stoc /dev/rstoc052 371M &
addfile.sh stoc /dev/rstoc053 371M &
addfile.sh stoc /dev/rstoc054 371M &
addfile.sh stoc /dev/rstoc055 371M &
addfile.sh stoc /dev/rstoc056 371M &
addfile.sh stoc /dev/rstoc057 371M &
addfile.sh stoc /dev/rstoc058 371M &
addfile.sh stoc /dev/rstoc059 371M &
addfile.sh stoc /dev/rstoc060 371M &
addfile.sh stoc /dev/rstoc061 371M &
addfile.sh stoc /dev/rstoc062 371M &
addfile.sh stoc /dev/rstoc063 371M &
addfile.sh stoc /dev/rstoc064 371M &
addfile.sh stoc /dev/rstoc065 371M &
addfile.sh stoc /dev/rstoc066 371M &

addfile.sh stoc /dev/rstoc067 371M &
addfile.sh stoc /dev/rstoc068 371M &
addfile.sh stoc /dev/rstoc069 371M &
addfile.sh stoc /dev/rstoc070 371M &
addfile.sh stoc /dev/rstoc071a 371M &
addfile.sh stoc /dev/rstoc072 371M &
addfile.sh stoc /dev/rstoc073 371M &
addfile.sh stoc /dev/rstoc074 371M &
addfile.sh stoc /dev/rstoc075 371M &
addfile.sh stoc /dev/rstoc076 371M &
addfile.sh stoc /dev/rstoc077 371M &
addfile.sh stoc /dev/rstoc078 371M &
addfile.sh stoc /dev/rstoc079 371M &
addfile.sh stoc /dev/rstoc080 371M &
addfile.sh stoc /dev/rstoc081 371M &
addfile.sh stoc /dev/rstoc082 371M &
addfile.sh stoc /dev/rstoc083 371M &
addfile.sh stoc /dev/rstoc084 371M &
addfile.sh stoc /dev/rstoc085 371M &
addfile.sh stoc /dev/rstoc086 371M &
addfile.sh stoc /dev/rstoc087 371M &
```

```
addfile.sh stoc /dev/rstoc088 371M &

addfile.sh stoc /dev/rstoc089 371M &
addfile.sh stoc /dev/rstoc090 371M &
addfile.sh stoc /dev/rstoc091 371M &
addfile.sh stoc /dev/rstoc092 371M &
addfile.sh stoc /dev/rstoc093 371M &
addfile.sh stoc /dev/rstoc094 371M &
addfile.sh stoc /dev/rstoc095 371M &
addfile.sh stoc /dev/rstoc096 371M &
addfile.sh stoc /dev/rstoc097 371M &
addfile.sh stoc /dev/rstoc098 371M &
addfile.sh stoc /dev/rstoc099 371M &
addfile.sh stoc /dev/rstoc100 371M &
addfile.sh stoc /dev/rstoc101 371M &
addfile.sh stoc /dev/rstoc102 371M &
addfile.sh stoc /dev/rstoc103 371M &
addfile.sh stoc /dev/rstoc104 371M &
addfile.sh stoc /dev/rstoc105 371M &
addfile.sh stoc /dev/rstoc106 371M &
addfile.sh stoc /dev/rstoc107 371M &
addfile.sh stoc /dev/rstoc108 371M &
addfile.sh stoc /dev/rstoc109 371M &
addfile.sh stoc /dev/rstoc110 371M &

addfile.sh stoc /dev/rstoc111 371M &
addfile.sh stoc /dev/rstoc112 371M &
addfile.sh stoc /dev/rstoc113 371M &
addfile.sh stoc /dev/rstoc114 371M &
addfile.sh stoc /dev/rstoc115 371M &
addfile.sh stoc /dev/rstoc116 371M &
addfile.sh stoc /dev/rstoc117 371M &
addfile.sh stoc /dev/rstoc118 371M &
addfile.sh stoc /dev/rstoc119 371M &
addfile.sh stoc /dev/rstoc120 371M &
addfile.sh stoc /dev/rstoc121 371M &
addfile.sh stoc /dev/rstoc122 371M &
addfile.sh stoc /dev/rstoc123 371M &
addfile.sh stoc /dev/rstoc124 371M &
addfile.sh stoc /dev/rstoc125 371M &
addfile.sh stoc /dev/rstoc126 371M &
addfile.sh stoc /dev/rstoc127 371M &
addfile.sh stoc /dev/rstoc128 371M &
addfile.sh stoc /dev/rstoc129 371M &
addfile.sh stoc /dev/rstoc130 371M &
addfile.sh stoc /dev/rstoc131 371M &
addfile.sh stoc /dev/rstoc132 371M &

addfile.sh stoc /dev/rstoc133 371M &
addfile.sh stoc /dev/rstoc134 371M &
addfile.sh stoc /dev/rstoc135 371M &
addfile.sh stoc /dev/rstoc136 371M &
addfile.sh stoc /dev/rstoc137 371M &
addfile.sh stoc /dev/rstoc138 371M &
addfile.sh stoc /dev/rstoc139 371M &
addfile.sh stoc /dev/rstoc140 371M &
addfile.sh stoc /dev/rstoc141 371M &
addfile.sh stoc /dev/rstoc142 371M &
addfile.sh stoc /dev/rstoc143 371M &
addfile.sh stoc /dev/rstoc144 371M &
addfile.sh stoc /dev/rstoc145 371M &
addfile.sh stoc /dev/rstoc146 371M &
addfile.sh stoc /dev/rstoc147 371M &
addfile.sh stoc /dev/rstoc148 371M &
addfile.sh stoc /dev/rstoc149 371M &
addfile.sh stoc /dev/rstoc150 371M &
addfile.sh stoc /dev/rstoc151 371M &
addfile.sh stoc /dev/rstoc152 371M &
addfile.sh stoc /dev/rstoc153 371M &
addfile.sh stoc /dev/rstoc154 371M &

addfile.sh stoc /dev/rstoc155 371M &
addfile.sh stoc /dev/rstoc156 371M &
addfile.sh stoc /dev/rstoc157 371M &
addfile.sh stoc /dev/rstoc158 371M &
addfile.sh stoc /dev/rstoc159 371M &
addfile.sh stoc /dev/rstoc160 371M &
addfile.sh stoc /dev/rstoc161 371M &
addfile.sh stoc /dev/rstoc162 371M &
addfile.sh stoc /dev/rstoc163 371M &
addfile.sh stoc /dev/rstoc164 371M &
addfile.sh stoc /dev/rstoc165 371M &
addfile.sh stoc /dev/rstoc166 371M &
addfile.sh stoc /dev/rstoc167 371M &
addfile.sh stoc /dev/rstoc168 371M &
addfile.sh stoc /dev/rstoc169 371M &
addfile.sh stoc /dev/rstoc170 371M &
addfile.sh stoc /dev/rstoc171 371M &
addfile.sh stoc /dev/rstoc172 371M &
addfile.sh stoc /dev/rstoc173 371M &
addfile.sh stoc /dev/rstoc174 371M &
addfile.sh stoc /dev/rstoc175 371M &
addfile.sh stoc /dev/rstoc176 371M &

addfile.sh stoc /dev/rstoc177 371M &
addfile.sh stoc /dev/rstoc178 371M &
addfile.sh stoc /dev/rstoc179 371M &
addfile.sh stoc /dev/rstoc180 371M &
addfile.sh stoc /dev/rstoc181 371M &
addfile.sh stoc /dev/rstoc182 371M &
addfile.sh stoc /dev/rstoc183 371M &
addfile.sh stoc /dev/rstoc184 371M &
addfile.sh stoc /dev/rstoc185 371M &
addfile.sh stoc /dev/rstoc186 371M &
addfile.sh stoc /dev/rstoc187 371M &
addfile.sh stoc /dev/rstoc188 371M &
addfile.sh stoc /dev/rstoc189 371M &
addfile.sh stoc /dev/rstoc190 371M &
addfile.sh stoc /dev/rstoc191 371M &
addfile.sh stoc /dev/rstoc192 371M &
addfile.sh stoc /dev/rstoc193 371M &
addfile.sh stoc /dev/rstoc194 371M &
addfile.sh stoc /dev/rstoc195 371M &
addfile.sh stoc /dev/rstoc196 371M &
addfile.sh stoc /dev/rstoc197 371M &
addfile.sh stoc /dev/rstoc198 371M &

addfile.sh stoc /dev/rstoc199 371M &
addfile.sh stoc /dev/rstoc200 371M &
addfile.sh stoc /dev/rstoc201 371M &
addfile.sh stoc /dev/rstoc202 371M &
addfile.sh stoc /dev/rstoc203 371M &
addfile.sh stoc /dev/rstoc204 371M &
addfile.sh stoc /dev/rstoc205 371M &
addfile.sh stoc /dev/rstoc206 371M &
addfile.sh stoc /dev/rstoc207 371M &
addfile.sh stoc /dev/rstoc208 371M &
addfile.sh stoc /dev/rstoc209 371M &
addfile.sh stoc /dev/rstoc210 371M &
addfile.sh stoc /dev/rstoc211 371M &
addfile.sh stoc /dev/rstoc212 371M &
addfile.sh stoc /dev/rstoc213 371M &
addfile.sh stoc /dev/rstoc214 371M &
addfile.sh stoc /dev/rstoc215 371M &
addfile.sh stoc /dev/rstoc216 371M &
addfile.sh stoc /dev/rstoc217 371M &
addfile.sh stoc /dev/rstoc218 371M &
addfile.sh stoc /dev/rstoc219 371M &
addfile.sh stoc /dev/rstoc220 371M &
addfile.sh stoc /dev/rstoc221 371M &
addfile.sh stoc /dev/rstoc222 371M &

wait

#############################################################################
#
#  FILE:  cr_n2_temp_ts.sh
#
#       Create temptablespace form node2
#
#############################################################################

. ~/.profile

addfile.sh temp /dev/rtemp022 1999M &
addfile.sh temp /dev/rtemp023 1999M &
addfile.sh temp /dev/rtemp024 1999M &
addfile.sh temp /dev/rtemp025 1999M &
addfile.sh temp /dev/rtemp026 1999M &
addfile.sh temp /dev/rtemp027 1999M &
addfile.sh temp /dev/rtemp028 1999M &
addfile.sh temp /dev/rtemp029 1999M &
addfile.sh temp /dev/rtemp030 1999M &
addfile.sh temp /dev/rtemp031 1999M &
addfile.sh temp /dev/rtemp032 1999M &
addfile.sh temp /dev/rtemp033 1999M &
addfile.sh temp /dev/rtemp034 1999M &
addfile.sh temp /dev/rtemp035 1999M &
addfile.sh temp /dev/rtemp036 1999M &
addfile.sh temp /dev/rtemp037 1999M &
addfile.sh temp /dev/rtemp038 1999M &
addfile.sh temp /dev/rtemp039 1999M &
addfile.sh temp /dev/rtemp040 1999M &
addfile.sh temp /dev/rtemp041 1999M &
addfile.sh temp /dev/rtemp042 1999M &


wait


#############################################################################
#
#  FILE:  cr_n3_cust_ts.sh
#
#       Createcust tablespace on node3
#
#############################################################################

. ~/.profile


addfile.sh cust /dev/rcust177 771M &
addfile.sh cust /dev/rcust178 771M &
addfile.sh cust /dev/rcust179 771M &
addfile.sh cust /dev/rcust180 771M &
addfile.sh cust /dev/rcust181 771M &
addfile.sh cust /dev/rcust182 771M &
addfile.sh cust /dev/rcust183 771M &
```

```
addfile.sh cust /dev/rcust184 771M &
addfile.sh cust /dev/rcust185 771M &
addfile.sh cust /dev/rcust186 771M &
addfile.sh cust /dev/rcust187 771M &
addfile.sh cust /dev/rcust188 771M &
addfile.sh cust /dev/rcust189 771M &
addfile.sh cust /dev/rcust190 771M &
addfile.sh cust /dev/rcust191 771M &
addfile.sh cust /dev/rcust192 771M &
addfile.sh cust /dev/rcust193 771M &
addfile.sh cust /dev/rcust194 771M &
addfile.sh cust /dev/rcust195 771M &
addfile.sh cust /dev/rcust196 771M &
addfile.sh cust /dev/rcust197 771M &
addfile.sh cust /dev/rcust198 771M &

wait

addfile.sh cust /dev/rcust199 771M &
addfile.sh cust /dev/rcust200 771M &
addfile.sh cust /dev/rcust201 771M &
addfile.sh cust /dev/rcust202 771M &
addfile.sh cust /dev/rcust203 771M &
addfile.sh cust /dev/rcust204 771M &
addfile.sh cust /dev/rcust205 771M &
addfile.sh cust /dev/rcust206 771M &
addfile.sh cust /dev/rcust207 771M &
addfile.sh cust /dev/rcust208 771M &
addfile.sh cust /dev/rcust209 771M &
addfile.sh cust /dev/rcust210 771M &
addfile.sh cust /dev/rcust211 771M &
addfile.sh cust /dev/rcust212 771M &
addfile.sh cust /dev/rcust213 771M &
addfile.sh cust /dev/rcust214 771M &
addfile.sh cust /dev/rcust215 771M &
addfile.sh cust /dev/rcust216 771M &
addfile.sh cust /dev/rcust217 771M &
addfile.sh cust /dev/rcust218 771M &
addfile.sh cust /dev/rcust219 771M &
addfile.sh cust /dev/rcust220 771M &

wait

addfile.sh cust /dev/rcust221 771M &
addfile.sh cust /dev/rcust222 771M &
addfile.sh cust /dev/rcust223 771M &
addfile.sh cust /dev/rcust224 771M &
addfile.sh cust /dev/rcust225 771M &
addfile.sh cust /dev/rcust226 771M &
addfile.sh cust /dev/rcust227 771M &
addfile.sh cust /dev/rcust228 771M &
addfile.sh cust /dev/rcust229 771M &
addfile.sh cust /dev/rcust230 771M &
addfile.sh cust /dev/rcust231 771M &
addfile.sh cust /dev/rcust232 771M &
addfile.sh cust /dev/rcust233 771M &
addfile.sh cust /dev/rcust234 771M &
addfile.sh cust /dev/rcust235 771M &
addfile.sh cust /dev/rcust236 771M &
addfile.sh cust /dev/rcust237 771M &
addfile.sh cust /dev/rcust238 771M &
addfile.sh cust /dev/rcust239 771M &
addfile.sh cust /dev/rcust240 771M &
addfile.sh cust /dev/rcust241 771M &
addfile.sh cust /dev/rcust242 771M &

wait

addfile.sh cust /dev/rcust243 771M &
addfile.sh cust /dev/rcust244 771M &
addfile.sh cust /dev/rcust245 771M &
addfile.sh cust /dev/rcust246 771M &
addfile.sh cust /dev/rcust247 771M &
addfile.sh cust /dev/rcust248 771M &
addfile.sh cust /dev/rcust249 771M &
addfile.sh cust /dev/rcust250 771M &
addfile.sh cust /dev/rcust251 771M &
addfile.sh cust /dev/rcust252 771M &
addfile.sh cust /dev/rcust253 771M &
addfile.sh cust /dev/rcust254 771M &
addfile.sh cust /dev/rcust255 771M &
addfile.sh cust /dev/rcust256 771M &
addfile.sh cust /dev/rcust257 771M &
addfile.sh cust /dev/rcust258 771M &
addfile.sh cust /dev/rcust259 771M &
addfile.sh cust /dev/rcust260 771M &
addfile.sh cust /dev/rcust261 771M &
addfile.sh cust /dev/rcust262 771M &
addfile.sh cust /dev/rcust263 771M &
addfile.sh cust /dev/rcust264 771M &

wait

################################################################################
#
#   FILE:  cr_n3_iord_ts.sh
#
#         Create ior1 and ior2 tablespaces on node3
#
################################################################################

. ~/.profile
set -x
svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ior1_n3 datafile '/dev/rior1021' size 607M reuse;
      exit;
!

addfile.sh ior1_n3  /dev/rior1022 607M &
addfile.sh ior1_n3  /dev/rior1023 607M &
addfile.sh ior1_n3  /dev/rior1024 607M &
addfile.sh ior1_n3  /dev/rior1025 607M &
addfile.sh ior1_n3  /dev/rior1026 607M &
addfile.sh ior1_n3  /dev/rior1027 607M &
addfile.sh ior1_n3  /dev/rior1028 607M &
addfile.sh ior1_n3  /dev/rior1029 607M &
addfile.sh ior1_n3  /dev/rior1030 607M &

wait
date
echo "Done: cr_n3_iord1_ts.sh"


################################################################################
#
#   FILE:  cr_n3_iord_ts.sh
#
#         Create ior1 and ior2 tablespaces on node3
#
################################################################################

. ~/.profile

set -x
svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace ior2_n3 datafile   '/dev/rior2041'   size 511M reuse;
  exit;
!

addfile.sh ior2_n3  /dev/rior2042 511M &
addfile.sh ior2_n3  /dev/rior2043 511M &
addfile.sh ior2_n3  /dev/rior2044 511M &
addfile.sh ior2_n3  /dev/rior2045 511M &
addfile.sh ior2_n3  /dev/rior2046 511M &
addfile.sh ior2_n3  /dev/rior2047 511M &
addfile.sh ior2_n3  /dev/rior2048 511M &
addfile.sh ior2_n3  /dev/rior2049 511M &
addfile.sh ior2_n3  /dev/rior2050 511M &
addfile.sh ior2_n3  /dev/rior2051 511M &
addfile.sh ior2_n3  /dev/rior2052 511M &
addfile.sh ior2_n3  /dev/rior2053 511M &
addfile.sh ior2_n3  /dev/rior2054 511M &
addfile.sh ior2_n3  /dev/rior2055 511M &
addfile.sh ior2_n3  /dev/rior2056 511M &
addfile.sh ior2_n3  /dev/rior2057 511M &
addfile.sh ior2_n3  /dev/rior2058 511M &
addfile.sh ior2_n3  /dev/rior2059 511M &
addfile.sh ior2_n3  /dev/rior2060 511M &
wait
date
echo "Done: cr_n3_iord2_ts.sh"

################################################################################
#
#   FILE:  cr_n3_iorl_ts.sh
#
#         Create iorl tablespace  on node3
#
################################################################################

. ~/.profile

svrmgrl <<!
      set echo on
      connect system/manager

      create tablespace iorl_n3_1 datafile   '/dev/riorl045'   size 2127M reuse;
      create tablespace iorl_n3_2 datafile   '/dev/riorl056'   size 2127M reuse;

      exit;
!

addfile.sh iorl_n3_1  /dev/riorl046 2127M &
addfile.sh iorl_n3_1  /dev/riorl047 2127M &
addfile.sh iorl_n3_1  /dev/riorl048 2127M &
addfile.sh iorl_n3_1  /dev/riorl049 2127M &
addfile.sh iorl_n3_1  /dev/riorl050 2127M &
addfile.sh iorl_n3_1  /dev/riorl051 2127M &
addfile.sh iorl_n3_1  /dev/riorl052 2127M &
addfile.sh iorl_n3_1  /dev/riorl053 2127M &
addfile.sh iorl_n3_1  /dev/riorl054 2127M &
addfile.sh iorl_n3_1  /dev/riorl055 2127M &

wait
```

```
addfile.sh iorl_n3_2  /dev/riorl057 2127M &
addfile.sh iorl_n3_2  /dev/riorl058 2127M &
addfile.sh iorl_n3_2  /dev/riorl059 2127M &
addfile.sh iorl_n3_2  /dev/riorl060 2127M &
addfile.sh iorl_n3_2  /dev/riorl061 2127M &
addfile.sh iorl_n3_2  /dev/riorl062 2127M &
addfile.sh iorl_n3_2  /dev/riorl063 2127M &
addfile.sh iorl_n3_2  /dev/riorl064 2127M &
addfile.sh iorl_n3_2  /dev/riorl065 2127M &
addfile.sh iorl_n3_2  /dev/riorl066 2127M &

wait

################################################################################
#
#  FILE:  cr_n3_istk_ts.sh
#
#        Createistk tablespace on node3
#
################################################################################

. ~/.profile

addfile.sh istk  /dev/ristk009 1126M &
addfile.sh istk  /dev/ristk010 1126M &
addfile.sh istk  /dev/ristk011 1126M &
addfile.sh istk  /dev/ristk012 1126M &

wait


################################################################################
#
#  FILE:  cr_n3_ordl_ts.sh
#
#        CreateORDL tablespaces on node3
#
################################################################################

. ~/.profile
set -x

echo "starting Node 3 - `date`"
svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ordl_n3 datafile '/dev/rordl161' size 1391M reuse;
      exit;
!

addfile.sh ordl_n3 /dev/rordl162 1391M &
addfile.sh ordl_n3 /dev/rordl163 1391M &
addfile.sh ordl_n3 /dev/rordl164 1391M &
addfile.sh ordl_n3 /dev/rordl165 1391M &
addfile.sh ordl_n3 /dev/rordl166 1391M &
addfile.sh ordl_n3 /dev/rordl167 1391M &
addfile.sh ordl_n3 /dev/rordl168 1391M &
addfile.sh ordl_n3 /dev/rordl169 1391M &
addfile.sh ordl_n3 /dev/rordl170 1391M &
addfile.sh ordl_n3 /dev/rordl171 1391M &
addfile.sh ordl_n3 /dev/rordl172 1391M &
addfile.sh ordl_n3 /dev/rordl173 1391M &
addfile.sh ordl_n3 /dev/rordl174 1391M &
addfile.sh ordl_n3 /dev/rordl175 1391M &
addfile.sh ordl_n3 /dev/rordl176 1391M &
addfile.sh ordl_n3 /dev/rordl177 1391M &
addfile.sh ordl_n3 /dev/rordl178 1391M &
addfile.sh ordl_n3 /dev/rordl179 1391M &
addfile.sh ordl_n3 /dev/rordl180 1391M &

#wait

addfile.sh ordl_n3 /dev/rordl181 1391M &
addfile.sh ordl_n3 /dev/rordl182 1391M &
addfile.sh ordl_n3 /dev/rordl183 1391M &
addfile.sh ordl_n3 /dev/rordl184 1391M &
addfile.sh ordl_n3 /dev/rordl185 1391M &
addfile.sh ordl_n3 /dev/rordl186 1391M &
addfile.sh ordl_n3 /dev/rordl187 1391M &
addfile.sh ordl_n3 /dev/rordl188 1391M &
addfile.sh ordl_n3 /dev/rordl189 1391M &
addfile.sh ordl_n3 /dev/rordl190 1391M &
addfile.sh ordl_n3 /dev/rordl191 1391M &
addfile.sh ordl_n3 /dev/rordl192 1391M &
addfile.sh ordl_n3 /dev/rordl193 1391M &
addfile.sh ordl_n3 /dev/rordl194 1391M &
addfile.sh ordl_n3 /dev/rordl195 1391M &
addfile.sh ordl_n3 /dev/rordl196 1391M &
addfile.sh ordl_n3 /dev/rordl197 1391M &
addfile.sh ordl_n3 /dev/rordl198 1391M &
addfile.sh ordl_n3 /dev/rordl199 1391M &
addfile.sh ordl_n3 /dev/rordl200 1391M &

#wait

addfile.sh ordl_n3 /dev/rordl201 1391M &
addfile.sh ordl_n3 /dev/rordl202 1391M &
addfile.sh ordl_n3 /dev/rordl203 1391M &
addfile.sh ordl_n3 /dev/rordl204 1391M &
addfile.sh ordl_n3 /dev/rordl205 1391M &
addfile.sh ordl_n3 /dev/rordl206 1391M &
addfile.sh ordl_n3 /dev/rordl207 1391M &
addfile.sh ordl_n3 /dev/rordl208 1391M &
addfile.sh ordl_n3 /dev/rordl209 1391M &
addfile.sh ordl_n3 /dev/rordl210 1391M &
addfile.sh ordl_n3 /dev/rordl211 1391M &
addfile.sh ordl_n3 /dev/rordl212 1391M &
addfile.sh ordl_n3 /dev/rordl213 1391M &
addfile.sh ordl_n3 /dev/rordl214 1391M &
addfile.sh ordl_n3 /dev/rordl215 1391M &
addfile.sh ordl_n3 /dev/rordl216 1391M &
addfile.sh ordl_n3 /dev/rordl217 1391M &
addfile.sh ordl_n3 /dev/rordl218 1391M &
addfile.sh ordl_n3 /dev/rordl219 1391M &
addfile.sh ordl_n3 /dev/rordl220 1391M &

#wait

addfile.sh ordl_n3 /dev/rordl221 1391M &
addfile.sh ordl_n3 /dev/rordl222 1391M &
addfile.sh ordl_n3 /dev/rordl223 1391M &
addfile.sh ordl_n3 /dev/rordl224 1391M &
addfile.sh ordl_n3 /dev/rordl225 1391M &
addfile.sh ordl_n3 /dev/rordl226 1391M &
addfile.sh ordl_n3 /dev/rordl227 1391M &
addfile.sh ordl_n3 /dev/rordl228 1391M &
addfile.sh ordl_n3 /dev/rordl229 1391M &
addfile.sh ordl_n3 /dev/rordl230 1391M &
addfile.sh ordl_n3 /dev/rordl231 1391M &
addfile.sh ordl_n3 /dev/rordl232 1391M &
addfile.sh ordl_n3 /dev/rordl233 1391M &
addfile.sh ordl_n3 /dev/rordl234 1391M &
addfile.sh ordl_n3 /dev/rordl235 1391M &
addfile.sh ordl_n3 /dev/rordl236 1391M &
addfile.sh ordl_n3 /dev/rordl237 1391M &
addfile.sh ordl_n3 /dev/rordl238 1391M &
addfile.sh ordl_n3 /dev/rordl239 1391M &
addfile.sh ordl_n3 /dev/rordl240 1391M &

wait

echo "cr_n1_ordl_ts.sh - `date`"


################################################################################
#
#  FILE:  cr_n3_ords_ts.sh
#
#        Createords tablespace on node3
#
################################################################################

. ~/.profile
set -x

svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ords_n3 datafile '/dev/rords021' size 607M reuse;
      exit;
!
addfile.sh ords_n3  /dev/rords022 607M &
addfile.sh ords_n3  /dev/rords023 607M &
addfile.sh ords_n3  /dev/rords024 607M &
addfile.sh ords_n3  /dev/rords025 607M &
addfile.sh ords_n3  /dev/rords026 607M &
addfile.sh ords_n3  /dev/rords027 607M &
addfile.sh ords_n3  /dev/rords028 607M &
addfile.sh ords_n3  /dev/rords029 607M &
addfile.sh ords_n3  /dev/rords030 607M &

wait
date
echo "Done: cr_n3_ords_ts.sh"


################################################################################
#
#  FILE:  cr_n3_stoc_ts.sh
#
#        Createstoc tablespace on node3
#
################################################################################

. ~/.profile

addfile.sh stoc /dev/rstoc445 371M &
addfile.sh stoc /dev/rstoc446 371M &
addfile.sh stoc /dev/rstoc447 371M &
addfile.sh stoc /dev/rstoc448 371M &
addfile.sh stoc /dev/rstoc449 371M &
addfile.sh stoc /dev/rstoc450 371M &
addfile.sh stoc /dev/rstoc451 371M &
addfile.sh stoc /dev/rstoc452 371M &
addfile.sh stoc /dev/rstoc453 371M &
addfile.sh stoc /dev/rstoc454 371M &
addfile.sh stoc /dev/rstoc455 371M &
```

```
addfile.sh stoc /dev/rstoc456 371M &
addfile.sh stoc /dev/rstoc457 371M &
addfile.sh stoc /dev/rstoc458 371M &
addfile.sh stoc /dev/rstoc459 371M &
addfile.sh stoc /dev/rstoc460 371M &
addfile.sh stoc /dev/rstoc461 371M &
addfile.sh stoc /dev/rstoc462 371M &
addfile.sh stoc /dev/rstoc463 371M &
addfile.sh stoc /dev/rstoc464 371M &
addfile.sh stoc /dev/rstoc465 371M &
addfile.sh stoc /dev/rstoc466 371M &

addfile.sh stoc /dev/rstoc467 371M &
addfile.sh stoc /dev/rstoc468 371M &
addfile.sh stoc /dev/rstoc469 371M &
addfile.sh stoc /dev/rstoc470 371M &
addfile.sh stoc /dev/rstoc471 371M &
addfile.sh stoc /dev/rstoc472 371M &
addfile.sh stoc /dev/rstoc473 371M &
addfile.sh stoc /dev/rstoc474 371M &
addfile.sh stoc /dev/rstoc475 371M &
addfile.sh stoc /dev/rstoc476 371M &
addfile.sh stoc /dev/rstoc477 371M &
addfile.sh stoc /dev/rstoc478 371M &
addfile.sh stoc /dev/rstoc479 371M &
addfile.sh stoc /dev/rstoc480 371M &
addfile.sh stoc /dev/rstoc481 371M &
addfile.sh stoc /dev/rstoc482 371M &
addfile.sh stoc /dev/rstoc483 371M &
addfile.sh stoc /dev/rstoc484 371M &
addfile.sh stoc /dev/rstoc485 371M &
addfile.sh stoc /dev/rstoc486 371M &
addfile.sh stoc /dev/rstoc487 371M &
addfile.sh stoc /dev/rstoc488 371M &

addfile.sh stoc /dev/rstoc489 371M &
addfile.sh stoc /dev/rstoc490 371M &
addfile.sh stoc /dev/rstoc491 371M &
addfile.sh stoc /dev/rstoc492 371M &
addfile.sh stoc /dev/rstoc493 371M &
addfile.sh stoc /dev/rstoc494 371M &
addfile.sh stoc /dev/rstoc495 371M &
addfile.sh stoc /dev/rstoc496 371M &
addfile.sh stoc /dev/rstoc497 371M &
addfile.sh stoc /dev/rstoc498 371M &
addfile.sh stoc /dev/rstoc499 371M &
addfile.sh stoc /dev/rstoc500 371M &
addfile.sh stoc /dev/rstoc501 371M &
addfile.sh stoc /dev/rstoc502 371M &
addfile.sh stoc /dev/rstoc503 371M &
addfile.sh stoc /dev/rstoc504 371M &
addfile.sh stoc /dev/rstoc505 371M &
addfile.sh stoc /dev/rstoc506 371M &
addfile.sh stoc /dev/rstoc507 371M &
addfile.sh stoc /dev/rstoc508 371M &
addfile.sh stoc /dev/rstoc509 371M &
addfile.sh stoc /dev/rstoc510 371M &

addfile.sh stoc /dev/rstoc511 371M &
addfile.sh stoc /dev/rstoc512 371M &
addfile.sh stoc /dev/rstoc513 371M &
addfile.sh stoc /dev/rstoc514 371M &
addfile.sh stoc /dev/rstoc515 371M &
addfile.sh stoc /dev/rstoc516 371M &
addfile.sh stoc /dev/rstoc517 371M &
addfile.sh stoc /dev/rstoc518 371M &
addfile.sh stoc /dev/rstoc519 371M &
addfile.sh stoc /dev/rstoc520 371M &
addfile.sh stoc /dev/rstoc521 371M &
addfile.sh stoc /dev/rstoc522 371M &
addfile.sh stoc /dev/rstoc523 371M &
addfile.sh stoc /dev/rstoc524 371M &
addfile.sh stoc /dev/rstoc525 371M &
addfile.sh stoc /dev/rstoc526 371M &
addfile.sh stoc /dev/rstoc527 371M &
addfile.sh stoc /dev/rstoc528 371M &
addfile.sh stoc /dev/rstoc529 371M &
addfile.sh stoc /dev/rstoc530 371M &
addfile.sh stoc /dev/rstoc531 371M &
addfile.sh stoc /dev/rstoc532 371M &

addfile.sh stoc /dev/rstoc533 371M &
addfile.sh stoc /dev/rstoc534 371M &
addfile.sh stoc /dev/rstoc535 371M &
addfile.sh stoc /dev/rstoc536 371M &
addfile.sh stoc /dev/rstoc537 371M &
addfile.sh stoc /dev/rstoc538 371M &
addfile.sh stoc /dev/rstoc539 371M &
addfile.sh stoc /dev/rstoc540 371M &
addfile.sh stoc /dev/rstoc541 371M &
addfile.sh stoc /dev/rstoc542 371M &
addfile.sh stoc /dev/rstoc543 371M &
addfile.sh stoc /dev/rstoc544 371M &
addfile.sh stoc /dev/rstoc545 371M &
addfile.sh stoc /dev/rstoc546 371M &
addfile.sh stoc /dev/rstoc547 371M &
addfile.sh stoc /dev/rstoc548 371M &
addfile.sh stoc /dev/rstoc549 371M &
addfile.sh stoc /dev/rstoc550 371M &
addfile.sh stoc /dev/rstoc551 371M &
addfile.sh stoc /dev/rstoc552 371M &
addfile.sh stoc /dev/rstoc553 371M &
addfile.sh stoc /dev/rstoc554 371M &

addfile.sh stoc /dev/rstoc555 371M &
addfile.sh stoc /dev/rstoc556 371M &
addfile.sh stoc /dev/rstoc557 371M &
addfile.sh stoc /dev/rstoc558 371M &
addfile.sh stoc /dev/rstoc559 371M &
addfile.sh stoc /dev/rstoc560 371M &
addfile.sh stoc /dev/rstoc561 371M &
addfile.sh stoc /dev/rstoc562 371M &
addfile.sh stoc /dev/rstoc563 371M &
addfile.sh stoc /dev/rstoc564 371M &
addfile.sh stoc /dev/rstoc565 371M &
addfile.sh stoc /dev/rstoc566 371M &
addfile.sh stoc /dev/rstoc567 371M &
addfile.sh stoc /dev/rstoc568 371M &
addfile.sh stoc /dev/rstoc569 371M &
addfile.sh stoc /dev/rstoc570 371M &
addfile.sh stoc /dev/rstoc571 371M &
addfile.sh stoc /dev/rstoc572 371M &
addfile.sh stoc /dev/rstoc573 371M &
addfile.sh stoc /dev/rstoc574 371M &
addfile.sh stoc /dev/rstoc575 371M &
addfile.sh stoc /dev/rstoc576 371M &

addfile.sh stoc /dev/rstoc577 371M &
addfile.sh stoc /dev/rstoc578 371M &
addfile.sh stoc /dev/rstoc579 371M &
addfile.sh stoc /dev/rstoc580 371M &
addfile.sh stoc /dev/rstoc581 371M &
addfile.sh stoc /dev/rstoc582 371M &
addfile.sh stoc /dev/rstoc583 371M &
addfile.sh stoc /dev/rstoc584 371M &
addfile.sh stoc /dev/rstoc585 371M &
addfile.sh stoc /dev/rstoc586 371M &
addfile.sh stoc /dev/rstoc587 371M &
addfile.sh stoc /dev/rstoc588 371M &
addfile.sh stoc /dev/rstoc589 371M &
addfile.sh stoc /dev/rstoc590 371M &
addfile.sh stoc /dev/rstoc591 371M &
addfile.sh stoc /dev/rstoc592 371M &
addfile.sh stoc /dev/rstoc593 371M &
addfile.sh stoc /dev/rstoc594 371M &
addfile.sh stoc /dev/rstoc595 371M &
addfile.sh stoc /dev/rstoc596 371M &
addfile.sh stoc /dev/rstoc597 371M &
addfile.sh stoc /dev/rstoc598 371M &

addfile.sh stoc /dev/rstoc599 371M &
addfile.sh stoc /dev/rstoc600 371M &
addfile.sh stoc /dev/rstoc601 371M &
addfile.sh stoc /dev/rstoc602 371M &
addfile.sh stoc /dev/rstoc603 371M &
addfile.sh stoc /dev/rstoc604 371M &
addfile.sh stoc /dev/rstoc605 371M &
addfile.sh stoc /dev/rstoc606 371M &
addfile.sh stoc /dev/rstoc607 371M &
addfile.sh stoc /dev/rstoc608 371M &
addfile.sh stoc /dev/rstoc609 371M &
addfile.sh stoc /dev/rstoc610 371M &
addfile.sh stoc /dev/rstoc611 371M &
addfile.sh stoc /dev/rstoc612 371M &
addfile.sh stoc /dev/rstoc613 371M &
addfile.sh stoc /dev/rstoc614 371M &
addfile.sh stoc /dev/rstoc615 371M &
addfile.sh stoc /dev/rstoc616 371M &
addfile.sh stoc /dev/rstoc617 371M &
addfile.sh stoc /dev/rstoc618 371M &
addfile.sh stoc /dev/rstoc619 371M &
addfile.sh stoc /dev/rstoc620 371M &

addfile.sh stoc /dev/rstoc621 371M &
addfile.sh stoc /dev/rstoc622 371M &
addfile.sh stoc /dev/rstoc623 371M &
addfile.sh stoc /dev/rstoc624 371M &
addfile.sh stoc /dev/rstoc625 371M &
addfile.sh stoc /dev/rstoc626 371M &
addfile.sh stoc /dev/rstoc627 371M &
addfile.sh stoc /dev/rstoc628 371M &
addfile.sh stoc /dev/rstoc629 371M &
addfile.sh stoc /dev/rstoc630 371M &
addfile.sh stoc /dev/rstoc631 371M &
addfile.sh stoc /dev/rstoc632 371M &
addfile.sh stoc /dev/rstoc633 371M &
addfile.sh stoc /dev/rstoc634 371M &
addfile.sh stoc /dev/rstoc635 371M &
addfile.sh stoc /dev/rstoc636 371M &
addfile.sh stoc /dev/rstoc637 371M &
addfile.sh stoc /dev/rstoc638 371M &
addfile.sh stoc /dev/rstoc639 371M &
addfile.sh stoc /dev/rstoc640 371M &
addfile.sh stoc /dev/rstoc641 371M &
addfile.sh stoc /dev/rstoc642 371M &

addfile.sh stoc /dev/rstoc643 371M &
addfile.sh stoc /dev/rstoc644 371M &
```

```
addfile.sh stoc /dev/rstoc645 371M &
addfile.sh stoc /dev/rstoc646 371M &
addfile.sh stoc /dev/rstoc647 371M &
addfile.sh stoc /dev/rstoc648 371M &
addfile.sh stoc /dev/rstoc649 371M &
addfile.sh stoc /dev/rstoc650 371M &
addfile.sh stoc /dev/rstoc651 371M &
addfile.sh stoc /dev/rstoc652 371M &
addfile.sh stoc /dev/rstoc653 371M &
addfile.sh stoc /dev/rstoc654 371M &
addfile.sh stoc /dev/rstoc655 371M &
addfile.sh stoc /dev/rstoc656 371M &
addfile.sh stoc /dev/rstoc657 371M &
addfile.sh stoc /dev/rstoc658 371M &
addfile.sh stoc /dev/rstoc659 371M &
addfile.sh stoc /dev/rstoc660 371M &
addfile.sh stoc /dev/rstoc661 371M &
addfile.sh stoc /dev/rstoc662 371M &
addfile.sh stoc /dev/rstoc663 371M &
addfile.sh stoc /dev/rstoc664 371M &
addfile.sh stoc /dev/rstoc665 371M &
addfile.sh stoc /dev/rstoc666 371M &
wait


################################################################################
#
#  FILE:  cr_n3_temp_ts.sh
#
#       Create temp tablespace form node3
#
################################################################################

. ~/.profile

addfile.sh temp /dev/rtemp043 1999M &
addfile.sh temp /dev/rtemp044 1999M &
addfile.sh temp /dev/rtemp045 1999M &
addfile.sh temp /dev/rtemp046 1999M &
addfile.sh temp /dev/rtemp047 1999M &
addfile.sh temp /dev/rtemp048 1999M &
addfile.sh temp /dev/rtemp049 1999M &
addfile.sh temp /dev/rtemp050 1999M &
addfile.sh temp /dev/rtemp051 1999M &
addfile.sh temp /dev/rtemp052 1999M &
addfile.sh temp /dev/rtemp053 1999M &
addfile.sh temp /dev/rtemp054 1999M &
addfile.sh temp /dev/rtemp055 1999M &
addfile.sh temp /dev/rtemp056 1999M &
addfile.sh temp /dev/rtemp057a 1999M &
addfile.sh temp /dev/rtemp058 1999M &
addfile.sh temp /dev/rtemp059 1999M &
addfile.sh temp /dev/rtemp060 1999M &
addfile.sh temp /dev/rtemp061 1999M &
addfile.sh temp /dev/rtemp062 1999M &
addfile.sh temp /dev/rtemp063 1999M &


wait


################################################################################
#
#  FILE:  cr_n4_cust_ts.sh
#
#       Create cust tablespace on node4
#
################################################################################

. ~/.profile


addfile.sh cust /dev/rcust265 771M &
addfile.sh cust /dev/rcust266 771M &
addfile.sh cust /dev/rcust267 771M &
addfile.sh cust /dev/rcust268 771M &
addfile.sh cust /dev/rcust269 771M &
addfile.sh cust /dev/rcust270 771M &
addfile.sh cust /dev/rcust271 771M &
addfile.sh cust /dev/rcust272 771M &
addfile.sh cust /dev/rcust273 771M &
addfile.sh cust /dev/rcust274 771M &
addfile.sh cust /dev/rcust275 771M &
addfile.sh cust /dev/rcust276 771M &
addfile.sh cust /dev/rcust277 771M &
addfile.sh cust /dev/rcust278 771M &
addfile.sh cust /dev/rcust279 771M &
addfile.sh cust /dev/rcust280 771M &
addfile.sh cust /dev/rcust281 771M &
addfile.sh cust /dev/rcust282 771M &
addfile.sh cust /dev/rcust283 771M &
addfile.sh cust /dev/rcust284 771M &
addfile.sh cust /dev/rcust285 771M &
addfile.sh cust /dev/rcust286 771M &

wait

addfile.sh cust /dev/rcust287 771M &
addfile.sh cust /dev/rcust288 771M &
addfile.sh cust /dev/rcust289 771M &
addfile.sh cust /dev/rcust290 771M &
addfile.sh cust /dev/rcust291 771M &
addfile.sh cust /dev/rcust292 771M &
addfile.sh cust /dev/rcust293 771M &
addfile.sh cust /dev/rcust294 771M &
addfile.sh cust /dev/rcust295 771M &
addfile.sh cust /dev/rcust296 771M &
addfile.sh cust /dev/rcust297 771M &
addfile.sh cust /dev/rcust298 771M &
addfile.sh cust /dev/rcust299 771M &
addfile.sh cust /dev/rcust300 771M &
addfile.sh cust /dev/rcust301 771M &
addfile.sh cust /dev/rcust302 771M &
addfile.sh cust /dev/rcust303 771M &
addfile.sh cust /dev/rcust304 771M &
addfile.sh cust /dev/rcust305 771M &
addfile.sh cust /dev/rcust306 771M &
addfile.sh cust /dev/rcust307 771M &
addfile.sh cust /dev/rcust308 771M &

wait

addfile.sh cust /dev/rcust309 771M &
addfile.sh cust /dev/rcust310 771M &
addfile.sh cust /dev/rcust311 771M &
addfile.sh cust /dev/rcust312 771M &
addfile.sh cust /dev/rcust313 771M &
addfile.sh cust /dev/rcust314 771M &
addfile.sh cust /dev/rcust315 771M &
addfile.sh cust /dev/rcust316 771M &
addfile.sh cust /dev/rcust317 771M &
addfile.sh cust /dev/rcust318 771M &
addfile.sh cust /dev/rcust319 771M &
addfile.sh cust /dev/rcust320 771M &
addfile.sh cust /dev/rcust321 771M &
addfile.sh cust /dev/rcust322 771M &
addfile.sh cust /dev/rcust323 771M &
addfile.sh cust /dev/rcust324 771M &
addfile.sh cust /dev/rcust325 771M &
addfile.sh cust /dev/rcust326 771M &
addfile.sh cust /dev/rcust327 771M &
addfile.sh cust /dev/rcust328 771M &
addfile.sh cust /dev/rcust329 771M &
addfile.sh cust /dev/rcust330 771M &

wait

addfile.sh cust /dev/rcust331 771M &
addfile.sh cust /dev/rcust332 771M &
addfile.sh cust /dev/rcust333 771M &
addfile.sh cust /dev/rcust334 771M &
addfile.sh cust /dev/rcust335 771M &
addfile.sh cust /dev/rcust336 771M &
addfile.sh cust /dev/rcust337 771M &
addfile.sh cust /dev/rcust338 771M &
addfile.sh cust /dev/rcust339 771M &
addfile.sh cust /dev/rcust340 771M &
addfile.sh cust /dev/rcust341 771M &
addfile.sh cust /dev/rcust342 771M &
addfile.sh cust /dev/rcust343 771M &
addfile.sh cust /dev/rcust344 771M &
addfile.sh cust /dev/rcust345 771M &
addfile.sh cust /dev/rcust346 771M &
addfile.sh cust /dev/rcust347 771M &
addfile.sh cust /dev/rcust348 771M &
addfile.sh cust /dev/rcust349 771M &
addfile.sh cust /dev/rcust350 771M &
addfile.sh cust /dev/rcust351 771M &
addfile.sh cust /dev/rcust352 771M &

wait


################################################################################
#
#  FILE:  cr_n4_iord_ts.sh
#
#       Create ior1 and ior2 tablespaces on node4
#
################################################################################

. ~/.profile
set -x

svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ior1_n4 datafile '/dev/rior1031' size 607M reuse;
      exit;
!

addfile.sh ior1_n4  /dev/rior1032 607M &
addfile.sh ior1_n4  /dev/rior1033 607M &
addfile.sh ior1_n4  /dev/rior1034 607M &
addfile.sh ior1_n4  /dev/rior1035 607M &
addfile.sh ior1_n4  /dev/rior1036 607M &
addfile.sh ior1_n4  /dev/rior1037 607M &
addfile.sh ior1_n4  /dev/rior1038 607M &
```

```
addfile.sh ior1_n4  /dev/rior1039 607M &
addfile.sh ior1_n4  /dev/rior1040 607M &

wait
date
echo "Done: cr_n4_iord1_ts.sh"


################################################################################
#
#   FILE:  cr_n4_iord_ts.sh
#
#         Createior1 and ior2 tablespaces on node4
#
################################################################################

. ~/.profile
set -x

svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace ior2_n4 datafile    '/dev/rior2061'    size 511M reuse;
  exit;
!
addfile.sh ior2_n4  /dev/rior2062 511M &
addfile.sh ior2_n4  /dev/rior2063 511M &
addfile.sh ior2_n4  /dev/rior2064 511M &
addfile.sh ior2_n4  /dev/rior2065 511M &
addfile.sh ior2_n4  /dev/rior2066 511M &
addfile.sh ior2_n4  /dev/rior2067 511M &
addfile.sh ior2_n4  /dev/rior2068 511M &
addfile.sh ior2_n4  /dev/rior2069 511M &
addfile.sh ior2_n4  /dev/rior2070 511M &
addfile.sh ior2_n4  /dev/rior2071 511M &
addfile.sh ior2_n4  /dev/rior2072 511M &
addfile.sh ior2_n4  /dev/rior2073 511M &
addfile.sh ior2_n4  /dev/rior2074 511M &
addfile.sh ior2_n4  /dev/rior2075 511M &
addfile.sh ior2_n4  /dev/rior2076 511M &
addfile.sh ior2_n4  /dev/rior2077 511M &
addfile.sh ior2_n4  /dev/rior2078 511M &
addfile.sh ior2_n4  /dev/rior2079 511M &
addfile.sh ior2_n4  /dev/rior2080 511M &
wait
date
echo "Done: cr_n4_iord2_ts.sh"


################################################################################
#
#   FILE:  cr_n4_ordl_ts.sh
#
#         Create ORDL tablespaces on node4
#
################################################################################
. ~/.profile
set -x

echo "starting Node 4 - `date`"
svrmgrl <<!
        set echo on
        connect system/manager
        create tablespace ordl_n4 datafile '/dev/rordl241' size 1391M reuse;
        exit;
!

addfile.sh ordl_n4 /dev/rordl242 1391M &
addfile.sh ordl_n4 /dev/rordl243 1391M &
addfile.sh ordl_n4 /dev/rordl244 1391M &
addfile.sh ordl_n4 /dev/rordl245 1391M &
addfile.sh ordl_n4 /dev/rordl246 1391M &
addfile.sh ordl_n4 /dev/rordl247 1391M &
addfile.sh ordl_n4 /dev/rordl248 1391M &
addfile.sh ordl_n4 /dev/rordl249 1391M &
addfile.sh ordl_n4 /dev/rordl250 1391M &
addfile.sh ordl_n4 /dev/rordl251 1391M &
addfile.sh ordl_n4 /dev/rordl252 1391M &
addfile.sh ordl_n4 /dev/rordl253 1391M &
addfile.sh ordl_n4 /dev/rordl254 1391M &
addfile.sh ordl_n4 /dev/rordl255 1391M &
addfile.sh ordl_n4 /dev/rordl256 1391M &
addfile.sh ordl_n4 /dev/rordl257 1391M &
addfile.sh ordl_n4 /dev/rordl258 1391M &
addfile.sh ordl_n4 /dev/rordl259 1391M &
addfile.sh ordl_n4 /dev/rordl260 1391M &

#wait

addfile.sh ordl_n4 /dev/rordl261 1391M &
addfile.sh ordl_n4 /dev/rordl262 1391M &
addfile.sh ordl_n4 /dev/rordl263 1391M &
addfile.sh ordl_n4 /dev/rordl264 1391M &
addfile.sh ordl_n4 /dev/rordl265 1391M &
addfile.sh ordl_n4 /dev/rordl266 1391M &
addfile.sh ordl_n4 /dev/rordl267 1391M &
addfile.sh ordl_n4 /dev/rordl268 1391M &
addfile.sh ordl_n4 /dev/rordl269 1391M &
addfile.sh ordl_n4 /dev/rordl270 1391M &
addfile.sh ordl_n4 /dev/rordl271 1391M &
addfile.sh ordl_n4 /dev/rordl272 1391M &
addfile.sh ordl_n4 /dev/rordl273 1391M &
addfile.sh ordl_n4 /dev/rordl274 1391M &
addfile.sh ordl_n4 /dev/rordl275 1391M &
addfile.sh ordl_n4 /dev/rordl276 1391M &
addfile.sh ordl_n4 /dev/rordl277 1391M &
addfile.sh ordl_n4 /dev/rordl278 1391M &
addfile.sh ordl_n4 /dev/rordl279 1391M &
addfile.sh ordl_n4 /dev/rordl280 1391M &

wait

addfile.sh ordl_n4 /dev/rordl281 1391M &
addfile.sh ordl_n4 /dev/rordl282 1391M &
addfile.sh ordl_n4 /dev/rordl283 1391M &
addfile.sh ordl_n4 /dev/rordl284 1391M &
addfile.sh ordl_n4 /dev/rordl285 1391M &
addfile.sh ordl_n4 /dev/rordl286 1391M &
addfile.sh ordl_n4 /dev/rordl287 1391M &
addfile.sh ordl_n4 /dev/rordl288 1391M &
addfile.sh ordl_n4 /dev/rordl289 1391M &
addfile.sh ordl_n4 /dev/rordl290 1391M &
addfile.sh ordl_n4 /dev/rordl291 1391M &
addfile.sh ordl_n4 /dev/rordl292 1391M &
addfile.sh ordl_n4 /dev/rordl293 1391M &
addfile.sh ordl_n4 /dev/rordl294 1391M &
addfile.sh ordl_n4 /dev/rordl295 1391M &
addfile.sh ordl_n4 /dev/rordl296 1391M &
addfile.sh ordl_n4 /dev/rordl297 1391M &
addfile.sh ordl_n4 /dev/rordl298 1391M &
addfile.sh ordl_n4 /dev/rordl299 1391M &
addfile.sh ordl_n4 /dev/rordl300 1391M &

#wait

addfile.sh ordl_n4 /dev/rordl301 1391M &
addfile.sh ordl_n4 /dev/rordl302 1391M &
addfile.sh ordl_n4 /dev/rordl303 1391M &
addfile.sh ordl_n4 /dev/rordl304 1391M &
addfile.sh ordl_n4 /dev/rordl305 1391M &
addfile.sh ordl_n4 /dev/rordl306 1391M &
addfile.sh ordl_n4 /dev/rordl307 1391M &
addfile.sh ordl_n4 /dev/rordl308 1391M &
addfile.sh ordl_n4 /dev/rordl309 1391M &
addfile.sh ordl_n4 /dev/rordl310 1391M &
addfile.sh ordl_n4 /dev/rordl311 1391M &
addfile.sh ordl_n4 /dev/rordl312 1391M &
addfile.sh ordl_n4 /dev/rordl313 1391M &
addfile.sh ordl_n4 /dev/rordl314 1391M &
addfile.sh ordl_n4 /dev/rordl315 1391M &
addfile.sh ordl_n4 /dev/rordl316 1391M &
addfile.sh ordl_n4 /dev/rordl317 1391M &
addfile.sh ordl_n4 /dev/rordl318 1391M &
addfile.sh ordl_n4 /dev/rordl319 1391M &
addfile.sh ordl_n4 /dev/rordl320 1391M &

wait

echo "Done: cr_n4_ordl_ts.sh - `date`"

################################################################################
#
#   FILE:  cr_n4_istk_ts.sh
#
#         Create istk tablespace on node4
#
################################################################################

. ~/.profile

addfile.sh istk  /dev/ristk013 1126M &
addfile.sh istk  /dev/ristk014 1126M &
addfile.sh istk  /dev/ristk015 1126M &
addfile.sh istk  /dev/ristk016 1126M &

wait

################################################################################
#
#   FILE:  cr_n4_ordl_ts.sh
#
#         CreateORDL tablespaces on node4
#
################################################################################
. ~/.profile
set -x

echo "starting Node 4 - `date`"
svrmgrl <<!
        set echo on
        connect system/manager
        create tablespace ordl_n4 datafile '/dev/rordl241' size 1391M reuse;
        exit;
!

addfile.sh ordl_n4 /dev/rordl242 1391M &
addfile.sh ordl_n4 /dev/rordl243 1391M &
addfile.sh ordl_n4 /dev/rordl244 1391M &
```

```
addfile.sh ordl_n4 /dev/rordl245 1391M &
addfile.sh ordl_n4 /dev/rordl246 1391M &
addfile.sh ordl_n4 /dev/rordl247 1391M &
addfile.sh ordl_n4 /dev/rordl248 1391M &
addfile.sh ordl_n4 /dev/rordl249 1391M &
addfile.sh ordl_n4 /dev/rordl250 1391M &
addfile.sh ordl_n4 /dev/rordl251 1391M &
addfile.sh ordl_n4 /dev/rordl252 1391M &
addfile.sh ordl_n4 /dev/rordl253 1391M &
addfile.sh ordl_n4 /dev/rordl254 1391M &
addfile.sh ordl_n4 /dev/rordl255 1391M &
addfile.sh ordl_n4 /dev/rordl256 1391M &
addfile.sh ordl_n4 /dev/rordl257 1391M &
addfile.sh ordl_n4 /dev/rordl258 1391M &
addfile.sh ordl_n4 /dev/rordl259 1391M &
addfile.sh ordl_n4 /dev/rordl260 1391M &

#wait

addfile.sh ordl_n4 /dev/rordl261 1391M &
addfile.sh ordl_n4 /dev/rordl262 1391M &
addfile.sh ordl_n4 /dev/rordl263 1391M &
addfile.sh ordl_n4 /dev/rordl264 1391M &
addfile.sh ordl_n4 /dev/rordl265 1391M &
addfile.sh ordl_n4 /dev/rordl266 1391M &
addfile.sh ordl_n4 /dev/rordl267 1391M &
addfile.sh ordl_n4 /dev/rordl268 1391M &
addfile.sh ordl_n4 /dev/rordl269 1391M &
addfile.sh ordl_n4 /dev/rordl270 1391M &
addfile.sh ordl_n4 /dev/rordl271 1391M &
addfile.sh ordl_n4 /dev/rordl272 1391M &
addfile.sh ordl_n4 /dev/rordl273 1391M &
addfile.sh ordl_n4 /dev/rordl274 1391M &
addfile.sh ordl_n4 /dev/rordl275 1391M &
addfile.sh ordl_n4 /dev/rordl276 1391M &
addfile.sh ordl_n4 /dev/rordl277 1391M &
addfile.sh ordl_n4 /dev/rordl278 1391M &
addfile.sh ordl_n4 /dev/rordl279 1391M &
addfile.sh ordl_n4 /dev/rordl280 1391M &

wait

addfile.sh ordl_n4 /dev/rordl281 1391M &
addfile.sh ordl_n4 /dev/rordl282 1391M &
addfile.sh ordl_n4 /dev/rordl283 1391M &
addfile.sh ordl_n4 /dev/rordl284 1391M &
addfile.sh ordl_n4 /dev/rordl285 1391M &
addfile.sh ordl_n4 /dev/rordl286 1391M &
addfile.sh ordl_n4 /dev/rordl287 1391M &
addfile.sh ordl_n4 /dev/rordl288 1391M &
addfile.sh ordl_n4 /dev/rordl289 1391M &
addfile.sh ordl_n4 /dev/rordl290 1391M &
addfile.sh ordl_n4 /dev/rordl291 1391M &
addfile.sh ordl_n4 /dev/rordl292 1391M &
addfile.sh ordl_n4 /dev/rordl293 1391M &
addfile.sh ordl_n4 /dev/rordl294 1391M &
addfile.sh ordl_n4 /dev/rordl295 1391M &
addfile.sh ordl_n4 /dev/rordl296 1391M &
addfile.sh ordl_n4 /dev/rordl297 1391M &
addfile.sh ordl_n4 /dev/rordl298 1391M &
addfile.sh ordl_n4 /dev/rordl299 1391M &
addfile.sh ordl_n4 /dev/rordl300 1391M &

#wait

addfile.sh ordl_n4 /dev/rordl301 1391M &
addfile.sh ordl_n4 /dev/rordl302 1391M &
addfile.sh ordl_n4 /dev/rordl303 1391M &
addfile.sh ordl_n4 /dev/rordl304 1391M &
addfile.sh ordl_n4 /dev/rordl305 1391M &
addfile.sh ordl_n4 /dev/rordl306 1391M &
addfile.sh ordl_n4 /dev/rordl307 1391M &
addfile.sh ordl_n4 /dev/rordl308 1391M &
addfile.sh ordl_n4 /dev/rordl309 1391M &
addfile.sh ordl_n4 /dev/rordl310 1391M &
addfile.sh ordl_n4 /dev/rordl311 1391M &
addfile.sh ordl_n4 /dev/rordl312 1391M &
addfile.sh ordl_n4 /dev/rordl313 1391M &
addfile.sh ordl_n4 /dev/rordl314 1391M &
addfile.sh ordl_n4 /dev/rordl315 1391M &
addfile.sh ordl_n4 /dev/rordl316 1391M &
addfile.sh ordl_n4 /dev/rordl317 1391M &
addfile.sh ordl_n4 /dev/rordl318 1391M &
addfile.sh ordl_n4 /dev/rordl319 1391M &
addfile.sh ordl_n4 /dev/rordl320 1391M &

wait

echo "Done: cr_n4_ordl_ts.sh - `date`"


################################################################################
#
#  FILE:  cr_n4_ords_ts.sh
#
#       Createords tablespace on node4
#
################################################################################
. ~/.profile
set -x

svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ords_n4 datafile '/dev/rords031' size 607M reuse;
      exit;
!

addfile.sh ords_n4  /dev/rords032 607M &
addfile.sh ords_n4  /dev/rords033 607M &
addfile.sh ords_n4  /dev/rords034 607M &
addfile.sh ords_n4  /dev/rords035 607M &
addfile.sh ords_n4  /dev/rords036 607M &
addfile.sh ords_n4  /dev/rords037 607M &
addfile.sh ords_n4  /dev/rords038 607M &
addfile.sh ords_n4  /dev/rords039 607M &
addfile.sh ords_n4  /dev/rords040 607M &

wait
date
echo "Done: cr_n4_ords_ts.sh"


################################################################################
#
#  FILE:  cr_n4_stoc_ts.sh
#
#       Createstoc tablespace on node4
#
################################################################################

. ~/.profile


addfile.sh stoc /dev/rstoc667 371M &
addfile.sh stoc /dev/rstoc668 371M &
addfile.sh stoc /dev/rstoc669 371M &
addfile.sh stoc /dev/rstoc670 371M &
addfile.sh stoc /dev/rstoc671 371M &
addfile.sh stoc /dev/rstoc672 371M &
addfile.sh stoc /dev/rstoc673 371M &
addfile.sh stoc /dev/rstoc674 371M &
addfile.sh stoc /dev/rstoc675 371M &
addfile.sh stoc /dev/rstoc676 371M &
addfile.sh stoc /dev/rstoc677 371M &
addfile.sh stoc /dev/rstoc678 371M &
addfile.sh stoc /dev/rstoc679 371M &
addfile.sh stoc /dev/rstoc680 371M &
addfile.sh stoc /dev/rstoc681 371M &
addfile.sh stoc /dev/rstoc682 371M &
addfile.sh stoc /dev/rstoc683 371M &
addfile.sh stoc /dev/rstoc684 371M &
addfile.sh stoc /dev/rstoc685 371M &
addfile.sh stoc /dev/rstoc686 371M &
addfile.sh stoc /dev/rstoc687 371M &
addfile.sh stoc /dev/rstoc688 371M &

addfile.sh stoc /dev/rstoc689 371M &
addfile.sh stoc /dev/rstoc690 371M &
addfile.sh stoc /dev/rstoc691 371M &
addfile.sh stoc /dev/rstoc692 371M &
addfile.sh stoc /dev/rstoc693 371M &
addfile.sh stoc /dev/rstoc694 371M &
addfile.sh stoc /dev/rstoc695 371M &
addfile.sh stoc /dev/rstoc696 371M &
addfile.sh stoc /dev/rstoc697 371M &
addfile.sh stoc /dev/rstoc698 371M &
addfile.sh stoc /dev/rstoc699 371M &
addfile.sh stoc /dev/rstoc700 371M &
addfile.sh stoc /dev/rstoc701 371M &
addfile.sh stoc /dev/rstoc702 371M &
addfile.sh stoc /dev/rstoc703 371M &
addfile.sh stoc /dev/rstoc704 371M &
addfile.sh stoc /dev/rstoc705 371M &
addfile.sh stoc /dev/rstoc706 371M &
addfile.sh stoc /dev/rstoc707 371M &
addfile.sh stoc /dev/rstoc708 371M &
addfile.sh stoc /dev/rstoc709 371M &
addfile.sh stoc /dev/rstoc710 371M &

addfile.sh stoc /dev/rstoc711 371M &
addfile.sh stoc /dev/rstoc712 371M &
addfile.sh stoc /dev/rstoc713 371M &
addfile.sh stoc /dev/rstoc714 371M &
addfile.sh stoc /dev/rstoc715 371M &
addfile.sh stoc /dev/rstoc716 371M &
addfile.sh stoc /dev/rstoc717 371M &
addfile.sh stoc /dev/rstoc718 371M &
addfile.sh stoc /dev/rstoc719 371M &
addfile.sh stoc /dev/rstoc720 371M &
addfile.sh stoc /dev/rstoc721 371M &
addfile.sh stoc /dev/rstoc722 371M &
addfile.sh stoc /dev/rstoc723 371M &
addfile.sh stoc /dev/rstoc724 371M &
addfile.sh stoc /dev/rstoc725 371M &
addfile.sh stoc /dev/rstoc726 371M &
addfile.sh stoc /dev/rstoc727 371M &
```

```
addfile.sh stoc /dev/rstoc728 371M &
addfile.sh stoc /dev/rstoc729 371M &
addfile.sh stoc /dev/rstoc730 371M &
addfile.sh stoc /dev/rstoc731 371M &
addfile.sh stoc /dev/rstoc732 371M &

addfile.sh stoc /dev/rstoc733 371M &
addfile.sh stoc /dev/rstoc734 371M &
addfile.sh stoc /dev/rstoc735 371M &
addfile.sh stoc /dev/rstoc736 371M &
addfile.sh stoc /dev/rstoc737 371M &
addfile.sh stoc /dev/rstoc738 371M &
addfile.sh stoc /dev/rstoc739 371M &
addfile.sh stoc /dev/rstoc740 371M &
addfile.sh stoc /dev/rstoc741 371M &
addfile.sh stoc /dev/rstoc742 371M &
addfile.sh stoc /dev/rstoc743 371M &
addfile.sh stoc /dev/rstoc744 371M &
addfile.sh stoc /dev/rstoc745 371M &
addfile.sh stoc /dev/rstoc746 371M &
addfile.sh stoc /dev/rstoc747 371M &
addfile.sh stoc /dev/rstoc748 371M &
addfile.sh stoc /dev/rstoc749 371M &
addfile.sh stoc /dev/rstoc750 371M &
addfile.sh stoc /dev/rstoc751 371M &
addfile.sh stoc /dev/rstoc752 371M &
addfile.sh stoc /dev/rstoc753 371M &
addfile.sh stoc /dev/rstoc754 371M &

addfile.sh stoc /dev/rstoc755 371M &
addfile.sh stoc /dev/rstoc756 371M &
addfile.sh stoc /dev/rstoc757 371M &
addfile.sh stoc /dev/rstoc758 371M &
addfile.sh stoc /dev/rstoc759 371M &
addfile.sh stoc /dev/rstoc760 371M &
addfile.sh stoc /dev/rstoc761 371M &
addfile.sh stoc /dev/rstoc762 371M &
addfile.sh stoc /dev/rstoc763 371M &
addfile.sh stoc /dev/rstoc764 371M &
addfile.sh stoc /dev/rstoc765 371M &
addfile.sh stoc /dev/rstoc766 371M &
addfile.sh stoc /dev/rstoc767 371M &
addfile.sh stoc /dev/rstoc768 371M &
addfile.sh stoc /dev/rstoc769 371M &
addfile.sh stoc /dev/rstoc770 371M &
addfile.sh stoc /dev/rstoc771 371M &
addfile.sh stoc /dev/rstoc772 371M &
addfile.sh stoc /dev/rstoc773 371M &
addfile.sh stoc /dev/rstoc774 371M &
addfile.sh stoc /dev/rstoc775 371M &
addfile.sh stoc /dev/rstoc776 371M &

addfile.sh stoc /dev/rstoc777 371M &
addfile.sh stoc /dev/rstoc778 371M &
addfile.sh stoc /dev/rstoc779 371M &
addfile.sh stoc /dev/rstoc780 371M &
addfile.sh stoc /dev/rstoc781 371M &
addfile.sh stoc /dev/rstoc782 371M &
addfile.sh stoc /dev/rstoc783 371M &
addfile.sh stoc /dev/rstoc784 371M &
addfile.sh stoc /dev/rstoc785 371M &
addfile.sh stoc /dev/rstoc786 371M &
addfile.sh stoc /dev/rstoc787 371M &
addfile.sh stoc /dev/rstoc788 371M &
addfile.sh stoc /dev/rstoc789 371M &
addfile.sh stoc /dev/rstoc790 371M &
addfile.sh stoc /dev/rstoc791 371M &
addfile.sh stoc /dev/rstoc792 371M &
addfile.sh stoc /dev/rstoc793 371M &
addfile.sh stoc /dev/rstoc794 371M &
addfile.sh stoc /dev/rstoc795 371M &
addfile.sh stoc /dev/rstoc796 371M &
addfile.sh stoc /dev/rstoc797 371M &
addfile.sh stoc /dev/rstoc798 371M &

addfile.sh stoc /dev/rstoc799 371M &
addfile.sh stoc /dev/rstoc800 371M &
addfile.sh stoc /dev/rstoc801 371M &
addfile.sh stoc /dev/rstoc802 371M &
addfile.sh stoc /dev/rstoc803 371M &
addfile.sh stoc /dev/rstoc804 371M &
addfile.sh stoc /dev/rstoc805 371M &
addfile.sh stoc /dev/rstoc806 371M &
addfile.sh stoc /dev/rstoc807 371M &
addfile.sh stoc /dev/rstoc808 371M &
addfile.sh stoc /dev/rstoc809 371M &
addfile.sh stoc /dev/rstoc810 371M &
addfile.sh stoc /dev/rstoc811 371M &
addfile.sh stoc /dev/rstoc812 371M &
addfile.sh stoc /dev/rstoc813 371M &
addfile.sh stoc /dev/rstoc814 371M &
addfile.sh stoc /dev/rstoc815 371M &
addfile.sh stoc /dev/rstoc816 371M &
addfile.sh stoc /dev/rstoc817 371M &
addfile.sh stoc /dev/rstoc818 371M &
addfile.sh stoc /dev/rstoc819 371M &
addfile.sh stoc /dev/rstoc820 371M &

addfile.sh stoc /dev/rstoc821 371M &
addfile.sh stoc /dev/rstoc822 371M &
addfile.sh stoc /dev/rstoc823 371M &
addfile.sh stoc /dev/rstoc824 371M &
addfile.sh stoc /dev/rstoc825 371M &
addfile.sh stoc /dev/rstoc826 371M &
addfile.sh stoc /dev/rstoc827 371M &
addfile.sh stoc /dev/rstoc828 371M &
addfile.sh stoc /dev/rstoc829 371M &
addfile.sh stoc /dev/rstoc830 371M &
addfile.sh stoc /dev/rstoc831 371M &
addfile.sh stoc /dev/rstoc832 371M &
addfile.sh stoc /dev/rstoc833 371M &
addfile.sh stoc /dev/rstoc834 371M &
addfile.sh stoc /dev/rstoc835 371M &
addfile.sh stoc /dev/rstoc836 371M &
addfile.sh stoc /dev/rstoc837 371M &
addfile.sh stoc /dev/rstoc838 371M &
addfile.sh stoc /dev/rstoc839 371M &
addfile.sh stoc /dev/rstoc840 371M &
addfile.sh stoc /dev/rstoc841 371M &
addfile.sh stoc /dev/rstoc842 371M &

addfile.sh stoc /dev/rstoc843 371M &
addfile.sh stoc /dev/rstoc844 371M &
addfile.sh stoc /dev/rstoc845 371M &
addfile.sh stoc /dev/rstoc846 371M &
addfile.sh stoc /dev/rstoc847 371M &
addfile.sh stoc /dev/rstoc848 371M &
addfile.sh stoc /dev/rstoc849 371M &
addfile.sh stoc /dev/rstoc850 371M &
addfile.sh stoc /dev/rstoc851 371M &
addfile.sh stoc /dev/rstoc852 371M &
addfile.sh stoc /dev/rstoc853 371M &
addfile.sh stoc /dev/rstoc854 371M &
addfile.sh stoc /dev/rstoc855 371M &
addfile.sh stoc /dev/rstoc856 371M &
addfile.sh stoc /dev/rstoc857 371M &
addfile.sh stoc /dev/rstoc858 371M &
addfile.sh stoc /dev/rstoc859 371M &
addfile.sh stoc /dev/rstoc860 371M &
addfile.sh stoc /dev/rstoc861 371M &
addfile.sh stoc /dev/rstoc862 371M &
addfile.sh stoc /dev/rstoc863 371M &
addfile.sh stoc /dev/rstoc864 371M &

addfile.sh stoc /dev/rstoc865 371M &
addfile.sh stoc /dev/rstoc866 371M &
addfile.sh stoc /dev/rstoc867 371M &
addfile.sh stoc /dev/rstoc868 371M &
addfile.sh stoc /dev/rstoc869 371M &
addfile.sh stoc /dev/rstoc870 371M &
addfile.sh stoc /dev/rstoc871 371M &
addfile.sh stoc /dev/rstoc872 371M &
addfile.sh stoc /dev/rstoc873 371M &
addfile.sh stoc /dev/rstoc874 371M &
addfile.sh stoc /dev/rstoc875 371M &
addfile.sh stoc /dev/rstoc876 371M &
addfile.sh stoc /dev/rstoc877 371M &
addfile.sh stoc /dev/rstoc878 371M &
addfile.sh stoc /dev/rstoc879 371M &
addfile.sh stoc /dev/rstoc880 371M &
addfile.sh stoc /dev/rstoc881 371M &
addfile.sh stoc /dev/rstoc882 371M &
addfile.sh stoc /dev/rstoc883 371M &
addfile.sh stoc /dev/rstoc884 371M &
addfile.sh stoc /dev/rstoc885 371M &
addfile.sh stoc /dev/rstoc886 371M &
addfile.sh stoc /dev/rstoc887 371M &
addfile.sh stoc /dev/rstoc888 371M &
wait


################################################################################
#
#  FILE:  cr_n4_temp_ts.sh
#
#       Create temptablespace form node4
#
################################################################################

. ~/.profile

addfile.sh temp /dev/rtemp064 1999M &
addfile.sh temp /dev/rtemp065 1999M &
addfile.sh temp /dev/rtemp066 1999M &
addfile.sh temp /dev/rtemp067 1999M &
addfile.sh temp /dev/rtemp068 1999M &
addfile.sh temp /dev/rtemp069 1999M &
addfile.sh temp /dev/rtemp070 1999M &
addfile.sh temp /dev/rtemp071 1999M &
addfile.sh temp /dev/rtemp072 1999M &
addfile.sh temp /dev/rtemp073 1999M &
addfile.sh temp /dev/rtemp074 1999M &
addfile.sh temp /dev/rtemp075 1999M &
addfile.sh temp /dev/rtemp076 1999M &
addfile.sh temp /dev/rtemp077 1999M &
addfile.sh temp /dev/rtemp078 1999M &
addfile.sh temp /dev/rtemp079 1999M &
addfile.sh temp /dev/rtemp080 1999M &
```

```
addfile.sh temp /dev/rtemp081 1999M &
addfile.sh temp /dev/rtemp082 1999M &
addfile.sh temp /dev/rtemp083 1999M &
addfile.sh temp /dev/rtemp084 1999M &


wait


#################################################################################
#
#  FILE:  cr_n5_cust_ts.sh
#
#        Createcust tablespace on node5
#
#################################################################################


. ~/.profile

addfile.sh cust /dev/rcust353 771M &
addfile.sh cust /dev/rcust354 771M &
addfile.sh cust /dev/rcust355 771M &
addfile.sh cust /dev/rcust356 771M &
addfile.sh cust /dev/rcust357 771M &
addfile.sh cust /dev/rcust358 771M &
addfile.sh cust /dev/rcust359 771M &
addfile.sh cust /dev/rcust360 771M &
addfile.sh cust /dev/rcust361 771M &
addfile.sh cust /dev/rcust362 771M &
addfile.sh cust /dev/rcust363 771M &
addfile.sh cust /dev/rcust364 771M &
addfile.sh cust /dev/rcust365 771M &
addfile.sh cust /dev/rcust366 771M &
addfile.sh cust /dev/rcust367 771M &
addfile.sh cust /dev/rcust368 771M &
addfile.sh cust /dev/rcust369 771M &
addfile.sh cust /dev/rcust370 771M &
addfile.sh cust /dev/rcust371 771M &
addfile.sh cust /dev/rcust372 771M &
addfile.sh cust /dev/rcust373 771M &
addfile.sh cust /dev/rcust374 771M &

wait

addfile.sh cust /dev/rcust375 771M &
addfile.sh cust /dev/rcust376 771M &
addfile.sh cust /dev/rcust377 771M &
addfile.sh cust /dev/rcust378 771M &
addfile.sh cust /dev/rcust379 771M &
addfile.sh cust /dev/rcust380 771M &
addfile.sh cust /dev/rcust381 771M &
addfile.sh cust /dev/rcust382 771M &
addfile.sh cust /dev/rcust383 771M &
addfile.sh cust /dev/rcust384 771M &
addfile.sh cust /dev/rcust385 771M &
addfile.sh cust /dev/rcust386 771M &
addfile.sh cust /dev/rcust387 771M &
addfile.sh cust /dev/rcust388 771M &
addfile.sh cust /dev/rcust389 771M &
addfile.sh cust /dev/rcust390 771M &
addfile.sh cust /dev/rcust391 771M &
addfile.sh cust /dev/rcust392 771M &
addfile.sh cust /dev/rcust393 771M &
addfile.sh cust /dev/rcust394 771M &
addfile.sh cust /dev/rcust395 771M &
addfile.sh cust /dev/rcust396 771M &

wait

addfile.sh cust /dev/rcust397 771M &
addfile.sh cust /dev/rcust398 771M &
addfile.sh cust /dev/rcust399 771M &
addfile.sh cust /dev/rcust400 771M &
addfile.sh cust /dev/rcust401 771M &
addfile.sh cust /dev/rcust402 771M &
addfile.sh cust /dev/rcust403 771M &
addfile.sh cust /dev/rcust404 771M &
addfile.sh cust /dev/rcust405 771M &
addfile.sh cust /dev/rcust406 771M &
addfile.sh cust /dev/rcust407 771M &
addfile.sh cust /dev/rcust408 771M &
addfile.sh cust /dev/rcust409 771M &
addfile.sh cust /dev/rcust410 771M &
addfile.sh cust /dev/rcust411 771M &
addfile.sh cust /dev/rcust412 771M &
addfile.sh cust /dev/rcust413 771M &
addfile.sh cust /dev/rcust414 771M &
addfile.sh cust /dev/rcust415 771M &
addfile.sh cust /dev/rcust416 771M &
addfile.sh cust /dev/rcust417 771M &
addfile.sh cust /dev/rcust418 771M &

wait

addfile.sh cust /dev/rcust419 771M &
addfile.sh cust /dev/rcust420 771M &
addfile.sh cust /dev/rcust421 771M &
addfile.sh cust /dev/rcust422 771M &
addfile.sh cust /dev/rcust423 771M &
addfile.sh cust /dev/rcust424 771M &
addfile.sh cust /dev/rcust425 771M &
addfile.sh cust /dev/rcust426 771M &
addfile.sh cust /dev/rcust427 771M &
addfile.sh cust /dev/rcust428 771M &
addfile.sh cust /dev/rcust429 771M &
addfile.sh cust /dev/rcust430 771M &
addfile.sh cust /dev/rcust431 771M &
addfile.sh cust /dev/rcust432 771M &
addfile.sh cust /dev/rcust433 771M &
addfile.sh cust /dev/rcust434 771M &
addfile.sh cust /dev/rcust435 771M &
addfile.sh cust /dev/rcust436 771M &
addfile.sh cust /dev/rcust437 771M &
addfile.sh cust /dev/rcust438 771M &
addfile.sh cust /dev/rcust439 771M &
addfile.sh cust /dev/rcust440 771M &

wait


#################################################################################
#
#  FILE:  cr_n5_iord_ts.sh
#
#        Createior1 and ior2 tablespaces on node5
#
#################################################################################

. ~/.profile
set -x
svrmgrl <<!
      set echo on
      connect system/manager
      create tablespace ior1_n5 datafile '/dev/rior1041' size 607M reuse;
      exit;
!

addfile.sh ior1_n5  /dev/rior1042 607M &
addfile.sh ior1_n5  /dev/rior1043 607M &
addfile.sh ior1_n5  /dev/rior1044 607M &
addfile.sh ior1_n5  /dev/rior1045 607M &
addfile.sh ior1_n5  /dev/rior1046 607M &
addfile.sh ior1_n5  /dev/rior1047 607M &
addfile.sh ior1_n5  /dev/rior1048 607M &
addfile.sh ior1_n5  /dev/rior1049 607M &
addfile.sh ior1_n5  /dev/rior1050 607M &

wait
date
echo "Done: cr_n5_iord1_ts.sh"
#################################################################################
#
#  FILE:  cr_n5_iord_ts.sh
#
#        Create ior1 and ior2 tablespaces on node5
#
#################################################################################

. ~/.profile
set -x

svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace ior2_n5 datafile   '/dev/rior2081'  size 511M reuse;
  exit;
!
addfile.sh ior2_n5  /dev/rior2082 511M &
addfile.sh ior2_n5  /dev/rior2083 511M &
addfile.sh ior2_n5  /dev/rior2084 511M &
addfile.sh ior2_n5  /dev/rior2085 511M &
addfile.sh ior2_n5  /dev/rior2086 511M &
addfile.sh ior2_n5  /dev/rior2087 511M &
addfile.sh ior2_n5  /dev/rior2088 511M &
addfile.sh ior2_n5  /dev/rior2089 511M &
addfile.sh ior2_n5  /dev/rior2090 511M &
addfile.sh ior2_n5  /dev/rior2091 511M &
addfile.sh ior2_n5  /dev/rior2092 511M &
addfile.sh ior2_n5  /dev/rior2093 511M &
addfile.sh ior2_n5  /dev/rior2094 511M &
addfile.sh ior2_n5  /dev/rior2095 511M &
addfile.sh ior2_n5  /dev/rior2096 511M &
addfile.sh ior2_n5  /dev/rior2097 511M &
addfile.sh ior2_n5  /dev/rior2098 511M &
addfile.sh ior2_n5  /dev/rior2099 511M &
addfile.sh ior2_n5  /dev/rior2100 511M &
wait
date
echo "Done: cr_n5_iord2_ts.sh"

#################################################################################
#
#  FILE:  cr_n5_iorl_ts.sh
#
#        Create iorl tablespace  on node5
#
#################################################################################
```

```
. ~/.profile

svrmgrl <<!
        set echo on
        connect system/manager

        create tablespace iorl_n5_1 datafile   '/dev/riorl089'  size 2127M reuse;
        create tablespace iorl_n5_2 datafile   '/dev/riorl100'  size 2127M reuse;

        exit;
!

addfile.sh iorl_n5_1  /dev/riorl090 2127M &
addfile.sh iorl_n5_1  /dev/riorl091 2127M &
addfile.sh iorl_n5_1  /dev/riorl092 2127M &
addfile.sh iorl_n5_1  /dev/riorl093 2127M &
addfile.sh iorl_n5_1  /dev/riorl094 2127M &
addfile.sh iorl_n5_1  /dev/riorl095 2127M &
addfile.sh iorl_n5_1  /dev/riorl096 2127M &
addfile.sh iorl_n5_1  /dev/riorl097 2127M &
addfile.sh iorl_n5_1  /dev/riorl098 2127M &
addfile.sh iorl_n5_1  /dev/riorl099 2127M &

wait

addfile.sh iorl_n5_2  /dev/riorl101 2127M &
addfile.sh iorl_n5_2  /dev/riorl102 2127M &
addfile.sh iorl_n5_2  /dev/riorl103 2127M &
addfile.sh iorl_n5_2  /dev/riorl104 2127M &
addfile.sh iorl_n5_2  /dev/riorl105 2127M &
addfile.sh iorl_n5_2  /dev/riorl106 2127M &
addfile.sh iorl_n5_2  /dev/riorl107 2127M &
addfile.sh iorl_n5_2  /dev/riorl108 2127M &
addfile.sh iorl_n5_2  /dev/riorl109 2127M &
addfile.sh iorl_n5_2  /dev/riorl110 2127M &

wait


################################################################################
#
#   FILE:  cr_n5_istk_ts.sh
#
#         Create istk tablespace on node5
#
################################################################################

. ~/.profile

addfile.sh istk  /dev/ristk017  1126M &
addfile.sh istk  /dev/ristk018  1126M &
addfile.sh istk  /dev/ristk019  1126M &
addfile.sh istk  /dev/ristk020  1126M &

wait


################################################################################
#
#   FILE:  cr_n5_ordl_ts.sh
#
#         Create ORDL tablespaces on node5
#
################################################################################

. ~/.profile
set -x

echo "starting Node 5 - `date`"
svrmgrl <<!
        set echo on
        connect system/manager
        create tablespace ordl_n5 datafile '/dev/rordl321' size 1391M reuse;
        exit;
!

addfile.sh ordl_n5 /dev/rordl322 1391M &
addfile.sh ordl_n5 /dev/rordl323 1391M &
addfile.sh ordl_n5 /dev/rordl324 1391M &
addfile.sh ordl_n5 /dev/rordl325 1391M &
addfile.sh ordl_n5 /dev/rordl326 1391M &
addfile.sh ordl_n5 /dev/rordl327 1391M &
addfile.sh ordl_n5 /dev/rordl328 1391M &
addfile.sh ordl_n5 /dev/rordl329 1391M &
addfile.sh ordl_n5 /dev/rordl330 1391M &
addfile.sh ordl_n5 /dev/rordl331 1391M &
addfile.sh ordl_n5 /dev/rordl332 1391M &
addfile.sh ordl_n5 /dev/rordl333 1391M &
addfile.sh ordl_n5 /dev/rordl334 1391M &
addfile.sh ordl_n5 /dev/rordl335 1391M &
addfile.sh ordl_n5 /dev/rordl336 1391M &
addfile.sh ordl_n5 /dev/rordl337 1391M &
addfile.sh ordl_n5 /dev/rordl338 1391M &
addfile.sh ordl_n5 /dev/rordl339 1391M &
addfile.sh ordl_n5 /dev/rordl340 1391M &

#wait
addfile.sh ordl_n5 /dev/rordl341 1391M &
addfile.sh ordl_n5 /dev/rordl342 1391M &
addfile.sh ordl_n5 /dev/rordl343 1391M &
addfile.sh ordl_n5 /dev/rordl344 1391M &
addfile.sh ordl_n5 /dev/rordl345 1391M &
addfile.sh ordl_n5 /dev/rordl346 1391M &
addfile.sh ordl_n5 /dev/rordl347 1391M &
addfile.sh ordl_n5 /dev/rordl348 1391M &
addfile.sh ordl_n5 /dev/rordl349 1391M &
addfile.sh ordl_n5 /dev/rordl350 1391M &
addfile.sh ordl_n5 /dev/rordl351 1391M &
addfile.sh ordl_n5 /dev/rordl352 1391M &
addfile.sh ordl_n5 /dev/rordl353 1391M &
addfile.sh ordl_n5 /dev/rordl354 1391M &
addfile.sh ordl_n5 /dev/rordl355 1391M &
addfile.sh ordl_n5 /dev/rordl356 1391M &
addfile.sh ordl_n5 /dev/rordl357 1391M &
addfile.sh ordl_n5 /dev/rordl358 1391M &
addfile.sh ordl_n5 /dev/rordl359 1391M &
addfile.sh ordl_n5 /dev/rordl360 1391M &

wait

addfile.sh ordl_n5 /dev/rordl361 1391M &
addfile.sh ordl_n5 /dev/rordl362 1391M &
addfile.sh ordl_n5 /dev/rordl363 1391M &
addfile.sh ordl_n5 /dev/rordl364 1391M &
addfile.sh ordl_n5 /dev/rordl365 1391M &
addfile.sh ordl_n5 /dev/rordl366 1391M &
addfile.sh ordl_n5 /dev/rordl367 1391M &
addfile.sh ordl_n5 /dev/rordl368 1391M &
addfile.sh ordl_n5 /dev/rordl369 1391M &
addfile.sh ordl_n5 /dev/rordl370 1391M &
addfile.sh ordl_n5 /dev/rordl371 1391M &
addfile.sh ordl_n5 /dev/rordl372 1391M &
addfile.sh ordl_n5 /dev/rordl373 1391M &
addfile.sh ordl_n5 /dev/rordl374 1391M &
addfile.sh ordl_n5 /dev/rordl375 1391M &
addfile.sh ordl_n5 /dev/rordl376 1391M &
addfile.sh ordl_n5 /dev/rordl377 1391M &
addfile.sh ordl_n5 /dev/rordl378 1391M &
addfile.sh ordl_n5 /dev/rordl379 1391M &
addfile.sh ordl_n5 /dev/rordl380 1391M &

#wait

addfile.sh ordl_n5 /dev/rordl381 1391M &
addfile.sh ordl_n5 /dev/rordl382 1391M &
addfile.sh ordl_n5 /dev/rordl383 1391M &
addfile.sh ordl_n5 /dev/rordl384 1391M &
addfile.sh ordl_n5 /dev/rordl385 1391M &
addfile.sh ordl_n5 /dev/rordl386 1391M &
addfile.sh ordl_n5 /dev/rordl387 1391M &
addfile.sh ordl_n5 /dev/rordl388 1391M &
addfile.sh ordl_n5 /dev/rordl389 1391M &
addfile.sh ordl_n5 /dev/rordl390 1391M &
addfile.sh ordl_n5 /dev/rordl391 1391M &
addfile.sh ordl_n5 /dev/rordl392 1391M &
addfile.sh ordl_n5 /dev/rordl393 1391M &
addfile.sh ordl_n5 /dev/rordl394 1391M &
addfile.sh ordl_n5 /dev/rordl395 1391M &
addfile.sh ordl_n5 /dev/rordl396 1391M &
addfile.sh ordl_n5 /dev/rordl397 1391M &
addfile.sh ordl_n5 /dev/rordl398 1391M &
addfile.sh ordl_n5 /dev/rordl399 1391M &
addfile.sh ordl_n5 /dev/rordl400 1391M &

wait

echo "Done: cr_n5_ordl_ts.sh - `date`"


################################################################################
#
#   FILE:  cr_n5_ords_ts.sh
#
#         Create ords tablespace on node5
#
################################################################################

. ~/.profile
set -x

svrmgrl <<!
        set echo on
        connect system/manager
        create tablespace ords_n5 datafile '/dev/rords041' size 607M reuse;
        exit;
!
addfile.sh ords_n5  /dev/rords042  607M &
addfile.sh ords_n5  /dev/rords043  607M &
addfile.sh ords_n5  /dev/rords044  607M &
addfile.sh ords_n5  /dev/rords045  607M &
addfile.sh ords_n5  /dev/rords046  607M &
addfile.sh ords_n5  /dev/rords047  607M &
addfile.sh ords_n5  /dev/rords048  607M &
addfile.sh ords_n5  /dev/rords049  607M &
addfile.sh ords_n5  /dev/rords050  607M &
```

```
wait
date
echo "Done: cr_n5_ords_ts.sh"

#############################################################################
#
#   FILE:  cr_n5_stoc_ts.sh
#
#         Createstoc tablespace on node5
#
#############################################################################

. ~/.profile

addfile.sh stoc /dev/rstoc889 742M &
addfile.sh stoc /dev/rstoc890 742M &
addfile.sh stoc /dev/rstoc891 742M &
addfile.sh stoc /dev/rstoc892 742M &
addfile.sh stoc /dev/rstoc893 742M &
addfile.sh stoc /dev/rstoc894 742M &
addfile.sh stoc /dev/rstoc895 742M &
addfile.sh stoc /dev/rstoc896 742M &
addfile.sh stoc /dev/rstoc897 742M &
addfile.sh stoc /dev/rstoc898 742M &
addfile.sh stoc /dev/rstoc899 742M &
addfile.sh stoc /dev/rstoc900 742M &
addfile.sh stoc /dev/rstoc901 742M &
addfile.sh stoc /dev/rstoc902 742M &
addfile.sh stoc /dev/rstoc903 742M &
addfile.sh stoc /dev/rstoc904 742M &
addfile.sh stoc /dev/rstoc905 742M &
addfile.sh stoc /dev/rstoc906 742M &
addfile.sh stoc /dev/rstoc907 742M &
addfile.sh stoc /dev/rstoc908 742M &
addfile.sh stoc /dev/rstoc909 742M &
addfile.sh stoc /dev/rstoc910 742M &

addfile.sh stoc /dev/rstoc911 742M &
addfile.sh stoc /dev/rstoc912 742M &
addfile.sh stoc /dev/rstoc913 742M &
addfile.sh stoc /dev/rstoc914 742M &
addfile.sh stoc /dev/rstoc915 742M &
addfile.sh stoc /dev/rstoc916 742M &
addfile.sh stoc /dev/rstoc917 742M &
addfile.sh stoc /dev/rstoc918 742M &
addfile.sh stoc /dev/rstoc919 742M &
addfile.sh stoc /dev/rstoc920 742M &
addfile.sh stoc /dev/rstoc921 742M &
addfile.sh stoc /dev/rstoc922 742M &
addfile.sh stoc /dev/rstoc923 742M &
addfile.sh stoc /dev/rstoc924 742M &
addfile.sh stoc /dev/rstoc925 742M &
addfile.sh stoc /dev/rstoc926 742M &
addfile.sh stoc /dev/rstoc927 742M &
addfile.sh stoc /dev/rstoc928 742M &
addfile.sh stoc /dev/rstoc929 742M &
addfile.sh stoc /dev/rstoc930 742M &
addfile.sh stoc /dev/rstoc931 742M &
addfile.sh stoc /dev/rstoc932 742M &

addfile.sh stoc /dev/rstoc933 742M &
addfile.sh stoc /dev/rstoc934 742M &
addfile.sh stoc /dev/rstoc935 742M &
addfile.sh stoc /dev/rstoc936 742M &
addfile.sh stoc /dev/rstoc937 742M &
addfile.sh stoc /dev/rstoc938 742M &
addfile.sh stoc /dev/rstoc939 742M &
addfile.sh stoc /dev/rstoc940 742M &
addfile.sh stoc /dev/rstoc941 742M &
addfile.sh stoc /dev/rstoc942 742M &
addfile.sh stoc /dev/rstoc943 742M &
addfile.sh stoc /dev/rstoc944 742M &
addfile.sh stoc /dev/rstoc945 742M &
addfile.sh stoc /dev/rstoc946 742M &
addfile.sh stoc /dev/rstoc947 742M &
addfile.sh stoc /dev/rstoc948 742M &
addfile.sh stoc /dev/rstoc949 742M &
addfile.sh stoc /dev/rstoc950 742M &
addfile.sh stoc /dev/rstoc951 742M &
addfile.sh stoc /dev/rstoc952 742M &
addfile.sh stoc /dev/rstoc953 742M &
addfile.sh stoc /dev/rstoc954 742M &

addfile.sh stoc /dev/rstoc955 742M &
addfile.sh stoc /dev/rstoc956 742M &
addfile.sh stoc /dev/rstoc957 742M &
addfile.sh stoc /dev/rstoc958 742M &
addfile.sh stoc /dev/rstoc959 742M &
addfile.sh stoc /dev/rstoc960 742M &
addfile.sh stoc /dev/rstoc961 742M &
addfile.sh stoc /dev/rstoc962 742M &
addfile.sh stoc /dev/rstoc963 742M &
addfile.sh stoc /dev/rstoc964 742M &
addfile.sh stoc /dev/rstoc965 742M &
addfile.sh stoc /dev/rstoc966 742M &
addfile.sh stoc /dev/rstoc967 742M &
addfile.sh stoc /dev/rstoc968 742M &
addfile.sh stoc /dev/rstoc969 742M &
addfile.sh stoc /dev/rstoc970 742M &
addfile.sh stoc /dev/rstoc971 742M &
addfile.sh stoc /dev/rstoc972 742M &
addfile.sh stoc /dev/rstoc973 742M &
addfile.sh stoc /dev/rstoc974 742M &
addfile.sh stoc /dev/rstoc975 742M &
addfile.sh stoc /dev/rstoc976 742M &

addfile.sh stoc /dev/rstoc977 742M &
addfile.sh stoc /dev/rstoc978 742M &
addfile.sh stoc /dev/rstoc979 742M &
addfile.sh stoc /dev/rstoc980 742M &
addfile.sh stoc /dev/rstoc981 742M &
addfile.sh stoc /dev/rstoc982 742M &
addfile.sh stoc /dev/rstoc983 742M &
addfile.sh stoc /dev/rstoc984 742M &
addfile.sh stoc /dev/rstoc985 742M &
addfile.sh stoc /dev/rstoc986 742M &
addfile.sh stoc /dev/rstoc987 742M &
addfile.sh stoc /dev/rstoc988 742M &
addfile.sh stoc /dev/rstoc989 742M &
addfile.sh stoc /dev/rstoc990 742M &
addfile.sh stoc /dev/rstoc991 742M &
addfile.sh stoc /dev/rstoc992 742M &
addfile.sh stoc /dev/rstoc993 742M &
addfile.sh stoc /dev/rstoc994 742M &
addfile.sh stoc /dev/rstoc995 742M &
addfile.sh stoc /dev/rstoc996 742M &
addfile.sh stoc /dev/rstoc997 742M &
addfile.sh stoc /dev/rstoc998 742M &
addfile.sh stoc /dev/rstoc999 742M &

wait



addfile.sh temp /dev/rtemp087 1999M &
addfile.sh temp /dev/rtemp088 1999M &
addfile.sh temp /dev/rtemp089 1999M &
addfile.sh temp /dev/rtemp090 1999M &
addfile.sh temp /dev/rtemp091 1999M &
addfile.sh temp /dev/rtemp092 1999M &
addfile.sh temp /dev/rtemp093 1999M &
addfile.sh temp /dev/rtemp094 1999M &
addfile.sh temp /dev/rtemp095 1999M &
addfile.sh temp /dev/rtemp096 1999M &
addfile.sh temp /dev/rtemp097 1999M &


addfile.sh temp /dev/rtemp098 1999M &
addfile.sh temp /dev/rtemp099 1999M &
addfile.sh temp /dev/rtemp100 1999M &
addfile.sh temp /dev/rtemp101 1999M &
addfile.sh temp /dev/rtemp102 1999M &
addfile.sh temp /dev/rtemp103 1999M &
addfile.sh temp /dev/rtemp104 1999M &
addfile.sh temp /dev/rtemp105 1999M &


wait
```

Cr_nord_ts.sh

```
. ~/.profile

echo "Starting $0: `date`"
svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace nord_n1_1 datafile   '/dev/rnord001'   size 71M reuse;
  create tablespace nord_n2_1 datafile   '/dev/rnord011'   size 71M reuse;
  create tablespace nord_n3_1 datafile   '/dev/rnord021'   size 71M reuse;
  create tablespace nord_n4_1 datafile   '/dev/rnord031'   size 71M reuse;
  create tablespace nord_n5_1 datafile   '/dev/rnord041'   size 71M reuse;
  exit;
!
  addfile.sh nord_n1_1  /dev/rnord002 71M &
  addfile.sh nord_n1_1  /dev/rnord003 71M &
  addfile.sh nord_n1_1  /dev/rnord004 71M &
  addfile.sh nord_n1_1  /dev/rnord005 71M &
  addfile.sh nord_n1_1  /dev/rnord006 71M &
  addfile.sh nord_n1_1  /dev/rnord007 71M &
  addfile.sh nord_n1_1  /dev/rnord008 71M &
  addfile.sh nord_n1_1  /dev/rnord009 71M &
  addfile.sh nord_n1_1  /dev/rnord010 71M &
  addfile.sh nord_n2_1  /dev/rnord012 71M &
  addfile.sh nord_n2_1  /dev/rnord013 71M &
  addfile.sh nord_n2_1  /dev/rnord014 71M &
  addfile.sh nord_n2_1  /dev/rnord015 71M &
  addfile.sh nord_n2_1  /dev/rnord016 71M &
  addfile.sh nord_n2_1  /dev/rnord017 71M &
  addfile.sh nord_n2_1  /dev/rnord018 71M &
  addfile.sh nord_n2_1  /dev/rnord019 71M &
  addfile.sh nord_n2_1  /dev/rnord020 71M &
  addfile.sh nord_n3_1  /dev/rnord022 71M &
  addfile.sh nord_n3_1  /dev/rnord023 71M &
  addfile.sh nord_n3_1  /dev/rnord024 71M &
  addfile.sh nord_n3_1  /dev/rnord025 71M &
  addfile.sh nord_n3_1  /dev/rnord026 71M &
```

```
  addfile.sh nord_n3_1  /dev/rnord027 71M &
  addfile.sh nord_n3_1  /dev/rnord028 71M &
  addfile.sh nord_n3_1  /dev/rnord029 71M &
  addfile.sh nord_n3_1  /dev/rnord030 71M &
  addfile.sh nord_n4_1  /dev/rnord032 71M &
  addfile.sh nord_n4_1  /dev/rnord033 71M &
  addfile.sh nord_n4_1  /dev/rnord034 71M &
  addfile.sh nord_n4_1  /dev/rnord035 71M &
  addfile.sh nord_n4_1  /dev/rnord036 71M &
  addfile.sh nord_n4_1  /dev/rnord037 71M &
  addfile.sh nord_n4_1  /dev/rnord038 71M &
  addfile.sh nord_n4_1  /dev/rnord039 71M &
  addfile.sh nord_n4_1  /dev/rnord040 71M &
  addfile.sh nord_n5_1  /dev/rnord042 71M &
  addfile.sh nord_n5_1  /dev/rnord043 71M &
  addfile.sh nord_n5_1  /dev/rnord044 71M &
  addfile.sh nord_n5_1  /dev/rnord045 71M &
  addfile.sh nord_n5_1  /dev/rnord046 71M &
  addfile.sh nord_n5_1  /dev/rnord047 71M &
  addfile.sh nord_n5_1  /dev/rnord048 71M &
  addfile.sh nord_n5_1  /dev/rnord049 71M &
  addfile.sh nord_n5_1  /dev/rnord050 71M &
wait
echo "ending $0: `date`"
```

Cr_ordl_ts.sh

```
set -x
date
cr_n1_ordl_ts.sh
cr_n2_ordl_ts.sh
cr_n3_ordl_ts.sh
cr_n4_ordl_ts.sh
cr_n5_ordl_ts.sh
date
echo "Done"
```

cr_ords_ts.sh

```
set -x
date
cr_n1_ords_ts.sh  &
cr_n2_ords_ts.sh  &
cr_n3_ords_ts.sh  &
cr_n4_ords_ts.sh  &
cr_n5_ords_ts.sh  &
wait
date
echo "Done"
```

cr_private_roll.sh

```
#!/bin/sh
#############################################################
#
# FILE:  cr_private_roll.sh
#
#####################################

. ~/.profile

TPCC_BUILD_SQL=$TPCC_SCRIPTS/5s7a/sql

sqlplus system/manager @$TPCC_BUILD_SQL/tpcc_n1_roll
sqlplus system/manager @$TPCC_BUILD_SQL/tpcc_n2_roll
sqlplus system/manager @$TPCC_BUILD_SQL/tpcc_n3_roll
sqlplus system/manager @$TPCC_BUILD_SQL/tpcc_n4_roll
sqlplus system/manager @$TPCC_BUILD_SQL/tpcc_n5_roll

exit
```

Cr_stoc_ts.sh

```
#############################################################
#
# FILE:  cr_stoc_ts.sh
#
#####################################
set -x

. ~/.profile

date

svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace stoc datafile '/dev/rstoc001' size 371M reuse;
  exit;
!

date
    cr_n1_stoc_ts.sh
date
    cr_n2_stoc_ts.sh
date
    cr_n3_stoc_ts.sh
date
    cr_n4_stoc_ts.sh
date
    cr_n5_stoc_ts.sh
date
```

Cr_temp_ts.sh

```
#########################################################################
#
#   FILE:  cr_temp_ts.sh
#
#         Create temp tablespace
#
#########################################################################

svrmgrl <<!
  set echo on
  connect system/manager
  create tablespace temp datafile '/dev/rtemp001' size 1999M reuse;
  exit;
!

cr_n1_temp_ts.sh   &
cr_n2_temp_ts.sh   &
cr_n3_temp_ts.sh   &
cr_n4_temp_ts.sh   &
cr_n5_temp_ts.sh   &

Wait
```

Cr_wdi_ts.sh

```
# tablespace for warehouse district and item
# node and partition suffix

. ~/.profile

svrmgrl <<!
       set echo on
       connect system/manager
       create tablespace ware   datafile '/dev/rware001'   size 79M reuse;
       create tablespace dist   datafile '/dev/ritem002'   size 95M reuse;
       create tablespace items  datafile '/dev/ritem003'   size 95M reuse;
       exit;
!
addfile.sh dist '/dev/ritem004' 95M &

wait
```

Crblog2.sh

```
set -x
svrmgrl <<!
 set echo on
 set termout on
 connect internal
 alter database add logfile thread 2
          GROUP  3('/dev/rlog1i2a', '/dev/rlog1i2b') size 18230M reuse,
          GROUP  4('/dev/rlog2i2a', '/dev/rlog2i2b') size 18230M reuse;
 exit
!
```

Crblog3.sh

```
set -x
svrmgrl <<!
 set echo on
 set termout on
 connect internal
 alter database add logfile thread 3
          GROUP  5('/dev/rlog1i3a', '/dev/rlog1i3b') size 18230M reuse,
          GROUP  6('/dev/rlog2i3a', '/dev/rlog2i3b') size 18230M reuse;
 exit
!
```

Crblog4.sh

```
set -x
svrmgrl <<!
 set echo on
 set termout on
 connect internal
 alter database add logfile thread 4
          GROUP  7 ('/dev/rlog1i4a', '/dev/rlog1i4b') size 18230M reuse,
          GROUP  8 ('/dev/rlog2i4a', '/dev/rlog2i4b') size 18230M reuse;
 exit
!
```

Crblog5.sh

```
set -x
svrmgrl <<!
 set echo on
 set termout on
 connect internal
 alter database add logfile thread 5
          GROUP  9 ('/dev/rlog1i5a', '/dev/rlog1i5b') size 18230M reuse,
          GROUP 10 ('/dev/rlog2i5a', '/dev/rlog2i5b') size 18230M reuse;
 exit
!
```

Dml.sh

```
# $Header: dml.sh 7030100.1 96/05/02 10:22:52 plai Generic<base> $ Copyr (c) 1995 Oracle
#

#+=================================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
#                  OPEN SYSTEMS PERFORMANCE GROUP                 |
#                     All Rights Reserved                         |
#+=================================================================+
# FILENAME
#     dml.sh
# DESCRIPTION
#     Disable table locks for TPC-C tables.
# USAGE
#     dml.sh
#=================================================================*/

sqlplus tpcc/tpcc <<!
   alter table warehouse  disable table lock;
   alter table district   disable table lock;
   alter table customer   disable table lock;
   alter table history    disable table lock;
   alter table item       disable table lock;
   alter table stock      disable table lock;
   alter table orders     disable table lock;
   alter table new_order  disable table lock;
   alter table order_line disable table lock;
   quit;
!
```

Pld_ord.sh

```
set -x
date
pld_ord1.sh
date
pld_ord2.sh
date
pld_ord3.sh
date
pld_ord4.sh
date
pld_ord5.sh
date
echo "Done with pld_ord.sh"
```

Pld_ord1.sh

```
#+=================================================================+
# FILENAME
#      pld_ord1.sh
# DESCRIPTION
#
#=================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts

MULT=11000

PATH=${PATH}:$TPCC_SOURCE
export PATH
export LDIR=/tmp/data
echo $LDIR

set -x

if [ -d ${LDIR} ]
then rm -f ${LDIR}/*
else mkdir ${LDIR}
fi

#-------------------------------------------------------------------
#  Load order and order-line table
#-------------------------------------------------------------------

OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/p1
SW=1
EW=220
INC=1100
for I in 1 2 3 4 5 6 7 8 9 10
do
     echo "Creating pipes ${LDIR}/ordline${I}.dat, ${LDIR}/order${I}.dat, SW=$SW,  EW=$EW,
MULT=$MULT"
     mknod ${LDIR}/ordline${I}.dat p
     mknod ${LDIR}/order${I}.dat p
     tpccload -M $MULT -o $LDIR/ordline${I}.dat -g -b $SW -e $EW > \
       $LDIR/order${I}.dat 2> ${OUTDIR}/order${I}.out&
     sleep 30
     sqlldr tpcc/tpcc control=$TPCC_LOADER/ordline_p${I}.ctl \
          log=${OUTDIR}/ordline${I}.log \
          bad=${OUTDIR}/ordline${I}.bad data=$LDIR/ordline${I}.dat \
          discard=${OUTDIR}/ordline${I}.dsc &
     sqlldr tpcc/tpcc control=$TPCC_LOADER/order_p${I}.ctl \
          log=${OUTDIR}/order${I}.log \
          bad=${OUTDIR}/order${I}.bad data=$LDIR/order${I}.dat \
          discard=${OUTDIR}/order${I}.dsc &
   SW=`expr $SW + $INC`
   EW=`expr $EW + $INC`
done

wait
for I in 1 2 3 4 5 6 7 8 9 10
do
   rm -f $LDIR/order${I}.dat
   rm -f $LDIR/ordline${I}.dat
done
date

echo "Done"
```

Pld_ord2.sh

```
#
#+=================================================================+
# FILENAME
#      pld_ord2.sh
# DESCRIPTION
#
#=================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts

MULT=11000

PATH=${PATH}:$TPCC_SOURCE
export PATH
export LDIR=/tmp/data
echo $LDIR

set -x

if [ -d ${LDIR} ]
then rm -f ${LDIR}/*
else mkdir ${LDIR}
fi

#-------------------------------------------------------------------
#  Load order and order-line table
#-------------------------------------------------------------------

OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/p2
SW=221
EW=440
INC=1100
for I in 1 2 3 4 5 6 7 8 9 10
do
     echo "Creating pipes ${LDIR}/ordline${I}.dat, ${LDIR}/order${I}.dat, SW=$SW, EW=$EW,
MULT=$MULT"
     mknod ${LDIR}/ordline${I}.dat p
     mknod ${LDIR}/order${I}.dat p
     tpccload -M $MULT -o $LDIR/ordline${I}.dat -g -b $SW -e $EW > \
       $LDIR/order${I}.dat 2> ${OUTDIR}/order${I}.out &
     sleep 30
     sqlldr tpcc/tpcc control=$TPCC_LOADER/ordline_p${I}.ctl \
          log=${OUTDIR}/ordline${I}.log \
```

```
        bad=${OUTDIR}/ordline${I}.bad data=$LDIR/ordline${I}.dat \
        discard=${OUTDIR}/ordline${I}.dsc &
    sqlldr tpcc/tpcc control=$TPCC_LOADER/order_p${I}.ctl \
        log=${OUTDIR}/order${I}.log \
        bad=${OUTDIR}/order${I}.bad data=$LDIR/order${I}.dat \
        discard=${OUTDIR}/order${I}.dsc &
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

wait
for I in 1 2 3 4 5 6 7 8 9 10
do
  rm -f $LDIR/order${I}.dat
  rm -f $LDIR/ordline${I}.dat
done
date

echo "Done"
```

Pld_ord3.sh

```
#
#================================================================+
# FILENAME
#      pld_ord4.sh
# DESCRIPTION
#
#================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts

MULT=11000

PATH=${PATH}:$TPCC_SOURCE
export PATH
export LDIR=/tmp/data
echo $LDIR

set -x

if [ -d ${LDIR} ]
then rm -f ${LDIR}/*
else mkdir ${LDIR}
fi

#-----------------------------------------------------------------------
#  Load order and order-line table
#-----------------------------------------------------------------------

OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/p4
SW=661
EW=880
INC=1100
for I in 1 2 3 4 5 6 7 8 9 10
do
    echo "Creating pipes ${LDIR}/ordline${I}.dat, ${LDIR}/order${I}.dat, SW=$SW, EW=$EW,
MULT=$MULT"
    mknod ${LDIR}/ordline${I}.dat p
    mknod ${LDIR}/order${I}.dat p
    tpccload -M $MULT -o $LDIR/ordline${I}.dat -g -b $SW -e $EW > \
      $LDIR/order${I}.dat 2> ${OUTDIR}/order${I}.out &
    sleep 30
    sqlldr tpcc/tpcc control=$TPCC_LOADER/ordline_p${I}.ctl \
        log=${OUTDIR}/ordline${I}.log \
        bad=${OUTDIR}/ordline${I}.bad data=$LDIR/ordline${I}.dat \
        discard=${OUTDIR}/ordline${I}.dsc &
    sqlldr tpcc/tpcc control=$TPCC_LOADER/order_p${I}.ctl \
        log=${OUTDIR}/order${I}.log \
        bad=${OUTDIR}/order${I}.bad data=$LDIR/order${I}.dat \
        discard=${OUTDIR}/order${I}.dsc &
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

wait
for I in 1 2 3 4 5 6 7 8 9 10
do
  rm -f $LDIR/order${I}.dat
  rm -f $LDIR/ordline${I}.dat
done
date

echo "Done"
```

pld_ord4.sh

```
#
#================================================================+
# FILENAME
#      pld_ord4.sh
# DESCRIPTION
#
#================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts

MULT=11000

PATH=${PATH}:$TPCC_SOURCE
export PATH
export LDIR=/tmp/data
echo $LDIR

set -x

if [ -d ${LDIR} ]
then rm -f ${LDIR}/*
else mkdir ${LDIR}
fi

#-----------------------------------------------------------------------
#  Load order and order-line table
#-----------------------------------------------------------------------

OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/p4
SW=661
EW=880
INC=1100
for I in 1 2 3 4 5 6 7 8 9 10
do
    echo "Creating pipes ${LDIR}/ordline${I}.dat, ${LDIR}/order${I}.dat, SW=$SW, EW=$EW,
MULT=$MULT"
    mknod ${LDIR}/ordline${I}.dat p
    mknod ${LDIR}/order${I}.dat p
    tpccload -M $MULT -o $LDIR/ordline${I}.dat -g -b $SW -e $EW > \
      $LDIR/order${I}.dat 2> ${OUTDIR}/order${I}.out &
    sleep 30
    sqlldr tpcc/tpcc control=$TPCC_LOADER/ordline_p${I}.ctl \
        log=${OUTDIR}/ordline${I}.log \
        bad=${OUTDIR}/ordline${I}.bad data=$LDIR/ordline${I}.dat \
        discard=${OUTDIR}/ordline${I}.dsc &
    sqlldr tpcc/tpcc control=$TPCC_LOADER/order_p${I}.ctl \
        log=${OUTDIR}/order${I}.log \
        bad=${OUTDIR}/order${I}.bad data=$LDIR/order${I}.dat \
        discard=${OUTDIR}/order${I}.dsc &
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

wait
for I in 1 2 3 4 5 6 7 8 9 10
do
  rm -f $LDIR/order${I}.dat
  rm -f $LDIR/ordline${I}.dat
done
date

echo "Done"
```

pld_ord5.sh

```
#
#================================================================+
# FILENAME
#      pld_ord5.sh
# DESCRIPTION
#
#================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
```

```
TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts

MULT=11000

PATH=${PATH}:$TPCC_SOURCE
export PATH
export LDIR=/tmp/data
echo $LDIR

set -x

if [ -d ${LDIR} ]
then rm -f ${LDIR}/*
else mkdir ${LDIR}
fi

#----------------------------------------------------------------------
#  Load order and order-line table
#----------------------------------------------------------------------

OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/p5
SW=881
EW=1100
INC=1100
for I in 1 2 3 4 5 6 7 8 9 10
do
    echo "Creating pipes ${LDIR}/ordline${I}.dat, ${LDIR}/order${I}.dat, SW=$SW, EW=$EW,
MULT=$MULT"
    mknod ${LDIR}/ordline${I}.dat p
    mknod ${LDIR}/order${I}.dat p
    tpccload -M $MULT -o $LDIR/ordline${I}.dat -g -b $SW -e $EW > \
      $LDIR/order${I}.dat 2> ${OUTDIR}/order${I}.out &
    sleep 30
    sqlldr tpcc/tpcc control=$TPCC_LOADER/ordline_p${I}.ctl \
        log=${OUTDIR}/ordline${I}.log \
        bad=${OUTDIR}/ordline${I}.bad data=$LDIR/ordline${I}.dat \
        discard=${OUTDIR}/ordline${I}.dsc &
    sqlldr tpcc/tpcc control=$TPCC_LOADER/order_p${I}.ctl \
        log=${OUTDIR}/order${I}.log \
        bad=${OUTDIR}/order${I}.bad data=$LDIR/order${I}.dat \
        discard=${OUTDIR}/order${I}.dsc &
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

wait
for I in 1 2 3 4 5 6 7 8 9 10
do
  rm -f $LDIR/order${I}.dat
  rm -f $LDIR/ordline${I}.dat
done
date

echo "Done"
```

pldhist.sh

```
#
#
#=====================================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
#                  OPEN SYSTEMS PERFORMANCE GROUP                 |
#                      All Rights Reserved                        |
#=====================================================================+
# FILENAME
#      pldhist.sh
# DESCRIPTION
#      Usage: pldhist.sh
#
#=====================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
#DDM###TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_LOADER=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/history
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/history/pld_log

MULT=11000

PATH=${PATH}:$TPCC_SOURCE
export PATH
export LDIR=$LDIR/n1
echo $LDIR

if [ -d ${LDIR} ]
then rm -f ${LDIR}/*
else mkdir ${LDIR}
fi

#----------------------------------------------------------------------
#  Load history table
#----------------------------------------------------------------------
SW=1
EW=2200
INC=2200
for I in  1 2 3 4 5
do
  echo Creating named pipe ${LDIR}/hist${I}.dat, SW=$SW, EW=$EW, MULT=$MULT.....
  mknod ${LDIR}/hist${I}.dat p
  tpccload -M $MULT -h -g -b $SW -e $EW > $LDIR/hist${I}.dat 2> \
  ${OUTDIR}/hist${I}.out &
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

sleep 30

#
#  1. Load flatfiles into partitions
#
for I in  1 2 3 4 5
do
    sqlldr tpcc/tpcc control=$TPCC_LOADER/hist_p$I.ctl log=${OUTDIR}/hist$I.log \
        bad=${OUTDIR}/hist$I.bad data=$LDIR/hist$I.dat \
        discard=${OUTDIR}/hist$I.dsc &
done

#----------------------------------------------------------------------
# Wait for all processes to end, then clean up the pipes
#----------------------------------------------------------------------

wait

for I in 1 2 3 4 5
do
  rm -f $LDIR/hist${I}.dat
done
```

Pldnord.sh

```
#
#=====================================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
#                  OPEN SYSTEMS PERFORMANCE GROUP                 |
#                      All Rights Reserved                        |
#=====================================================================+
# FILENAME
#      pldnord.sh
# DESCRIPTION
#      Usage: pldnord.sh
#
#=====================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
#DDM###TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_LOADER=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/new_order
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/new_order/pld_log

MULT=11000

PATH=${PATH}:$TPCC_SOURCE
export PATH
export LDIR=$LDIR/n1
echo $LDIR

if [ -d ${LDIR} ]
then rm -f ${LDIR}/*
else mkdir ${LDIR}
fi

#----------------------------------------------------------------------
#  Load new-order table
#----------------------------------------------------------------------
echo "Starting $0: `date`"
SW=1
EW=2200
INC=2200
```

```
for I in 1 2 3 4 5
do
    echo Creating named pipe ${LDIR}/neword${I}.dat, SW=$SW, EW=$EW,
MULT=$MULT.....
      mknod ${LDIR}/neword${I}.dat p
      tpccload -M $MULT -n -g -b $SW -e $EW > $LDIR/neword${I}.dat 2> \
        ${OUTDIR}/neword${I}.out &
   SW=`expr $SW + $INC`
   EW=`expr $EW + $INC`
done

sleep 30

for I in 1 2 3 4 5
do
      sqlldr tpcc/tpcc control=$TPCC_LOADER/nord_p${I}.ctl \
          log=${OUTDIR}/neword${I}.log \
          bad=${OUTDIR}/neword${I}.bad data=$LDIR/neword${I}.dat \
          discard=${OUTDIR}/neword${I}.dsc &
done

#-------------------------------------------------------------------
# Wait for all processes to end, then clean up the pipes
#-------------------------------------------------------------------

wait

for I in 1 2 3 4 5
do
   rm -f $LDIR/neword${I}.dat
done

echo "Ending $0: `date`"
```

pload.sh

```
#
# $Header: pload.sh 7030100.1 96/05/02 19:06:06 plai Generic<base> $ Copyr (c) 1995
Oracle
#

#
#===================================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
#                  OPEN SYSTEMS PERFORMANCE GROUP                 |
#                     All Rights Reserved                         |
#===================================================================+
# FILENAME
#      pload.sh
# DESCRIPTION
#      Usage: pload.sh [options]
#           -mu <multiplier>     (# of warehouses)
# MODIFIED
#===================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_LOADER=$BENCH_HOME/tpcc/loader

LDIR=data
OUTDIR=outdir
MULT=3000

PATH=${PATH}:$TPCC_SOURCE
export PATH

if echo "\c" | grep c >/dev/null 2>&1; then
   N='-n'
else
   C='\c'
fi
export N C

while [ "$#" != "0" ]
do
  case $1 in
     -mu) shift
         if [ "$1" != "" ]
         then
            MULT=$1
            shift
         fi
         ;;
     -nd) shift
         NO_DB="y"
         ;;
     -nt) shift
         NO_TAB="y"
         ;;
     -nx) shift
         NO_IND="y"
         ;;
      *) echo "Bag arg: $1"
         exit 1;
         ;;
  esac
done

if [ "$MULT" = "" ]
then
  echo $N "Database multiplier (# of warehouses)? [1]" $C
  read MULT
  if [ "$MULT" = "" ]
  then
     MULT=1
  fi
fi

if [ ! -d $LDIR ]
then
  mkdir $LDIR
fi

if [ ! -d $OUTDIR ]
then
  mkdir $OUTDIR
fi

#
#  Load history table
#

I=1
while [ $I -le 4 ]
do
  mknod ${LDIR}/hist${I}.dat p
  I=`expr $I + 1`
done

I=1
SW=1
EW=750
INC=750
while [ $I -le 4 ]
do
  tpccload -M $MULT -h -g -b $SW -e $EW > ${LDIR}/hist${I}.dat 2> \
    ${OUTDIR}/hist${I}.out &
  I=`expr $I + 1`
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

sleep 30

I=1
while [ $I -le 4 ]
do
  sqlldr tpcc/tpcc control=$TPCC_LOADER/hist.ctl \
       log=${OUTDIR}/hist${I}.log \
       bad=${OUTDIR}/hist${I}.bad data=${LDIR}/hist${I}.dat \
       discard=${OUTDIR}/hist${I}.dsc \
       file=/dev/rlvhist${I} &
  I=`expr $I + 1`
done

wait

I=1
while [ $I -le 4 ]
do
  rm -f ${LDIR}/hist${I}.dat
  I=`expr $I + 1`
done

#
#  Load new-order table
#

I=1
mknod ${LDIR}/neword${I}.dat p

SW=1
tpccload -M $MULT -n -g -b $SW -e $MULT > ${LDIR}/neword${I}.dat 2> \
    ${OUTDIR}/neword${I}.out &

sleep 30

  sqlldr tpcc/tpcc control=$TPCC_LOADER/neword.ctl \
       log=${OUTDIR}/neword${I}.log \
       bad=${OUTDIR}/neword${I}.bad data=${LDIR}/neword${I}.dat \
       discard=${OUTDIR}/neword${I}.dsc \
       file=/dev/rlvnord${I}  &

wait

  rm -f ${LDIR}/neword${I}.dat

#
```

```
#  Load order and order-line table
#

l=1
while [ $l -le 44 ]
do
  mknod ${LDIR}/order${l}.dat p
  mknod ${LDIR}/ordline${l}.dat p
  l=`expr $l + 1`
done

l=1
SW=1
EW=69
INC=69
while [ $l -le 43 ]
do
  tpccload -M $MULT -o ${LDIR}/ordline${l}.dat -g -b $SW -e $EW > \
    ${LDIR}/order${l}.dat 2> ${OUTDIR}/order${l}.out &
  l=`expr $l + 1`
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

  tpccload -M $MULT -o ${LDIR}/ordline${l}.dat -g -b $SW -e $MULT > \
    ${LDIR}/order${l}.dat 2> ${OUTDIR}/order${l}.out &

sleep 30

l=1
while [ $l -le 44 ]
do
   J=`expr $l - 1`
   J=`expr $J / 11`
   J=`expr $J + 1`
  sqlldr tpcc/tpcc control=$TPCC_LOADER/order.ctl \
       log=${OUTDIR}/order${l}.log \
       bad=${OUTDIR}/order${l}.bad data=${LDIR}/order${l}.dat \
        discard=${OUTDIR}/order${l}.dsc \
        file=/dev/rlvord${J} &
  sqlldr tpcc/tpcc control=$TPCC_LOADER/ordline.ctl \
       log=${OUTDIR}/ordline${l}.log \
       bad=${OUTDIR}/ordline${l}.bad data=${LDIR}/ordline${l}.dat \
        discard=${OUTDIR}/ordline${l}.dsc \
        file=/dev/rlvordl${l} &
  l=`expr $l + 1`
done

wait


l=1
while [ $l -le 44 ]
do
  rm -f ${LDIR}/order${l}.dat
  rm -f ${LDIR}/ordline${l}.dat
  l=`expr $l + 1`
Done
```

Pload_cust.sh

```
#
#================================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                  OPEN SYSTEMS PERFORMANCE GROUP                |
#                     All Rights Reserved                        |
#================================================================+
# FILENAME
#       pload_cust.sh
# DESCRIPTION
#  Usage: pload_cust.sh
# MODIFIED
#      YPang for 5 nodes BB cluster.
#        Database size 11000 Warehouses.
#        Partitioned for 5 nodes.
#
#================================================================*/


#
#    Run pload_cust_n*.sh on each node
#
date
    pload_cust_n1.sh
date
    pload_cust_n2.sh
date
    pload_cust_n3.sh
date
    pload_cust_n4.sh
date
    pload_cust_n5.sh
date


Exit
```

Pload_cust_n1.sh

```
#================================================================+
# FILE_NAME
#         pload_cust_n1.sh
#================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/n1

MULT=11000

PATH=${PATH}:$TPCC_SOURCE

 N=1 #for node1


 # Load customer table (Nodes * Procs/Node * Warehouses/Proc --> 5*20*110 = 11000
TotalWarehouses) -----

 l=1
 LIMIT=20

 SW=1
 EW=110
 INC=110
 echo "`hostname -s`: l=${l}, LIMIT=${LIMIT}, SW=${SW}, EW=${EW}, INC=${INC}"
 while [ ${l} -le ${LIMIT} ]
 do
   echo "`hostname -s`: tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${l}.out
2>&1 &"
   tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${l}.out  2>&1 &
   l=`expr $l + 1`
   SW=`expr $SW + $INC`
   EW=`expr $EW + $INC`
 done


Wait
```

Pload_cust_n2.sh

```
#================================================================+
# FILE_NAME
#         pload_cust_n2.sh
#================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/n2

MULT=11000

PATH=${PATH}:$TPCC_SOURCE

N=2 #No for node2


 # Load customer table (Nodes * Procs/Node * Warehouses/Proc --> 5*20*110 = 11000
TotalWarehouses) -----

 l=1
 LIMIT=20

 SW=2201
 EW=2310
 INC=110
 echo "`hostname -s`: l=${l}, LIMIT=${LIMIT}, SW=${SW}, EW=${EW}, INC=${INC}"
 while [ ${l} -le ${LIMIT} ]
 do
```

```
    echo "`hostname -s`: tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${I}.out
2>&1 &"
    tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${I}.out  2>&1 &
    I=`expr $I + 1`
    SW=`expr $SW + $INC`
    EW=`expr $EW + $INC`
  done


Wait
```

Pload_cust_n3.sh

```
#================================================================+
# FILE_NAME
#          pload_cust_n3.sh
#================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/n3

MULT=11000

PATH=${PATH}:$TPCC_SOURCE

N=3 #No for node3


  # Load customer table (Nodes *  Procs/Node * Warehouses/Proc  --> 5*20*110 = 11000
TotalWarehouses) -----

  I=1
  LIMIT=20

  SW=4401
  EW=4510
  INC=110
  echo "`hostname -s`: I=${I}, LIMIT=${LIMIT},SW=${SW},  EW=${EW},  INC=${INC}"
  while [ ${I} -le ${LIMIT} ]
  do
    echo "`hostname -s`: tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${I}.out 2>&1
&"
    tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${I}.out 2>&1 &
    I=`expr $I + 1`
    SW=`expr $SW + $INC`
    EW=`expr $EW + $INC`
  done


Wait
```

pload_cust_n5.sh

```
#================================================================+
# FILE_NAME
#          pload_cust_n5.sh
#================================================================
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_OUTPUT=$BENCH_HOME/tpcc/output
TPCC_ADMIN=$BENCH_HOME/tpcc/admin
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_LOADER=$BENCH_HOME/tpcc/loader
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/n5

MULT=11000

PATH=${PATH}:$TPCC_SOURCE

N=5 #No for node5

  # Load customer table (Nodes * Procs/Node * Warehouses/Proc --> 5*20*110 = 11000
TotalWarehouses) -----

  I=1
  LIMIT=20

  SW=8801
  EW=8910
  INC=110
  echo "`hostname -s`: I=${I}, LIMIT=${LIMIT}, SW=${SW}, EW=${EW}, INC=${INC}"
  while [ ${I} -le ${LIMIT} ]
  do
    echo "`hostname -s`: tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${I}.out
2>&1 &"
    tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${I}.out  2>&1 &
    I=`expr $I + 1`
    SW=`expr $SW + $INC`
    EW=`expr $EW + $INC`
  done


Wait
```

Pload_stk.sh

```
#================================================================+
# FILE_NAME
#          pload_stk_n1.sh
#================================================================
#
  BENCH_HOME=$ORACLE_HOME/bench/tpc
  BENCH_GEN=$ORACLE_HOME/bench/gen
  GEN_SQL=$BENCH_GEN/sql
  TPCC_SOURCE=$BENCH_HOME/tpcc/source
  TPCC_SQL=$BENCH_HOME/tpcc/sql
  TPCC_OUTPUT=$BENCH_HOME/tpcc/output
  TPCC_ADMIN=$BENCH_HOME/tpcc/admin
  TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
  TPCC_LOADER=$BENCH_HOME/tpcc/loader
  TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
  OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir

  MULT=11000

  PATH=${PATH}:$TPCC_SOURCE

#  N=1 #for node1



#---- Load stock table (Nodes * Procs/Node * Items/Proc --> 5*20*1000 = 100000 =
TotalItems) -----

  I=1
  LIMIT=20

  SI=1
  EI=1000
  INC=1000

while [ ${I} -le ${LIMIT} ]
do
  echo "`hostname -s`: tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &"
  tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &
  I=`expr $I + 1`
  SI=`expr $SI + $INC`
  EI=`expr $EI + $INC`
done

Wait
```

pload_stk_n1.sh

```
#================================================================+
# FILE_NAME
#          pload_stk_n1.sh
#================================================================
#
  BENCH_HOME=$ORACLE_HOME/bench/tpc
  BENCH_GEN=$ORACLE_HOME/bench/gen
  GEN_SQL=$BENCH_GEN/sql
  TPCC_SOURCE=$BENCH_HOME/tpcc/source
  TPCC_SQL=$BENCH_HOME/tpcc/sql
  TPCC_OUTPUT=$BENCH_HOME/tpcc/output
  TPCC_ADMIN=$BENCH_HOME/tpcc/admin
  TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
  TPCC_LOADER=$BENCH_HOME/tpcc/loader
  TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
  OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir

  MULT=11000

  PATH=${PATH}:$TPCC_SOURCE

#  N=1 #for node1
```

```
#---- Load stock table (Nodes * Procs/Node * Items/Proc --> 5*20*1000 = 100000 =
TotalItems) -----

 I=1
 LIMIT=20

 SI=1
 EI=1000
 INC=1000

while [ ${I} -le ${LIMIT} ]
do
  echo "`hostname -s`: tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &"
  tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &
  I=`expr $I + 1`
  SI=`expr $SI + $INC`
  EI=`expr $EI + $INC`
done

wait
```

pload_stk_n2.sh

```
#================================================================+
# FILE_NAME
#         pload_stk_n2.sh
#================================================================
#
  BENCH_HOME=$ORACLE_HOME/bench/tpc
  BENCH_GEN=$ORACLE_HOME/bench/gen
  GEN_SQL=$BENCH_GEN/sql
  TPCC_SOURCE=$BENCH_HOME/tpcc/source
  TPCC_SQL=$BENCH_HOME/tpcc/sql
  TPCC_OUTPUT=$BENCH_HOME/tpcc/output
  TPCC_ADMIN=$BENCH_HOME/tpcc/admin
  TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
  TPCC_LOADER=$BENCH_HOME/tpcc/loader
  TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
  OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/n2

  MULT=11000

  PATH=${PATH}:$TPCC_SOURCE

  N=2 #for node2



#---- Load stock table (Nodes * Procs/Node * Items/Proc --> 5*20*1000 = 100000 =
TotalItems) -----

 I=1
 LIMIT=20

 SI=20001
 EI=21000
 INC=1000

while [ ${I} -le ${LIMIT} ]
do
  echo "`hostname -s`: tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &"
  tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &
  I=`expr $I + 1`
  SI=`expr $SI + $INC`
  EI=`expr $EI + $INC`
done

wait
```

pload_stk_n3.sh

```
#================================================================+
# FILE_NAME
#         pload_stk_n3.sh
#================================================================
#
  BENCH_HOME=$ORACLE_HOME/bench/tpc
  BENCH_GEN=$ORACLE_HOME/bench/gen
  GEN_SQL=$BENCH_GEN/sql
  TPCC_SOURCE=$BENCH_HOME/tpcc/source
  TPCC_SQL=$BENCH_HOME/tpcc/sql
  TPCC_OUTPUT=$BENCH_HOME/tpcc/output
  TPCC_ADMIN=$BENCH_HOME/tpcc/admin
  TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
  TPCC_LOADER=$BENCH_HOME/tpcc/loader
  TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
  OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/n3

  MULT=11000

  PATH=${PATH}:$TPCC_SOURCE

  N=3 #for node3



#---- Load stock table (Nodes * Procs/Node * Items/Proc --> 5*20*1000 = 100000 =
TotalItems) -----

 I=1
 LIMIT=20

 SI=40001
 EI=41000
 INC=1000

while [ ${I} -le ${LIMIT} ]
do
  echo "`hostname -s`: tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &"
  tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &
  I=`expr $I + 1`
  SI=`expr $SI + $INC`
  EI=`expr $EI + $INC`
done

Wait
```

Pload_stk_n4.sh

```
#================================================================+
# FILE_NAME
#         pload_stk_n4.sh
#================================================================
#
set -x
  BENCH_HOME=$ORACLE_HOME/bench/tpc
  BENCH_GEN=$ORACLE_HOME/bench/gen
  GEN_SQL=$BENCH_GEN/sql
  TPCC_SOURCE=$BENCH_HOME/tpcc/source
  TPCC_SQL=$BENCH_HOME/tpcc/sql
  TPCC_OUTPUT=$BENCH_HOME/tpcc/output
  TPCC_ADMIN=$BENCH_HOME/tpcc/admin
  TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
  TPCC_LOADER=$BENCH_HOME/tpcc/loader
  TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
  OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/n4

  MULT=11000

  PATH=${PATH}:$TPCC_SOURCE

  N=4 #for node4



#---- Load stock table (Nodes * Procs/Node * Items/Proc --> 5*20*1000 = 100000 =
TotalItems) -----

 I=1
 LIMIT=20

 SI=60001
 EI=61000
 INC=1000

while [ ${I} -le ${LIMIT} ]
do
  echo "`hostname -s`: tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &"
  tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &
  I=`expr $I + 1`
  SI=`expr $SI + $INC`
  EI=`expr $EI + $INC`
done

Wait
```

pload_stk_n5.sh

```
#================================================================+
# FILE_NAME
#         pload_stk_n5.sh
#================================================================
#
set -x
  BENCH_HOME=$ORACLE_HOME/bench/tpc
  BENCH_GEN=$ORACLE_HOME/bench/gen
  GEN_SQL=$BENCH_GEN/sql
  TPCC_SOURCE=$BENCH_HOME/tpcc/source
  TPCC_SQL=$BENCH_HOME/tpcc/sql
  TPCC_OUTPUT=$BENCH_HOME/tpcc/output
```

```
  TPCC_ADMIN=$BENCH_HOME/tpcc/admin
  TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
  TPCC_LOADER=$BENCH_HOME/tpcc/loader
  TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
  OUTDIR=$ORACLE_HOME/bench/tpc/tpcc/scripts/5s7a/outdir/n5

  MULT=11000

  PATH=${PATH}:$TPCC_SOURCE

  N=5 #for node5



#---- Load stock table (Nodes * Procs/Node * Items/Proc --> 5*20*1000 = 100000 =
TotalItems) -----

  I=1
  LIMIT=20

  SI=80001
  EI=81000
  INC=1000

while [ ${I} -le ${LIMIT} ]
do
   echo "`hostname -s`: tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &"
   tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out  2>&1 &
   I=`expr $I + 1`
   SI=`expr $SI + $INC`
   EI=`expr $EI + $INC`
done


wait
```

switchlog.sh

```
#
# $Header: switchlog.sh 7030100.1 96/05/02 10:20:11 plai Generic<base> $ Copyr (c) 1995
Oracle
#

#=============================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
#                  OPEN SYSTEMS PERFORMANCE GROUP                 |
#                     All Rights Reserved                         |
#=============================================================+
# FILENAME
#     switchlog.sh
# DESCRIPTION
#     Switch to next log file twice.
# USAGE
#     switchlog.sh
#=============================================================*/

svrmgrl <<!
   connect internal;
   alter system switch logfile;
   alter system switch logfile;
   exit;
!
```

undml.sh

```
#
# $Header: undml.sh 7030100.2 96/05/02 10:29:30 plai Generic<base> $ Copyr (c) 1995
Oracle
#

#=============================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
#                  OPEN SYSTEMS PERFORMANCE GROUP                 |
#                     All Rights Reserved                         |
#=============================================================+
# FILENAME
#     undml.sh
# DESCRIPTION
#     Enable table locks for TPC-C tables.
# USAGE
#     undml.sh
#=============================================================*/

sqlplus tpcc/tpcc <<!
   alter table warehouse  enable table lock;
   alter table district   enable table lock;
   alter table customer   enable table lock;
   alter table history    enable table lock;
   alter table item       enable table lock;
   alter table stock      enable table lock;
   alter table orders     enable table lock;
   alter table new_order  enable table lock;
   alter table order_line enable table lock;
   quit;
!
```

C.2 SQL Scripts

Pb_roll_cr.sql

```
set echo on
set termout on
host date;
create rollback segment s1   storage(initial 500k minextents 2 next 500k);
create rollback segment s2   storage(initial 500k minextents 2 next 500k);
create rollback segment s3   storage(initial 500k minextents 2 next 500k);
create rollback segment s4   storage(initial 500k minextents 2 next 500k);
create rollback segment s5   storage(initial 500k minextents 2 next 500k);
create rollback segment s6   storage(initial 500k minextents 2 next 500k);
create rollback segment s7   storage(initial 500k minextents 2 next 500k);
create rollback segment s8   storage(initial 500k minextents 2 next 500k);
create rollback segment s9   storage(initial 500k minextents 2 next 500k);
create rollback segment s10  storage(initial 500k minextents 2 next 500k);
create rollback segment s11  storage(initial 500k minextents 2 next 500k);
create rollback segment s12  storage(initial 500k minextents 2 next 500k);
create rollback segment s13  storage(initial 500k minextents 2 next 500k);
create rollback segment s14  storage(initial 500k minextents 2 next 500k);
create rollback segment s15  storage(initial 500k minextents 2 next 500k);
create rollback segment s16  storage(initial 500k minextents 2 next 500k);
create rollback segment s17  storage(initial 500k minextents 2 next 500k);
create rollback segment s18  storage(initial 500k minextents 2 next 500k);
create rollback segment s19  storage(initial 500k minextents 2 next 500k);
create rollback segment s20  storage(initial 500k minextents 2 next 500k);
create rollback segment s21  storage(initial 500k minextents 2 next 500k);
create rollback segment s22  storage(initial 500k minextents 2 next 500k);
create rollback segment s23  storage(initial 500k minextents 2 next 500k);
create rollback segment s24  storage(initial 500k minextents 2 next 500k);
create rollback segment s25  storage(initial 500k minextents 2 next 500k);
create rollback segment s26  storage(initial 500k minextents 2 next 500k);
create rollback segment s27  storage(initial 500k minextents 2 next 500k);
create rollback segment s28  storage(initial 500k minextents 2 next 500k);
create rollback segment s29  storage(initial 500k minextents 2 next 500k);
create rollback segment s30  storage(initial 500k minextents 2 next 500k);
create rollback segment s31  storage(initial 500k minextents 2 next 500k);
create rollback segment s32  storage(initial 500k minextents 2 next 500k);
create rollback segment s33  storage(initial 500k minextents 2 next 500k);
create rollback segment s34  storage(initial 500k minextents 2 next 500k);
create rollback segment s35  storage(initial 500k minextents 2 next 500k);
create rollback segment s36  storage(initial 500k minextents 2 next 500k);
create rollback segment s37  storage(initial 500k minextents 2 next 500k);
create rollback segment s38  storage(initial 500k minextents 2 next 500k);
create rollback segment s39  storage(initial 500k minextents 2 next 500k);
create rollback segment s40  storage(initial 500k minextents 2 next 500k);
create rollback segment s41  storage(initial 500k minextents 2 next 500k);
create rollback segment s42  storage(initial 500k minextents 2 next 500k);
create rollback segment s43  storage(initial 500k minextents 2 next 500k);
create rollback segment s44  storage(initial 500k minextents 2 next 500k);
create rollback segment s45  storage(initial 500k minextents 2 next 500k);
create rollback segment s46  storage(initial 500k minextents 2 next 500k);
create rollback segment s47  storage(initial 500k minextents 2 next 500k);
create rollback segment s48  storage(initial 500k minextents 2 next 500k);
create rollback segment s49  storage(initial 500k minextents 2 next 500k);
create rollback segment s50  storage(initial 500k minextents 2 next 500k);
host date;
exit;
```

Tpcc_ana_idx.sql

```
spool ana_index1.info
analyze index iwarehouse validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, '          ' as rows_p_brblk,
                    distinct_keys, pct_used from index_stats;
analyze index idistrict validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, round(br_rows/br_blks) as rows_p_brblk,
                    distinct_keys, pct_used from index_stats;
analyze index icustomer validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, round(br_rows/br_blks) as rows_p_brblk,
                    distinct_keys, pct_used from index_stats;
analyze index icustomer2 validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, round(br_rows/br_blks) as rows_p_brblk,
                    distinct_keys, pct_used from index_stats;
analyze index istock validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, round(br_rows/br_blks) as rows_p_brblk,
```

```
                    distinct_keys, pct_used from index_stats;
analyze index iitem validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, round(br_rows/br_blks) as rows_p_brblk,
                    distinct_keys, pct_used from index_stats;
analyze index iorders validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, round(br_rows/br_blks) as rows_p_brblk,
                    distinct_keys, pct_used from index_stats;
analyze index iorders2 validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, round(br_rows/br_blks) as rows_p_brblk,
                    distinct_keys, pct_used from index_stats;
analyze index inew_order validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, round(br_rows/br_blks) as rows_p_brblk,
                    distinct_keys, pct_used from index_stats;
analyze index iorder_line validate structure;
select name, height, blocks, lf_rows, lf_blks, round(lf_rows/lf_blks) as rows_p_lfblk,
                    br_rows, br_blks, round(br_rows/br_blks) as rows_p_brblk,
                    distinct_keys, pct_used from index_stats;
spool off
```

tpcc_ana.sql

```
rem
rem ================================================================+
rem       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
rem                 OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                    All Rights Reserved                         |
rem ================================================================+
rem  FILENAME
rem      tpcc_ana.sql
rem  DESCRIPTION
rem      Analyze all tables and indexes of TPC-C database.
rem ================================================================
rem

set timing on;
analyze table warehouse compute statistics;
analyze table district compute statistics;
analyze table item estimate statistics;
analyze table history estimate statistics;
analyze table customer estimate statistics;
analyze table stock estimate statistics;
analyze table orders estimate statistics;
analyze table new_order estimate statistics;
analyze table order_line estimate statistics;
analyze cluster icluster estimate statistics;
analyze cluster scluster estimate statistics;
analyze cluster ccluster estimate statistics;
analyze index iwarehouse compute statistics;
analyze index idistrict compute statistics;
analyze index icustomer estimate statistics;
analyze index icustomer2 estimate statistics;
analyze index istock estimate statistics;
analyze index iitem estimate statistics;
analyze index iorders estimate statistics;
analyze index iorders2 estimate statistics;
analyze index inew_order estimate statistics;
analyze index iorder_line estimate statistics;
Quit;
```

Tpcc_ana_tab.sql

```
rem
rem ================================================================+
rem       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
rem                 OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                    All Rights Reserved                         |
rem ================================================================+
rem  FILENAME
rem      tpcc_ana.sql
rem  DESCRIPTION
rem      Analyze all tables and indexes of  TPC-C database.
rem ================================================================
rem

set timing on;
analyze table   warehouse compute statistics;
analyze table   district compute statistics;
analyze table   item estimate statistics;
analyze table   history estimate statistics;
analyze table   customer estimate statistics;
analyze table   stock estimate statistics;
analyze table   orders estimate statistics;
analyze table   new_order estimate statistics;
analyze table   order_line estimate statistics;
analyze cluster icluster estimate statistics;
analyze cluster scluster estimate statistics;
analyze cluster ccluster estimate statistics;
analyze index   iwarehouse compute statistics;
analyze index   idistrict compute statistics;
analyze index   icustomer estimate statistics;
analyze index   icustomer2 estimate statistics;
analyze index   istock estimate statistics;
analyze index   iitem estimate statistics;
analyze index   iorders estimate statistics;
analyze index   iorders2 estimate statistics;
analyze index   inew_order estimate statistics;
analyze index   iorder_line estimate statistics;
quit;
```

tpcc_ix1.sql

```
rem
rem ================================================================+
rem       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
rem                 OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                    All Rights Reserved                         |
rem ================================================================+
rem  FILENAME
rem      tpcc_ix1.sql
rem  DESCRIPTION
rem      Create indexes for TPC-C database.
rem  MODIFIED
rem      YPang for 5 nodes BB cluster.
rem         Database size 11000 Warehouses.
rem         Partitioned for 5 nodes.
rem
rem ================================================================
rem

drop index iwarehouse;
drop index idistrict;
drop index icustomer;
drop index icustomer2;
drop index istock;
drop index iitem;

set timing on

create unique index iwarehouse on warehouse(w_id)
      tablespace items
     pctfree 1 initrans 3 nologging
     storage(initial 2M next 1M pctincrease 0);

create unique index idistrict on district(d_w_id, d_id)
      tablespace items
     pctfree 1 initrans 3 nologging
     storage(initial 5M next 1M pctincrease 0);

create unique index iitem on item(i_id)
      tablespace items
     pctfree 1 initrans 3 nologging
     storage(initial 6M next 1M pctincrease 0);

create unique index icustomer on customer(c_w_id, c_d_id, c_id)
     tablespace icu1
     pctfree 1 initrans 3 nologging
     parallel ( degree 20 )
     storage (initial 227M next 25M pctincrease 0);

create unique index icustomer2 on customer(c_last,c_w_id,c_d_id,c_first,c_id)
     tablespace icu2
     pctfree 1 initrans 3 nologging
     parallel (degree 60)
     storage (initial 234M next 26M pctincrease 0);

create unique index istock on stock(s_i_id, s_w_id)
     tablespace istk
     pctfree 1 initrans 3 nologging
     parallel ( degree 25 )
     storage (initial 825M next 50M pctincrease 0);

exit;
```

tpcc_ix2.sql

```
rem
rem ================================================================+
rem       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
rem                 OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                    All Rights Reserved                         |
rem ================================================================+
rem  FILENAME
rem      tpcc_ix2.sql
rem  DESCRIPTION
rem      Create indexes for TPC-C database.
rem  MODIFIED
rem      YPang for 5 nodes BB cluster.
rem         Database size 11000 Warehouses.
rem         Partitioned for 5 nodes.
rem
rem ================================================================
rem
```

```
drop index iorders;
drop index iorders2;
drop index inew_order;
drop index iorder_line;

set timing on


create unique index iorders on orders(o_w_id, o_d_id, o_id)
      initrans 3
      pctfree 1
      nologging
      LOCAL (
           partition iord_p01 tablespace ior1_n1,
           partition iord_p02 tablespace ior1_n2,
           partition iord_p03 tablespace ior1_n3,
           partition iord_p04 tablespace ior1_n4,
           partition iord_p05 tablespace ior1_n5
          )
  storage (initial 101M next 101M pctincrease 0
         freelist groups 95 freelists 47)
  parallel ( degree 10 );


create unique index iorders2 on orders(o_w_id, o_d_id, o_c_id, o_id)
  initrans 3
  pctfree  25
  nologging
  LOCAL (
    partition ior2_p01 tablespace ior2_n1,
    partition ior2_p02 tablespace ior2_n2,
    partition ior2_p03 tablespace ior2_n3,
    partition ior2_p04 tablespace ior2_n4,
    partition ior2_p05 tablespace ior2_n5
  )
  storage (initial 102M next 102M pctincrease 0
         freelist groups 95 freelists 47)
  parallel ( degree 10 );


create unique index inew_order on new_order(no_w_id, no_d_id, no_o_id)
  LOCAL (
    partition inor_p01 tablespace inor_n1_1,
    partition inor_p02 tablespace inor_n2_1,
    partition inor_p03 tablespace inor_n3_1,
    partition inor_p04 tablespace inor_n4_1,
    partition inor_p05 tablespace inor_n5_1
  )
  nologging
  storage (initial 48M next 48M pctincrease 0 freelist groups 100 freelists 47)
  parallel ( degree 10 );


create unique index iorder_line
  on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
  initrans 4
  pctfree  1
  nologging
  LOCAL (
    partition iorl_p01 tablespace iorl_n1,
    partition iorl_p02 tablespace iorl_n2,
    partition iorl_p03 tablespace iorl_n3,
    partition iorl_p04 tablespace iorl_n4,
    partition iorl_p05 tablespace iorl_n5
  )
  storage (initial 265M next 265M pctincrease 0
         freelist groups 95 freelists 47)
  parallel ( degree 10 );

exit;
```

tpcc_n1_roll.sql

```
rem ------------------------------------------------------------------------
rem - Generating 350 segments per tablespace
rem
rem ------------------------------------------------------------------------

host date;
rem ------ Rollback segments for tablespace rol1
create rollback segment a1   tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a2   tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a3   tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a4   tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a5   tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a6   tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a7   tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a8   tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a9   tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a10  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a11  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a12  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a13  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a14  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a15  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a16  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a17  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a18  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a19  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a20  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a21  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a22  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a23  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a24  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a25  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a26  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a27  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a28  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a29  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a30  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a31  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a32  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a33  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a34  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a35  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a36  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a37  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a38  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a39  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a40  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a41  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a42  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a43  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a44  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a45  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a46  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a47  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a48  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a49  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a50  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a51  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a52  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a53  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a54  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a55  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a56  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a57  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a58  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a59  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a60  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a61  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a62  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a63  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a64  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a65  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a66  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a67  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a68  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a69  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a70  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a71  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a72  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a73  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a74  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a75  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a76  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a77  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a78  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a79  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a80  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a81  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a82  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a83  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a84  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a85  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a86  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a87  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a88  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a89  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a90  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a91  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a92  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a93  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a94  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a95  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a96  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a97  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a98  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a99  tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a100 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a101 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a102 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a103 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a104 tablespace rol1 storage(initial 100K next 100K minextents 2);
```

```
create rollback segment a105 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a106 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a107 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a108 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a109 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a110 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a111 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a112 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a113 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a114 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a115 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a116 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a117 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a118 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a119 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a120 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a121 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a122 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a123 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a124 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a125 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a126 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a127 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a128 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a129 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a130 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a131 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a132 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a133 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a134 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a135 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a136 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a137 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a138 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a139 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a140 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a141 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a142 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a143 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a144 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a145 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a146 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a147 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a148 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a149 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a150 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a151 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a152 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a153 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a154 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a155 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a156 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a157 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a158 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a159 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a160 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a161 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a162 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a163 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a164 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a165 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a166 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a167 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a168 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a169 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a170 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a171 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a172 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a173 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a174 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a175 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a176 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a177 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a178 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a179 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a180 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a181 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a182 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a183 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a184 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a185 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a186 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a187 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a188 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a189 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a190 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a191 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a192 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a193 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a194 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a195 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a196 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a197 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a198 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a199 tablespace rol1 storage(initial 100K next 100K minextents 2);
```

```
create rollback segment a200 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a201 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a202 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a203 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a204 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a205 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a206 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a207 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a208 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a209 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a210 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a211 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a212 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a213 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a214 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a215 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a216 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a217 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a218 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a219 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a220 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a221 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a222 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a223 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a224 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a225 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a226 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a227 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a228 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a229 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a230 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a231 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a232 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a233 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a234 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a235 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a236 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a237 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a238 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a239 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a240 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a241 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a242 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a243 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a244 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a245 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a246 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a247 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a248 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a249 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a250 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a251 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a252 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a253 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a254 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a255 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a256 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a257 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a258 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a259 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a260 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a261 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a262 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a263 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a264 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a265 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a266 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a267 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a268 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a269 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a270 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a271 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a272 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a273 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a274 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a275 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a276 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a277 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a278 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a279 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a280 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a281 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a282 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a283 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a284 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a285 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a286 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a287 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a288 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a289 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a290 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a291 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a292 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a293 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a294 tablespace rol1 storage(initial 100K next 100K minextents 2);
```

```
create rollback segment a295 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a296 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a297 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a298 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a299 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a300 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a301 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a302 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a303 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a304 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a305 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a306 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a307 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a308 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a309 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a310 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a311 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a312 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a313 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a314 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a315 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a316 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a317 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a318 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a319 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a320 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a321 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a322 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a323 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a324 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a325 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a326 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a327 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a328 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a329 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a330 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a331 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a332 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a333 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a334 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a335 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a336 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a337 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a338 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a339 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a340 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a341 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a342 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a343 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a344 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a345 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a346 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a347 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a348 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a349 tablespace rol1 storage(initial 100K next 100K minextents 2);
create rollback segment a350 tablespace rol1 storage(initial 100K next 100K minextents 2);
host date;
exit;
```

tpcc_tab.sql

```
rem
rem ============================================================+
rem       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA     |
rem                OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                   All Rights Reserved                         |
rem ============================================================+
rem  FILENAME
rem      tpcc_tab.sql
rem  DESCRIPTION
rem      Create tables for TPC-C database.
rem  MODIFIED
rem      YPang for 5 nodes BB cluster.
rem         Database size 11000 Warehouses.
rem         Partitioned for 5 nodes.
rem
rem ============================================================
rem

rem
rem FIRST, create TPCC userid and connect to it.
rem
      grant connect,resource,unlimited tablespace to tpcc identified by tpcc;
      alter user tpcc temporary tablespace temp;
      connect tpcc/tpcc;

rem
rem NEXT, DROP all first
rem
      drop cluster icluster including tables;
      drop table warehouse;
      drop table district;
      drop table history;
      drop table orders;
      drop table new_order;
      drop table order_line;
      drop table item;

set timing on;

rem
rem LAST, CREATE all tables
rem

rem
rem WAREHOUSE table
rem

            create table warehouse (
                  w_id          number,
                  w_ytd         number(12),
                  w_tax         number(4),
                  w_name        varchar2(10),
                  w_street_1    varchar2(20),
                  w_street_2    varchar2(20),
                  w_city        varchar2(20),
                  w_state       char(2),
                  w_zip         char(9)
            )
      tablespace ware
      initrans 4
      pctfree  98 pctused 1
      storage (initial 45000K next 1M pctincrease 0);


rem
rem DISTRICT table
rem

            create table district (
      d_id          number,
      d_w_id        number,
      d_ytd         number(12),
      d_tax         number(4),
      d_next_o_id   number,
      d_name        varchar2(10),
      d_street_1    varchar2(20),
      d_street_2    varchar2(20),
      d_city        varchar2(20),
      d_state       char(2),
      d_zip         char(9)
      )
      tablespace dist
      initrans 4
      pctfree  90 pctused 2
      storage (initial 94M next 94M pctincrease 0);

rem
rem done
rem

rem
rem HISTORY table
rem

          create table history (
      h_c_id        number,
      h_c_d_id      number,
      h_c_w_id      number,
      h_d_id        number,
      h_w_id        number,
      h_date        date,
      h_amount      number(6),
      h_data        varchar2(24)
      )
      partition by range (h_w_id)
      (
      partition hist_p01 values less than (2201)    tablespace hist_n1_1,
      partition hist_p02 values less than (4401)    tablespace hist_n2_1,
      partition hist_p03 values less than (6601)    tablespace hist_n3_1,
      partition hist_p04 values less than (8801)    tablespace hist_n4_1,
      partition hist_p05 values less than (MAXVALUE) tablespace hist_n5_1
      )
      initrans  3
      pctfree   1
      pctused  99
      storage (initial 1038M next 1038M maxextents unlimited
            pctincrease 0 freelist groups 100 freelists 47);

rem
rem ORDER table
rem

      create table orders (
      o_id          number,
      o_d_id        number,
      o_w_id        number,
      o_c_id        number,
```

```
    o_entry_d      date,
    o_carrier_id   number,
    o_ol_cnt       number,
    o_all_local    number
    )
    partition by range (o_w_id)
    (
    partition ords_p01 values less than (2201)    tablespace ords_n1,
    partition ords_p02 values less than (4401)    tablespace ords_n2,
    partition ords_p03 values less than (6601)    tablespace ords_n3,
    partition ords_p04 values less than (8801)    tablespace ords_n4,
    partition ords_p05 values less than (MAXVALUE) tablespace ords_n5
    )
    initrans  4
    pctfree   5
    pctused  95
    storage (initial 202M next 202M maxextents unlimited
          pctincrease 0 freelist groups 95 freelists 47);

rem
rem NEW_ORDER table
rem

    create table new_order (
    no_o_id        number,
    no_d_id        number,
    no_w_id        number
    )
    partition by range (no_w_id)
    (
    partition nord_p01 values less than (2201)    tablespace nord_n1_1,
    partition nord_p02 values less than (4401)    tablespace nord_n2_1,
    partition nord_p03 values less than (6601)    tablespace nord_n3_1,
    partition nord_p04 values less than (8801)    tablespace nord_n4_1,
    partition nord_p05 values less than (MAXVALUE) tablespace nord_n5_1
    )
    initrans  4
    pctfree   5
    pctused  95
    storage (initial 70M next 70M maxextents unlimited
          pctincrease 0 freelist groups 100 freelists 47);

rem
rem ORDER_LINE table
rem

    create table order_line (
    ol_o_id        number,
    ol_d_id        number,
    ol_w_id        number,
    ol_number      number,
    ol_delivery_d  date,
    ol_i_id        number,
    ol_supply_w_id number,
    ol_quantity    number,
    ol_amount      number(6),
    ol_dist_info   char(24)
    )
    partition by range (ol_w_id)
    (
    partition ordl_p01 values less than (2201)    tablespace ordl_n1,
    partition ordl_p02 values less than (4401)    tablespace ordl_n2,
    partition ordl_p03 values less than (6601)    tablespace ordl_n3,
    partition ordl_p04 values less than (8801)    tablespace ordl_n4,
    partition ordl_p05 values less than (MAXVALUE) tablespace ordl_n5
    )
    initrans  4
    pctfree   5
    pctused  95
    storage (initial 320M next 320M maxextents unlimited
          pctincrease 0 freelist groups 95 freelists 47);

rem
rem ITEM table
rem

    create cluster icluster (
    i_id           number(6,0)
    )
    hashkeys       100000
    hash is        i_id
    size           120
    initrans       3
    pctfree        0
    tablespace     items
    storage (initial 13M next 1M pctincrease 0);

    create table item (
    i_id           number(6,0),
    i_im_id        number,
    i_name         varchar2(24),
    i_price        number(5,0),
    i_data         varchar2(50)
    )
    cluster icluster(i_id);

rem
rem done
rem

    exit;
```

tpcc_tab2.sql

```
rem
rem +-------------------------------------------------------------+
rem |      Copyright (c) 1996  Oracle Corp, Redwood Shores, CA    |
rem |                OPEN SYSTEMS PERFORMANCE GROUP               |
rem |                   All Rights Reserved                       |
rem +-------------------------------------------------------------+
rem  FILENAME
rem      tpcc_tab2.sql
rem  DESCRIPTION
rem      Create customer table for TPC-C database.
rem  MODIFIED
rem      YPang for 5 nodes BB cluster.
rem         Database size 11000 Warehouses.
rem         Partitioned for 5 nodes.
rem
rem ============================================================
rem

rem
rem DROP all first
rem
    drop cluster ccluster including tables;
            drop table customer;

set timing on

rem
rem CUSTOMER table
rem

    create cluster ccluster (
      c_id         number(5,0),
      c_d_id       number(2,0),
      c_w_id       number(5,0)
    )
    hashkeys       330000000
    hash is        (c_w_id * 30000 + c_d_id * 3000 + c_id) - 33001
    size           850
    initrans       3
    pctfree        0
    tablespace     cust
    storage (initial 787504k  next 787500k pctincrease 0 minextents 440
    maxextents 1000 freelist groups 4);

            create table customer (
    c_id           number(5,0),
    c_d_id         number(2,0),
    c_w_id         number(5,0),
    c_first        varchar2(16),
    c_middle       char(2),
    c_last         varchar2(16),
    c_street_1     varchar2(20),
    c_street_2     varchar2(20),
    c_city         varchar2(20),
    c_state        char(2),
    c_zip          char(9),
    c_phone        char(16),
    c_since        date,
    c_credit       char(2),
    c_credit_lim   number(12),
    c_discount     number(4),
    c_balance      number(12),
    c_ytd_payment  number(12),
    c_payment_cnt  number(8),
    c_delivery_cnt number(8),
    c_data         varchar2(500)
    )
    cluster ccluster (c_id, c_d_id, c_w_id);

rem
rem done
rem

    exit;
```

tpcc_tab3.sql

```
rem
rem +-------------------------------------------------------------+
rem |      Copyright (c) 1996  Oracle Corp, Redwood Shores, CA    |
rem |                OPEN SYSTEMS PERFORMANCE GROUP               |
```

```
rem                    All Rights Reserved                    |
rem ============================================================+
rem  FILENAME
rem       tpcc_tab3.sql
rem  DESCRIPTION
rem       Create stock table for TPC-C database.
rem  MODIFIED
rem       YPang for 5 nodes BB cluster.
rem          Database size 11000 Warehouses.
rem          Partitioned for 5 nodes.
rem
rem ============================================================
rem
rem
rem
rem DROP all first
rem
       drop cluster scluster including tables;
                drop table stock;

set timing on

rem
rem STOCK table
rem

       create cluster scluster (
         s_i_id       number(6,0),
         s_w_id       number(5,0)
       )
       hashkeys       1100000000
       hash is        (abs(s_i_id - 1) * 2200 + mod((s_w_id - 1),2200) +
                    trunc ((s_w_id - 1) /2200) * 220000000)
       size           350
       initrans       3
       pctfree        0
       tablespace     stoc
       storage (initial 378383k next 378379k minextents 1110 pctincrease 0
       maxextents UNLIMITED freelist groups 3);

                create table stock (
       s_i_id         number(6,0),
       s_w_id         number(5,0),
       s_quantity     number(6,0),
       s_dist_01      char(24),
       s_dist_02      char(24),
       s_dist_03      char(24),
       s_dist_04      char(24),
       s_dist_05      char(24),
       s_dist_06      char(24),
       s_dist_07      char(24),
       s_dist_08      char(24),
       s_dist_09      char(24),
       s_dist_10      char(24),
       s_ytd          number(10,0),
       s_order_cnt    number(6,0),
       s_remote_cnt   number(6,0),
       s_data         varchar2(50)
       )
       cluster scluster (s_i_id, s_w_id);

rem
rem done
rem

       exit;
```

C.3 Data Generation Code

Cust.ctl

```
--
-- $Header: cust.ctl 7030100.1 95/08/07 15:46:36 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ============================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                OPEN SYSTEMS PERFORMANCE GROUP                  |
--                   All Rights Reserved                          |
-- ============================================================+
--  FILENAME
--      cust.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading customers to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE customer
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
   c_id              integer external,
   c_d_id            integer external,
   c_w_id            integer external,
   c_first           char(16),
   c_middle          char(2),
   c_last            char(16),
   c_street_1        char(20),
   c_street_2        char(20),
   c_city            char(20),
   c_state           char(2),
   c_zip             char(9),
   c_phone           char(16),
   c_since           date,
   c_credit          char(2),
   c_credit_lim      int external,
   c_discount        int external,
   c_balance         integer external,
   c_ytd_payment     integer external,
   c_payment_cnt     integer external,
   c_delivery_cnt    integer external,
   c_data            char(500)
)
```

Hist_p1.ctl

```
--
-- $Header: hist.ctl 7030100.1 95/08/07 15:47:23 plai Generic<base> $ Copyr (c) 1994 Oracle
--
-- ============================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                OPEN SYSTEMS PERFORMANCE GROUP                  |
--                   All Rights Reserved                          |
-- ============================================================+
--  FILENAME
--      hist_p1.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading history rows to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE history PARTITION(hist_p01)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
   h_c_id            integer external,
   h_c_d_id          integer external,
   h_c_w_id          integer external,
   h_d_id            integer external,
   h_w_id            integer external,
   h_date            date,
   h_amount          integer external,
   h_data            char(24)
)
```

Hist_p2.ctl

```
--
-- $Header: hist_p2.ctl 7030100.1 95/08/07 15:47:23 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ============================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                OPEN SYSTEMS PERFORMANCE GROUP                  |
--                   All Rights Reserved                          |
-- ============================================================+
--  FILENAME
--      hist_p2.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading history rows to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)
```

```
UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE history PARTITION(hist_p02)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  h_c_id          integer external,
  h_c_d_id        integer external,
  h_c_w_id        integer external,
  h_d_id          integer external,
  h_w_id          integer external,
  h_date          date,
  h_amount        integer external,
  h_data          char(24)
)
```

Hist_p3.ctl

```
--
-- $Header: hist_p3.ctl 7030100.1 95/08/07 15:47:23 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ============================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                  OPEN SYSTEMS PERFORMANCE GROUP                |
--                     All Rights Reserved                        |
-- ============================================================+
-- FILENAME
--     hist_p3.ctl
-- DESCRIPTION
--     This is a SQL*Loader control file.  It is used for
--     loading history rows to the tpcc database.
-- USAGE
--     sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE history PARTITION(hist_p03)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  h_c_id          integer external,
  h_c_d_id        integer external,
  h_c_w_id        integer external,
  h_d_id          integer external,
  h_w_id          integer external,
  h_date          date,
  h_amount        integer external,
  h_data          char(24)
)
```

Hist_p4.ctl

```
--
-- $Header: hist_p4.ctl 7030100.1 95/08/07 15:47:23 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ============================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                  OPEN SYSTEMS PERFORMANCE GROUP                |
--                     All Rights Reserved                        |
-- ============================================================+
-- FILENAME
--     hist_p4.ctl
-- DESCRIPTION
--     This is a SQL*Loader control file.  It is used for
--     loading history rows to the tpcc database.
-- USAGE
--     sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE history PARTITION(hist_p04)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  h_c_id          integer external,
  h_c_d_id        integer external,
  h_c_w_id        integer external,
  h_d_id          integer external,
  h_w_id          integer external,
  h_date          date,
  h_amount        integer external,
  h_data          char(24)
)
```

Hist_p5.ctl

```
--
-- $Header: hist_p5.ctl 7030100.1 95/08/07 15:47:23 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ============================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                  OPEN SYSTEMS PERFORMANCE GROUP                |
--                     All Rights Reserved                        |
-- ============================================================+
-- FILENAME
--     hist_p5.ctl
-- DESCRIPTION
--     This is a SQL*Loader control file.  It is used for
--     loading history rows to the tpcc database.
-- USAGE
--     sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE history PARTITION(hist_p05)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  h_c_id          integer external,
  h_c_d_id        integer external,
  h_c_w_id        integer external,
  h_d_id          integer external,
  h_w_id          integer external,
  h_date          date,
  h_amount        integer external,
  h_data          char(24)
)
```

Nord_p1.ctl

```
--
-- $Header: nord_p1.ctl 7030100.1 95/08/07 15:47:44 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ============================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                  OPEN SYSTEMS PERFORMANCE GROUP                |
--                     All Rights Reserved                        |
-- ============================================================+
-- FILENAME
--     nord_p1.ctl
-- DESCRIPTION
--     This is a SQL*Loader control file.  It is used for
--     loading new orders to the tpcc database.
-- USAGE
--     sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- MODIFIED
--     G. Han, 06/97, Added partition.
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE new_order PARTITION (nord_p01)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  no_o_id         integer external,
  no_d_id         integer external,
  no_w_id         integer external
)
```

Nord_p2.ctl

```
--
-- $Header: nord_p02.ctl 7030100.1 95/08/07 15:47:44 plai Generic<base> $ Copyr (c) 1994
Oracle
```

```
--
-- ================================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
--                  OPEN SYSTEMS PERFORMANCE GROUP                 |
--                      All Rights Reserved                        |
-- ================================================================+
--  FILENAME
--      nord_p2.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading new orders to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
--  MODIFIED
--      G. Han, 06/97, Added partition.
-- ================================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE new_order partition (nord_p02)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  no_o_id           integer external,
  no_d_id           integer external,
  no_w_id           integer external
)
```

Nord_p3.ctl

```
--
-- $Header: nord_p3.ctl 7030100.1 95/08/07 15:47:44 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ================================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
--                  OPEN SYSTEMS PERFORMANCE GROUP                 |
--                      All Rights Reserved                        |
-- ================================================================+
--  FILENAME
--      nord_p3.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading new orders to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
--  MODIFIED
--      G. Han, 06/97, Added partition.
-- ================================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE new_order partition (nord_p03)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  no_o_id           integer external,
  no_d_id           integer external,
  no_w_id           integer external
)
```

Nord_p4.ctl

```
--
-- $Header: nord_p4.ctl 7030100.1 95/08/07 15:47:44 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ================================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
--                  OPEN SYSTEMS PERFORMANCE GROUP                 |
--                      All Rights Reserved                        |
-- ================================================================+
--  FILENAME
--      nord_p4.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading new orders to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
--  MODIFIED
--      G. Han, 06/97, Added partition.
-- ================================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE new_order partition (nord_p04)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  no_o_id           integer external,
  no_d_id           integer external,
  no_w_id           integer external
)
```

Nord_p5.ctl

```
--
-- $Header: neword.ctl 7030100.1 95/08/07 15:47:44 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ================================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
--                  OPEN SYSTEMS PERFORMANCE GROUP                 |
--                      All Rights Reserved                        |
-- ================================================================+
--  FILENAME
--      nord_p5.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading new orders to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
--  MODIFIED
--      G. Han, 06/97, Added partition.
-- ================================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE new_order PARTITION (nord_p05)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  no_o_id           integer external,
  no_d_id           integer external,
  no_w_id           integer external
)
```

Order_p1.ctl

```
--
-- $Header: order_p1.ctl 7030100.1 95/08/07 15:54:01 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ================================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
--                  OPEN SYSTEMS PERFORMANCE GROUP                 |
--                      All Rights Reserved                        |
-- ================================================================+
--  FILENAME
--      order_p1.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading orders to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
--  MODIFIED
--      G. Han, 06/97, Added partition.
-- ================================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE orders PARTITION (ords_p01)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  o_id              integer external,
  o_d_id            integer external,
  o_w_id            integer external,
  o_c_id            integer external,
  o_entry_d         date,
  o_carrier_id      integer external,
  o_ol_cnt          integer external,
  o_all_local       integer external
)
```

Order_p2.ctl

```
--
-- $Header: order_p2.ctl 7030100.1 95/08/07 15:54:01 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- +-------------------------------------------------------------+
--      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                  OPEN SYSTEMS PERFORMANCE GROUP               |
--                      All Rights Reserved                      |
-- +-------------------------------------------------------------+
-- FILENAME
--     order_p2.ctl
-- DESCRIPTION
--     This is a SQL*Loader control file.  It is used for
--     loading orders to the tpcc database.
-- USAGE
--     sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- MODIFIED
--     G. Han, 06/97, Added partition.
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE orders PARTITION(ords_p01)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  o_id              integer external,
  o_d_id            integer external,
  o_w_id            integer external,
  o_c_id            integer external,
  o_entry_d         date,
  o_carrier_id      integer external,
  o_ol_cnt          integer external,
  o_all_local       integer external
)
```

Order_p3.ctl

```
--
-- $Header: order_p3.ctl 7030100.1 95/08/07 15:54:01 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- +-------------------------------------------------------------+
--      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                  OPEN SYSTEMS PERFORMANCE GROUP               |
--                      All Rights Reserved                      |
-- +-------------------------------------------------------------+
-- FILENAME
--     order_p3.ctl
-- DESCRIPTION
--     This is a SQL*Loader control file.  It is used for
--     loading orders to the tpcc database.
-- USAGE
--     sqlldr[st] <user_name>/<password>  <SQL*Loader control file>
-- MODIFIED
--     G. Han, 06/97, Added partition.
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE orders PARTITION(ords_p02)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  o_id              integer external,
  o_d_id            integer external,
  o_w_id            integer external,
  o_c_id            integer external,
  o_entry_d         date,
  o_carrier_id      integer external,
  o_ol_cnt          integer external,
  o_all_local       integer external
)
```

Order_p4.ctl

```
--
-- $Header: order_p4.ctl 7030100.1 95/08/07 15:54:01 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- +-------------------------------------------------------------+
--      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                  OPEN SYSTEMS PERFORMANCE GROUP               |
--                      All Rights Reserved                      |
-- +-------------------------------------------------------------+
-- FILENAME
--     order_p4.ctl
-- DESCRIPTION
--     This is a SQL*Loader control file.  It is used for
--     loading orders to the tpcc database.
-- USAGE
--     sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- MODIFIED
--     G. Han, 06/97, Added partition.
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE orders PARTITION(ords_p02)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  o_id              integer external,
  o_d_id            integer external,
  o_w_id            integer external,
  o_c_id            integer external,
  o_entry_d         date,
  o_carrier_id      integer external,
  o_ol_cnt          integer external,
  o_all_local       integer external
)
```

Order_p5.ctl

```
--
-- $Header: order_p5.ctl 7030100.1 95/08/07 15:54:01 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- +-------------------------------------------------------------+
--      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                  OPEN SYSTEMS PERFORMANCE GROUP               |
--                      All Rights Reserved                      |
-- +-------------------------------------------------------------+
-- FILENAME
--     order_p5.ctl
-- DESCRIPTION
--     This is a SQL*Loader control file.  It is used for
--     loading orders to the tpcc database.
-- USAGE
--     sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- MODIFIED
--     G. Han, 06/97, Added partition.
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE orders PARTITION(ords_p03)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  o_id              integer external,
  o_d_id            integer external,
  o_w_id            integer external,
  o_c_id            integer external,
  o_entry_d         date,
  o_carrier_id      integer external,
  o_ol_cnt          integer external,
  o_all_local       integer external
)
```

Orderline_p1.ctl

```
--
-- $Header: ordline_p1.ctl 7030100.1 95/08/07 15:54:11 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- +-------------------------------------------------------------+
--      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                  OPEN SYSTEMS PERFORMANCE GROUP               |
--                      All Rights Reserved                      |
-- +-------------------------------------------------------------+
-- FILENAME
--     ordline_p1.ctl
-- DESCRIPTION
--     This is a SQL*Loader control file.  It is used for
--     loading order lines to the tpcc database.
-- USAGE
--     sqlldr[st] <user_name>/<password> <SQL*Loader control file>
-- MODIFIED
--     G. Han, 07/97, Added partition.
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)
```

```
UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE order_line PARTITION (ordl_p01)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  ol_o_id           integer external,
  ol_d_id           integer external,
  ol_w_id           integer external,
  ol_number         integer external,
  ol_delivery_d     date "DD-Mon-YYYY",
  ol_i_id           integer external,
  ol_supply_w_id    integer external,
  ol_quantity       integer external,
  ol_amount         integer external,
  ol_dist_info      char(24)
)
```

Orderline_p2.ctl

```
--
-- $Header: ordline_p2.ctl 7030100.1 95/08/07 15:54:11 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ================================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
--                  OPEN SYSTEMS PERFORMANCE GROUP                 |
--                      All Rights Reserved                        |
-- ================================================================+
--  FILENAME
--      ordline_p2.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading order lines to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
--  MODIFIED
--      G. Han, 07/97, Added partition.
-- ================================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE order_line PARTITION (ordl_p01)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  ol_o_id           integer external,
  ol_d_id           integer external,
  ol_w_id           integer external,
  ol_number         integer external,
  ol_delivery_d     date "DD-Mon-YYYY",
  ol_i_id           integer external,
  ol_supply_w_id    integer external,
  ol_quantity       integer external,
  ol_amount         integer external,
  ol_dist_info      char(24)
)
```

Orderline_p3.ctl

```
--
-- $Header: ordline_p3.ctl 7030100.1 95/08/07 15:54:11 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ================================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
--                  OPEN SYSTEMS PERFORMANCE GROUP                 |
--                      All Rights Reserved                        |
-- ================================================================+
--  FILENAME
--      ordline_p3.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading order lines to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
--  MODIFIED
--      G. Han, 07/97, Added partition.
-- ================================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE order_line PARTITION (ordl_p02)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  ol_o_id           integer external,
  ol_d_id           integer external,
  ol_w_id           integer external,
  ol_number         integer external,
  ol_delivery_d     date "DD-Mon-YYYY",
  ol_i_id           integer external,
  ol_supply_w_id    integer external,
  ol_quantity       integer external,
  ol_amount         integer external,
  ol_dist_info      char(24)
)
```

Orderline_p4.ctl

```
--
-- $Header: ordline.ctl 7030100.1 95/08/07 15:54:11 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ================================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
--                  OPEN SYSTEMS PERFORMANCE GROUP                 |
--                      All Rights Reserved                        |
-- ================================================================+
--  FILENAME
--      ordline_p4.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading order lines to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
--  MODIFIED
--      G. Han, 07/97, Added partition.
-- ================================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE order_line PARTITION (ordl_p02)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  ol_o_id           integer external,
  ol_d_id           integer external,
  ol_w_id           integer external,
  ol_number         integer external,
  ol_delivery_d     date "DD-Mon-YYYY",
  ol_i_id           integer external,
  ol_supply_w_id    integer external,
  ol_quantity       integer external,
  ol_amount         integer external,
  ol_dist_info      char(24)
)
```

Orderline_p5.ctl

```
--
-- $Header: ordline_p5.ctl 7030100.1 95/08/07 15:54:11 plai Generic<base> $ Copyr (c) 1994
Oracle
--
-- ================================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
--                  OPEN SYSTEMS PERFORMANCE GROUP                 |
--                      All Rights Reserved                        |
-- ================================================================+
--  FILENAME
--      ordline_p5.ctl
--  DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading order lines to the tpcc database.
--  USAGE
--      sqlldr[st] <user_name>/<password> <SQL*Loader control file>
--  MODIFIED
--      G. Han, 07/97, Added partition.
-- ================================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE order_line PARTITION (ordl_p03)
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  ol_o_id           integer external,
  ol_d_id           integer external,
  ol_w_id           integer external,
  ol_number         integer external,
  ol_delivery_d     date "DD-Mon-YYYY",
  ol_i_id           integer external,
  ol_supply_w_id    integer external,
```

```
  ol_quantity       integer external,
  ol_amount         integer external,
  ol_dist_info      char(24)
)
```

tpccload.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: tpccload.c 7030100.1 96/05/13 16:20:36 plai Generic<base> $ Copyr (c) 1993
Oracle";
#endif /* RCSID */

/*=================================================================+
|      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA        |
|                OPEN SYSTEMS PERFORMANCE GROUP                   |
|                   All Rights Reserved                           |
 +=================================================================+
| FILENAME
|    tpccload.c
| DESCRIPTION
|    Load or generate TPC-C database tables.
|    Usage: tpccload -M <# of warehouses> [options]
|              options: -A  load all tables
|                       -w  load warehouse table
|                       -d  load district table
|                       -c  load customer table
|                       -i  load item table
|                       -s  load stock table (cluster around s_w_id)
|                       -S  load stock table (cluster around s_i_id)
|                       -h  load history table
|                       -n  load new-order table
|                       -o <oline file> load order and order-line table
|                       -b <ware#>  beginning warehouse number
|                       -e <ware#>  ending warehouse number
|                       -j <item#>  beginning item number (with -S)
|                       -k <item#>  ending item number (with -S)
|                       -g  generate rows to standard output
| DDM 09/14/97 - added CNUM1 value to w_name for w_id=1
 +=================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include "tpcc.h"

#define DISTARR 10                      /* district insert array size               */
#define CUSTARR 100                     /* customer insert array size               */
#define STOCARR 100                     /* stock insert array size  */
#define ITEMARR 100                     /* item insert array size   */
#define HISTARR 100             /* history insert array size   */
#define ORDEARR 100              /* order insert array size     */
#define NEWOARR 100              /* new order insert array size  */

#define DISTFAC 10                      /* max. disctrict id          */
#define CUSTFAC 3000                    /* max. customer id         */
#define STOCFAC 100000                          /* max. stock id              */
#define ITEMFAC 100000                  /* max. item id                             */
#define HISTFAC 30000           /* history / warehouse  */
#define ORDEFAC 3000             /* order / district     */
#define NEWOFAC 900              /* new order / district */

#define C       0               /* constant in non-uniform dist. eqt. */
#define CNUM1   1               /* first constant in non-uniform dist. eqt. */
#define CNUM2   2               /* second constant in non-uniform dist. eqt. */
#define CNUM3   3               /* third constant in non-uniform dist. eqt. */

#define SEED    2               /* seed for random functions */

#define SQLTXTW "INSERT INTO warehouse VALUES (:w_id, 30000000, :w_tax, :w_name,
:w_street_1, \
  :w_street_2, :w_city, :w_state, :w_zip)"

#define SQLTXTD "INSERT INTO district VALUES (:d_id, :d_w_id,3000000, :d_tax, \
    3001, :d_name, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip)"

#define SQLTXTC "INSERT INTO customer VALUES (:c_id, :c_d_id, :c_w_id, \
  :c_first, 'OE', :c_last, :c_street_1, :c_street_2, :c_city, :c_state, \
  :c_zip, :c_phone, SYSDATE, :c_credit, 5000000, :c_discount, -1000, 1000, 1, \
  0, :c_data)"

#define SQLTXTH "INSERT INTO history VALUES (:h_c_id, :h_c_d_id, :h_c_w_id, \
  :h_d_id, :h_w_id, SYSDATE, 1000, :h_data)"

#define SQLTXTS "INSERT INTO stock VALUES (:s_i_id, :s_w_id, :s_quantity, \
  :s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05, :s_dist_06, \
  :s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10, 0, 0, 0, :s_data)" \

#define SQLTXTI "INSERT INTO item VALUES (:i_id, :i_im_id, :i_name, :i_price, \
  :i_data)"

#define SQLTXTO1 "INSERT INTO orders VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
  SYSDATE, :o_carrier_id, :o_ol_cnt, 1)"

#define SQLTXTO2 "INSERT INTO orders VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
  SYSDATE, 11, :o_ol_cnt, 1)"

#define SQLTXTOL1 "INSERT INTO order_line VALUES (:ol_o_id, :ol_d_id, \
  :ol_w_id, :ol_number, SYSDATE, :ol_i_id, :ol_supply_w_id, 5, 0, \
  :ol_dist_info)"

#define SQLTXTOL2 "INSERT INTO order_line VALUES (:ol_o_id, :ol_d_id, \
  :ol_w_id, :ol_number, to_date('01-Jan-1811'), :ol_i_id, :ol_supply_w_id, 5, :ol_amount, \
  :ol_dist_info)"

#define SQLTXTNO "INSERT INTO new_order VALUES (:no_o_id, :no_d_id, :no_w_id)"

ldadef tpclda;
csrdef curw, curd, curc, curh, curs, curi, curo1, curo2, curol1, curol2, curno;
unsigned long tpchda[256];

static char *lastname[] = {
  "BAR",
  "OUGHT",
  "ABLE",
  "PRI",
  "PRES",
  "ESE",
  "ANTI",
  "CALLY",
  "ATION",
  "EING"
};

char num9[10];
char num16[17];
char str2[3];
char str24[15][25];
int randperm3000[3000];



myusage()

{

  fprintf (stderr, "\n");
  fprintf (stderr, "Usage:\ttpccload -M <multiplier> [options]\n");
  fprintf (stderr, "options:\n");
  fprintf (stderr, "\t-A :\tload all tables\n");
  fprintf (stderr, "\t-w :\tload warehouse table\n");
  fprintf (stderr, "\t-d :\tload district table\n");
  fprintf (stderr, "\t-c :\tload customer table\n");
  fprintf (stderr, "\t-i :\tload item table\n");
  fprintf (stderr, "\t-s :\tload stock table (cluster around s_w_id)\n");
  fprintf (stderr, "\t-S :\tload stock table (cluster around s_i_id)\n");
  fprintf (stderr, "\t-h :\tload history table\n");
  fprintf (stderr, "\t-n :\tload new-order table\n");
  fprintf (stderr, "\t-o <oline file> :\tload order and order-line table\n");
  fprintf (stderr, "\t-b <ware#> :\tbeginning warehouse number\n");
  fprintf (stderr, "\t-e <ware#> :\tending warehouse number\n");
  fprintf (stderr, "\t-j <item#> :\tbeginning item number (with-S)\n");
  fprintf (stderr, "\t-k <item#> :\tending item number (with-S)\n");
  fprintf (stderr, "\t-g :\tgenerate rows to standard output\n");
  fprintf (stderr, "\n");
  exit(1);

}



errrpt (lda, cur)

csrdef *lda;
csrdef *cur;

{

  text msg[2048];

  if (cur->rc) {
    oerhms (lda, cur->rc, msg, 2048);
    fprintf (stderr, "TPC-C load error: %s\n",msg);
  }

}



quit ()

{

  if (oclose (&curw))
    errrpt (&tpclda, &curw);

  if (oclose (&curd))
```

```
    errrpt (&tpclda, &curd);

  if (oclose (&curc))
    errrpt (&tpclda, &curc);

  if (oclose (&curh))
    errrpt (&tpclda, &curh);

  if (oclose (&curs))
    errrpt (&tpclda, &curs);

  if (oclose (&curi))
    errrpt (&tpclda, &curi);

  if (oclose (&curo1))
    errrpt (&tpclda, &curo1);

  if (oclose (&curo2))
    errrpt (&tpclda, &curo2);

  if (oclose (&curol1))
    errrpt (&tpclda, &curol1);

  if (oclose (&curol2))
    errrpt (&tpclda, &curol2);

  if (oclose (&curno))
    errrpt (&tpclda, &curno);

  if (ologof (&tpclda))
    fprintf (stderr, "TPC-C load error: Error in logging off\n");

}



main (argc, argv)

int argc;
char *argv[];

{

  char *uid="tpcc/tpcc";
  text sqlbuf[1024];
  int scale=0;
  int i, j;
  int loop;
  int loopcount;
  int cid;
  int dwid;
  int cdid;
  int cwid;
  int sid;
  int swid;
  int olcnt;
  int nrows;
  int row;

  int w_id;
  char w_name[11];
  char w_street_1[21];
  char w_street_2[21];
  char w_city[21];
  char w_state[2];
  char w_zip[9];
  int w_tax;

  int d_id[10];
  int d_w_id[10];
  char d_name[10][11];
  char d_street_1[10][21];
  char d_street_2[10][21];
  char d_city[10][21];
  char d_state[10][2];
  char d_zip[10][9];
  int d_tax[10];

  int c_id[100];
  int c_d_id[100];
  int c_w_id[100];
  char c_first[100][17];
  char c_last[100][17];
  char c_street_1[100][21];
  char c_street_2[100][21];
  char c_city[100][21];
  char c_state[100][2];
  char c_zip[100][9];
  char c_phone[100][16];
  char c_credit[100][2];
  int c_discount[100];
  char c_data[100][501];

  int i_id[100];
  int i_im_id[100];
  int i_price[100];
  char i_name[100][25];
  char i_data[100][51];

  int s_i_id[100];
  int s_w_id[100];
  int s_quantity[100];
  char s_dist_01[100][24];
  char s_dist_02[100][24];
  char s_dist_03[100][24];
  char s_dist_04[100][24];
  char s_dist_05[100][24];
  char s_dist_06[100][24];
  char s_dist_07[100][24];
  char s_dist_08[100][24];
  char s_dist_09[100][24];
  char s_dist_10[100][24];
  char s_data[100][51];

  int h_w_id[100];
  int h_d_id[100];
  int h_c_id[100];
  char h_data[100][25];

  int o_id[100];
  int o_d_id[100];
  int o_w_id[100];
  int o_c_id[100];
  int o_carrier_id[100];
  int o_ol_cnt[100];

  int ol_o_id[15];
  int ol_d_id[15];
  int ol_w_id[15];
  int ol_number[15];
  int ol_i_id[15];
  int ol_supply_w_id[15];
  int ol_amount[15];
  char ol_dist_info[15][24];

  int no_o_id[100];
  int no_d_id[100];
  int no_w_id[100];

  char sdate[30];

  double begin_time, end_time;
  double begin_cpu, end_cpu;
  double gettime(), getcpu();

  extern int getopt();
  extern char *optarg;
  extern int optind, opterr;

  char        *argstr="M:AwdcisShno:b:e:j:k:g";
  int opt;
  int do_A=0;
  int do_w=0;
  int do_d=0;
  int do_i=0;
  int do_c=0;
  int do_s=0;
  int do_S=0;
  int do_h=0;
  int do_o=0;
  int do_n=0;
  int gen=0;
  int bware=1;
  int eware=0;
  int bitem=1;
  int eitem=0;

  FILE *olfp=NULL;
  char olfname[100];

/*---------------------------------------------------------------+
 | Parse command line -- look for scale factor.                  |
 +---------------------------------------------------------------*/

  if (argc == 1) {
    myusage ();
  }

  while ((opt = getopt (argc, argv, argstr)) != -1) {
    switch (opt) {
      case '?': myusage ();
                break;
      case 'M': scale = atoi (optarg);
                break;
      case 'A': do_A = 1;
                break;
      case 'w': do_w = 1;
                break;
      case 'd': do_d = 1;
                break;
```

```
      case 'c': do_c = 1;
                break;
      case 'i': do_i = 1;
                break;
      case 's': do_s = 1;
                break;
      case 'S': do_S = 1;
                break;
      case 'h': do_h = 1;
                break;
      case 'n': do_n = 1;
                break;
      case 'o': do_o = 1;
                strcpy (olfname, optarg);
                break;
      case 'b': bware = atoi (optarg);
                break;
      case 'e': eware = atoi (optarg);
                break;
      case 'j': bitem = atoi (optarg);
                break;
      case 'k': eitem = atoi (optarg);
                break;
      case 'g': gen = 1;
                break;
      default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
               fprintf (stderr, "(reached default case in getopt ())\n");
               myusage ();
    }
  }

/*---------------------------------------------------------------*|
|                 Rudimentary error checking                    |
|*---------------------------------------------------------------*/

  if (scale  < 1) {
    fprintf (stderr, "Invalid scale factor: '%d'\n", scale);
    myusage ();
  }

  if (!(do_A || do_w || do_d || do_c || do_i || do_s || do_S || do_h || do_o ||
        do_n)) {
    fprintf (stderr, "What should I load???\n");
    myusage ();
  }

  if (gen && (do_A || (do_w + do_d + do_c + do_i + do_s + do_S + do_h + do_o +
                       do_n > 1))) {
    fprintf (stderr, "Can only generate table one at a time\n");
    myusage ();
  }

  if (do_S && (do_A || do_s)) {
    fprintf (stderr, "Cluster stock table around s_w_id or s_i_id?\n");
    myusage ();
  }

  if (eware <= 0)
    eware = scale;
  if (eitem <= 0)
    eitem = STOCFAC;

  if (do_S) {
    if ((bitem  < 1) || (bitem > STOCFAC)) {
      fprintf (stderr, "Invalid beginning item number: '%d'\n", bitem);
      myusage ();
    }

    if ((eitem  < bitem) || (eitem > STOCFAC)) {
      fprintf (stderr, "Invalid ending item number: '%d'\n", eitem);
      myusage ();
    }
  }

  if ((bware  < 1) || (bware > scale)) {
    fprintf (stderr, "Invalid beginning warehouse number: '%d'\n", bware);
    myusage ();
  }

  if ((eware  < bware) || (eware > scale)) {
    fprintf (stderr, "Invalid ending warehouse number: '%d'\n", eware);
    myusage ();
  }

  if (gen && do_o) {
    if ((olfp = fopen (olfname, "w")) == NULL) {
      fprintf (stderr, "Can't open '%s' for writing orderlines\n", olfname);
      myusage ();
    }
  }

/*---------------------------------------------------------------+
 | Prepare to insert into database.                              |
 +---------------------------------------------------------------*/
```

```
  sysdate (sdate);
  if (!gen) {

    /* log on to Oracle */

    if (orlon (&tpclda, (ub1 *) tpchda, (text *) uid, -1, (text *) 0, -1, 0)) {
      fprintf (stderr, "TPC-C load error: Error in logging on\n");
      errrpt (&tpclda, &tpclda);
      exit (1);
    }

    fprintf (stderr, "\nConnected to Oracle userid '%s'.\n", uid);

    /* turn off auto-commit*/

    if (ocof (&tpclda)) {
      errrpt (&tpclda, &tpclda);
      ologof (&tpclda);
      exit (1);
    }

    /* open cursors */

    if (oopen (&curw, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
      errrpt (&tpclda, &curw);
      ologof (&tpclda);
      exit (1);
    }

    if (oopen (&curd, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
      errrpt (&tpclda, &curd);
      oclose (&curw);
      ologof (&tpclda);
      exit (1);
    }

    if (oopen (&curc, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
      errrpt (&tpclda, &curc);
      oclose (&curw);
      oclose (&curd);
      ologof (&tpclda);
      exit (1);
    }

    if (oopen (&curh, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
      errrpt (&tpclda, &curh);
      oclose (&curw);
      oclose (&curd);
      oclose (&curc);
      ologof (&tpclda);
      exit (1);
    }

    if (oopen (&curs, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
      errrpt (&tpclda, &curs);
      oclose (&curw);
      oclose (&curd);
      oclose (&curc);
      oclose (&curh);
      ologof (&tpclda);
      exit (1);
    }

    if (oopen (&curi, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
      errrpt (&tpclda, &curi);
      oclose (&curw);
      oclose (&curd);
      oclose (&curc);
      oclose (&curh);
      oclose (&curs);
      ologof (&tpclda);
      exit (1);
    }

    if (oopen (&curo1, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
      errrpt (&tpclda, &curo1);
      oclose (&curw);
      oclose (&curd);
      oclose (&curc);
      oclose (&curh);
      oclose (&curs);
      oclose (&curi);
      ologof (&tpclda);
      exit (1);
    }

    if (oopen (&curo2, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
      errrpt (&tpclda, &curo2);
      oclose (&curw);
      oclose (&curd);
      oclose (&curc);
      oclose (&curh);
      oclose (&curs);
      oclose (&curi);
      oclose (&curo1);
```

```
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curol1, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curol1);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curo1);
    oclose (&curo2);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curol2, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curol2);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curo1);
    oclose (&curo2);
    oclose (&curol1);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curno, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curno);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curo1);
    oclose (&curo2);
    oclose (&curol1);
    oclose (&curol2);
    ologof (&tpclda);
    exit (1);
  }

  /* parse statements */

  sprintf ((char *) sqlbuf, SQLTXTW);
  if (oparse (&curw, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTD);
  if (oparse (&curd, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTC);
  if (oparse (&curc, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTH);
  if (oparse (&curh, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curh);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTS);
  if (oparse (&curs, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTI);
  if (oparse (&curi, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTO1);
  if (oparse (&curo1, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curo1);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTO2);
  if (oparse (&curo2, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curo2);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTOL1);
  if (oparse (&curol1, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curol1);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTOL2);
  if (oparse (&curol2, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curol2);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTNO);
  if (oparse (&curno, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curno);
    quit ();
    exit (1);
  }

  /* bind variables */

  /* warehouse */

  if (obndrv (&curw, (text *) ":w_id", -1, (ub1 *) &w_id, sizeof (w_id),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  if (obndrv (&curw, (text *) ":w_name", -1, (ub1 *) w_name, 11,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  if (obndrv (&curw, (text *) ":w_street_1", -1, (ub1 *) w_street_1, 21,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  if (obndrv (&curw, (text *) ":w_street_2", -1, (ub1 *) w_street_2, 21,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  if (obndrv (&curw, (text *) ":w_city", -1, (ub1 *) w_city, 21,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  if (obndrv (&curw, (text *) ":w_state", -1, (ub1 *) w_state, 2,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  if (obndrv (&curw, (text *) ":w_zip", -1, (ub1 *) w_zip, 9,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  if (obndrv (&curw, (text *) ":w_tax", -1, (ub1 *) &w_tax, sizeof (w_tax),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  /* district */
```

```
  if (obndrv (&curd, (text *) ":d_id", -1, (ub1 *) d_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  if (obndrv (&curd, (text *) ":d_w_id", -1, (ub1 *) d_w_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  if (obndrv (&curd, (text *) ":d_name", -1, (ub1 *) d_name, 11,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  if (obndrv (&curd, (text *) ":d_street_1", -1, (ub1 *) d_street_1, 21,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  if (obndrv (&curd, (text *) ":d_street_2", -1, (ub1 *) d_street_2, 21,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  if (obndrv (&curd, (text *) ":d_city", -1, (ub1 *) d_city, 21,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  if (obndrv (&curd, (text *) ":d_state", -1, (ub1 *) d_state, 2,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  if (obndrv (&curd, (text *) ":d_zip", -1, (ub1 *) d_zip, 9,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  if (obndrv (&curd, (text *) ":d_tax", -1, (ub1 *) d_tax, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  /* customer */

  if (obndrv (&curc, (text *) ":c_id", -1, (ub1 *) c_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_d_id", -1, (ub1 *) c_d_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_w_id", -1, (ub1 *) c_w_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_first", -1, (ub1 *) c_first, 17,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_last", -1, (ub1 *) c_last, 17,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_street_1", -1, (ub1 *) c_street_1, 21,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_street_2", -1, (ub1 *) c_street_2, 21,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_city", -1, (ub1 *) c_city, 21,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_state", -1, (ub1 *) c_state, 2,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_zip", -1, (ub1 *) c_zip, 9,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_phone", -1, (ub1 *) c_phone, 16,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_credit", -1, (ub1 *) c_credit, 2,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_discount", -1, (ub1 *) c_discount,
          sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1,
          -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  if (obndrv (&curc, (text *) ":c_data", -1, (ub1 *) c_data, 501,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  /* item */

  if (obndrv (&curi, (text *) ":i_id", -1, (ub1 *) i_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
  }

  if (obndrv (&curi, (text *) ":i_im_id", -1, (ub1 *) i_im_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
  }

  if (obndrv (&curi, (text *) ":i_name", -1, (ub1 *) i_name, 25,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
  }

  if (obndrv (&curi, (text *) ":i_price", -1, (ub1 *) i_price,
          sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1,
          -1)) {
```

```
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
  }

  if (obndrv (&curi, (text *) ":i_data", -1, (ub1 *) i_data, 51,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
  }

  /* stock */

  if (obndrv (&curs, (text *) ":s_i_id", -1, (ub1 *) s_i_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_w_id", -1, (ub1 *) s_w_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_quantity", -1, (ub1 *) s_quantity,
          sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_01", -1, (ub1 *) s_dist_01, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_02", -1, (ub1 *) s_dist_02, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_03", -1, (ub1 *) s_dist_03, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_04", -1, (ub1 *) s_dist_04, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_05", -1, (ub1 *) s_dist_05, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_06", -1, (ub1 *) s_dist_06, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_07", -1, (ub1 *) s_dist_07, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_08", -1, (ub1 *) s_dist_08, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_09", -1, (ub1 *) s_dist_09, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_dist_10", -1, (ub1 *) s_dist_10, 24,
          SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  if (obndrv (&curs, (text *) ":s_data", -1, (ub1 *) s_data, 51,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  /* history */

  if (obndrv (&curh, (text *) ":h_c_id", -1, (ub1 *) h_c_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curh);
    quit ();
    exit (1);
  }

  if (obndrv (&curh, (text *) ":h_c_d_id", -1, (ub1 *) h_d_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curh);
    quit ();
    exit (1);
  }

  if (obndrv (&curh, (text *) ":h_c_w_id", -1, (ub1 *) h_w_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curh);
    quit ();
    exit (1);
  }

  if (obndrv (&curh, (text *) ":h_d_id", -1, (ub1 *) h_d_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curh);
    quit ();
    exit (1);
  }

  if (obndrv (&curh, (text *) ":h_w_id", -1, (ub1 *) h_w_id, sizeof (int),
          SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curh);
    quit ();
    exit (1);
  }

  if (obndrv (&curh, (text *) ":h_data", -1, (ub1 *) h_data, 25,
          SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curh);
    quit ();
    exit (1);
  }

  /* order_line (delivered) */

  if (obndrv (&curol1, (text *) ":ol_o_id", -1, (ub1 *) ol_o_id,
          sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol1);
    quit ();
    exit (1);
  }

  if (obndrv (&curol1, (text *) ":ol_d_id", -1, (ub1 *) ol_d_id,
          sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol1);
    quit ();
    exit (1);
  }

  if (obndrv (&curol1, (text *) ":ol_w_id", -1, (ub1 *) ol_w_id,
          sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol1);
    quit ();
    exit (1);
  }

  if (obndrv (&curol1, (text *) ":ol_number", -1, (ub1 *) ol_number,
      sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol1);
    quit ();
    exit (1);
  }

  if (obndrv (&curol1, (text *) ":ol_i_id", -1, (ub1 *) ol_i_id,
      sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol1);
    quit ();
```

```
    exit (1);
  }

  if (obndrv (&curol1, (text *) ":ol_supply_w_id", -1,
            (ub1 *) ol_supply_w_id, sizeof (int), SQLT_INT, -1,
            (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol1);
    quit ();
    exit (1);
  }

  if (obndrv (&curol1, (text *) ":ol_dist_info", -1, (ub1 *) ol_dist_info,
            24, SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol1);
    quit ();
    exit (1);
  }

  /* order_line (not delivered) */

  if (obndrv (&curol2, (text *) ":ol_o_id", -1, (ub1 *) ol_o_id,
            sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol2);
    quit ();
    exit (1);
  }

  if (obndrv (&curol2, (text *) ":ol_d_id", -1, (ub1 *) ol_d_id,
            sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol2);
    quit ();
    exit (1);
  }

  if (obndrv (&curol2, (text *) ":ol_w_id", -1, (ub1 *) ol_w_id,
            sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol2);
    quit ();
    exit (1);
  }

  if (obndrv (&curol2, (text *) ":ol_number", -1, (ub1 *) ol_number,
      sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol2);
    quit ();
    exit (1);
  }

  if (obndrv (&curol2, (text *) ":ol_i_id", -1, (ub1 *) ol_i_id,
      sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol2);
    quit ();
    exit (1);
  }

  if (obndrv (&curol2, (text *) ":ol_supply_w_id", -1,
            (ub1 *) ol_supply_w_id, sizeof (int), SQLT_INT, -1,
            (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol2);
    quit ();
    exit (1);
  }

  if (obndrv (&curol2, (text *) ":ol_amount", -1, (ub1 *) ol_amount,
      sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol2);
    quit ();
    exit (1);
  }

  if (obndrv (&curol2, (text *) ":ol_dist_info", -1, (ub1 *) ol_dist_info,
            24, SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curol2);
    quit ();
    exit (1);
  }

  /* orders (delivered) */

  if (obndrv (&curo1, (text *) ":o_id", -1, (ub1 *) o_id, sizeof (int),
            SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo1);
    quit ();
    exit (1);
  }

  if (obndrv (&curo1, (text *) ":o_d_id", -1, (ub1 *) o_d_id, sizeof (int),
            SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo1);
    quit ();
    exit (1);
  }

  if (obndrv (&curo1, (text *) ":o_w_id", -1, (ub1 *) o_w_id, sizeof (int),
            SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo1);
    quit ();
    exit (1);
  }

  if (obndrv (&curo1, (text *) ":o_c_id", -1, (ub1 *) o_c_id, sizeof (int),
            SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo1);
    quit ();
    exit (1);
  }

  if (obndrv (&curo1, (text *) ":o_carrier_id", -1, (ub1 *) o_carrier_id,
      sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo1);
    quit ();
    exit (1);
  }

  if (obndrv (&curo1, (text *) ":o_ol_cnt", -1, (ub1 *) o_ol_cnt,
      sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo1);
    quit ();
    exit (1);
  }

  /* orders (not delivered) */

  if (obndrv (&curo2, (text *) ":o_id", -1, (ub1 *) o_id, sizeof (int),
            SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo2);
    quit ();
    exit (1);
  }

  if (obndrv (&curo2, (text *) ":o_d_id", -1, (ub1 *) o_d_id, sizeof (int),
            SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo2);
    quit ();
    exit (1);
  }

  if (obndrv (&curo2, (text *) ":o_w_id", -1, (ub1 *) o_w_id, sizeof (int),
            SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo2);
    quit ();
    exit (1);
  }

  if (obndrv (&curo2, (text *) ":o_c_id", -1, (ub1 *) o_c_id, sizeof (int),
            SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo2);
    quit ();
    exit (1);
  }

  if (obndrv (&curo2, (text *) ":o_ol_cnt", -1, (ub1 *) o_ol_cnt,
            sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curo2);
    quit ();
    exit (1);
  }

  /* new order */

  if (obndrv (&curno, (text *) ":no_o_id", -1, (ub1 *) no_o_id,
            sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curno);
    quit ();
    exit (1);
  }

  if (obndrv (&curno, (text *) ":no_d_id", -1, (ub1 *) no_d_id,
            sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curno);
    quit ();
    exit (1);
  }

  if (obndrv (&curno, (text *) ":no_w_id", -1, (ub1 *) no_w_id,
            sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curno);
    quit ();
    exit (1);
  }
 }

/*---------------------------------------------------------------+
 | Initialize random number generator                            |
 +---------------------------------------------------------------*/

  srand (getpid ());
  srand48 (getpid ());
  initperm ();
```

```
/*---------------------------------------------------------------+
| Load the WAREHOUSE table.                                      |
+---------------------------------------------------------------*/

  if (do_A || do_w) {
    nrows = eware - bware + 1;

    fprintf (stderr, "Loading/generating warehouse: w%d - w%d (%d rows)\n",
             bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    for (loop = bware; loop <= eware; loop++) {

      w_tax = (rand () % 2001);
      randstr (w_name, 6, 10);
      randstr (w_street_1, 10, 20);
      randstr (w_street_2, 10, 20);
      randstr (w_city, 10, 20);
      randstr (str2, 2, 2);
      randnum (num9, 9);
      num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

      /*---------------------------------------------------------*/
      /* Following block per Auditors request 09/14/97 DDM       */
      /*---------------------------------------------------------*/
      if (w_id==1) {
        sprintf(w_name,"C_LAST=%03d",CNUM1);
      }
      /*---------------------------------------------------------*/

      if (gen) {
        printf ("%d 30000000 %d %s %s %s %s %s\n", loop, w_tax,
                w_name, w_street_1, w_street_2, w_city, str2, num9);
        fflush (stdout);
      }
      else {
        w_id = loop;
        strncpy (w_state, str2, 2);
        strncpy (w_zip, num9, 9);

        if (oexec (&curw)) {
          errrpt (&tpclda, &curw);
          orol (&tpclda);
          fprintf (stderr, "Aborted at warehouse %d\n", loop);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at warehouse %d\n", loop);
          quit ();
          exit (1);
        }
      }
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
             nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
| Load the DISTRICT table.                                       |
+---------------------------------------------------------------*/

  if (do_A || do_d) {
    nrows = (eware - bware + 1) * DISTFAC;

    fprintf (stderr, "Loading/generating district: w%d - w%d (%d rows)\n",
             bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    dwid = bware - 1;

    for (row = 0; row < nrows; ) {
      dwid++;

      for (i = 0; i < DISTARR; i++, row++) {
        d_tax[i] = (rand () % 2001);
        randstr (d_name[i], 6, 10);
        randstr (d_street_1[i], 10, 20);
        randstr (d_street_2[i], 10, 20);
        randstr (d_city[i], 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

        if (gen) {
          /* printf ("%d %d %s %s %s %s %s %s %d 30000.0 3001\n",
                i + 1, dwid, d_name[i], d_street_1[i], d_street_2[i],
                d_city[i], str2, num9, d_tax[i]); */
                              /* Reordered columns */
          printf ("%d %d 3000000 %d 3001 %s %s %s %s %s %s\n",
                i + 1, dwid, d_tax[i], d_name[i], d_street_1[i],
                              d_street_2[i], d_city[i], str2, num9 );
        }
        else {
          d_id[i] = i + 1;
          d_w_id[i] = dwid;
          strncpy (d_state[i], str2, 2);
          strncpy (d_zip[i], num9, 9);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curd, DISTARR, 0)) {
          errrpt (&tpclda, &curd);
          orol (&tpclda);
          fprintf (stderr, "Aborted at warehouse %d, district 1\n",dwid);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at warehouse %d, district 1\n",dwid);
          quit ();
          exit (1);
        }
      }
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
             nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
| Load the CUSTOMER table.                                       |
+---------------------------------------------------------------*/

  if (do_A || do_c) {
    nrows = (eware - bware + 1) * CUSTFAC * DISTFAC;

    fprintf (stderr, "Loading/generating customer: w%d - w%d (%d rows)\n  ",
             bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < CUSTARR; i++, row++) {
        cid++;
        if (cid > CUSTFAC) {       /* cycle cust id */
          cid = 1;                 /* cheap mod */
          cdid++;                  /* shift district cycle */
          if (cdid > DISTFAC) {
            cdid = 1;
            cwid++;                /* shift warehouse cycle */
          }
        }
        c_id[i] = cid;
        c_d_id[i] = cdid;
        c_w_id[i] = cwid;
        if (cid <= 1000)
          randlastname (c_last[i], cid - 1);
        else
          randlastname (c_last[i], NURand (255, 0, 999, CNUM1));
        c_credit[i][1] = 'C';
        if (rand () % 10)
          c_credit[i][0] = 'G';
        else
          c_credit[i][0] = 'B';
        c_discount[i] = (rand () % 5001);
        randstr (c_first[i], 8, 16);
        randstr (c_street_1[i], 10, 20);
        randstr (c_street_2[i], 10, 20);
        randstr (c_city[i], 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
        randnum (num16, 16);
        randstr (c_data[i], 300, 500);
```

```
        if (gen) {
          printf ("%d %d %d %s OE %s %s %s %s %s %s %s %s C 5000000 %d -1000 1000
1 0 %s\n",
                  cid, cdid, cwid, c_first[i], c_last[i],
                  c_street_1[i], c_street_2[i], c_city[i], str2, num9,
                  num16, sdate, c_credit[i][0], c_discount[i], c_data[i]);
        }
        else {
          strncpy (c_state[i], str2, 2);
          strncpy (c_zip[i], num9, 9);
          strncpy (c_phone[i], num16, 16);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curc, CUSTARR, 0)) {
          errrpt (&tpclda, &curc);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                c_w_id[0], c_d_id[0], c_id[0]);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                c_w_id[0], c_d_id[0], c_id[0]);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
  }


/*---------------------------------------------------------------+
| Load the ITEM table.                                           |
+---------------------------------------------------------------*/

  if (do_A || do_i) {
    nrows = ITEMFAC;

    fprintf (stderr, "Loading/generating item: (%d rows)\n   ", nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < ITEMARR; i++, row++) {
        i_im_id[i] = (rand () % 10000) + 1;
        i_price[i] = ((rand () % 9901) + 100);
        randstr (i_name[i], 14, 24);
        randdatastr (i_data[i], 26, 50);

        if (gen) {
          printf ("%d %d %s %d %s\n", row + 1, i_im_id[i], i_name[i],
                i_price[i], i_data[i]);
        }
        else {
          i_id[i] = row + 1;
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curi, ITEMARR, 0)) {
          errrpt (&tpclda, &curi);
          orol (&tpclda);
          fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
  }


/*---------------------------------------------------------------+
| Load the STOCK table.                                          |
+---------------------------------------------------------------*/

  if (do_A || do_s) {

    nrows = (eware - bware + 1) * STOCFAC;

    fprintf (stderr, "Loading/generating stock: w%d - w%d (%d rows)\n   ",
          bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    sid = 0;
    swid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < STOCARR; i++, row++) {
        if (++sid > STOCFAC) {       /* cheap mod */
          sid = 1;
          swid++;
        }
        s_quantity[i] = (rand () % 91) + 10;
        randstr (str24[0], 24, 24);
        randstr (str24[1], 24, 24);
        randstr (str24[2], 24, 24);
        randstr (str24[3], 24, 24);
        randstr (str24[4], 24, 24);
        randstr (str24[5], 24, 24);
        randstr (str24[6], 24, 24);
        randstr (str24[7], 24, 24);
        randstr (str24[8], 24, 24);
        randstr (str24[9], 24, 24);
        randdatastr (s_data[i], 26, 50);

        if (gen) {
          printf ("%d %d %d %s %s %s %s %s %s %s %s %s %d %s\n",
                sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
                str24[3], str24[4], str24[5], str24[6], str24[7],
                str24[8], str24[9], s_data[i]);
        }
        else {
          s_i_id[i] = sid;
          s_w_id[i] = swid;
          strncpy (s_dist_01[i], str24[0], 24);
          strncpy (s_dist_02[i], str24[1], 24);
          strncpy (s_dist_03[i], str24[2], 24);
          strncpy (s_dist_04[i], str24[3], 24);
          strncpy (s_dist_05[i], str24[4], 24);
          strncpy (s_dist_06[i], str24[5], 24);
          strncpy (s_dist_07[i], str24[6], 24);
          strncpy (s_dist_08[i], str24[7], 24);
          strncpy (s_dist_09[i], str24[8], 24);
          strncpy (s_dist_10[i], str24[9], 24);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curs, STOCARR, 0)) {
          errrpt (&tpclda, &curs);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
          quit ();
          exit (1);
```

```
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the STOCK table (cluster around s_i_id).                 |
 +---------------------------------------------------------------*/

  if (do_S) {

    nrows = (eitem - bitem + 1) * (eware - bware + 1);

    fprintf (stderr, "Loading/generating stock: i%d - i%d, w%d - w%d (%d rows)\n   ",
          bitem, eitem, bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    sid = bitem;
    swid = bware - 1;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < STOCARR; i++, row++) {
        if (++swid > eware) {          /* cheap mod */
          swid = bware;
          sid++;
        }
        s_quantity[i] = (rand () % 91) + 10;
        randstr (str24[0], 24, 24);
        randstr (str24[1], 24, 24);
        randstr (str24[2], 24, 24);
        randstr (str24[3], 24, 24);
        randstr (str24[4], 24, 24);
        randstr (str24[5], 24, 24);
        randstr (str24[6], 24, 24);
        randstr (str24[7], 24, 24);
        randstr (str24[8], 24, 24);
        randstr (str24[9], 24, 24);
        randdatastr (s_data[i], 26, 50);

        if (gen) {
          printf ("%d %d %d %s %s %s %s %s %s %s %s %s %s\n",
                sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
                str24[3], str24[4], str24[5], str24[6], str24[7],
                str24[8], str24[9], s_data[i]);
        }
        else {
          s_i_id[i] = sid;
          s_w_id[i] = swid;
          strncpy (s_dist_01[i], str24[0], 24);
          strncpy (s_dist_02[i], str24[1], 24);
          strncpy (s_dist_03[i], str24[2], 24);
          strncpy (s_dist_04[i], str24[3], 24);
          strncpy (s_dist_05[i], str24[4], 24);
          strncpy (s_dist_06[i], str24[5], 24);
          strncpy (s_dist_07[i], str24[6], 24);
          strncpy (s_dist_08[i], str24[7], 24);
          strncpy (s_dist_09[i], str24[8], 24);
          strncpy (s_dist_10[i], str24[9], 24);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curs, STOCARR, 0)) {
          errrpt (&tpclda, &curs);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the HISTORY table.                                       |
 +---------------------------------------------------------------*/

  if (do_A || do_h) {
    nrows = (eware - bware + 1) * HISTFAC;

    fprintf (stderr, "Loading/generating history: w%d - w%d (%d rows)\n   ",
          bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < HISTARR; i++, row++) {
        cid++;
        if (cid > CUSTFAC) {       /* cycle cust id */
          cid = 1;                 /* cheap mod */
          cdid++;                  /* shift district cycle */
          if (cdid > DISTFAC) {
            cdid = 1;
            cwid++;                /* shift warehouse cycle */
          }
        }
        h_c_id[i] = cid;
        h_d_id[i] = cdid;
        h_w_id[i] = cwid;
        randstr (h_data[i], 12, 24);
        if (gen) {
          printf ("%d %d %d %d %d %s 100.00 %s\n", cid, cdid, cwid, cdid,
                cwid, sdate, h_data[i]);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curh, HISTARR, 0)) {
          errrpt (&tpclda, &curh);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                h_w_id[0], h_d_id[0], h_c_id[0]);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                h_w_id[0], h_d_id[0], h_c_id[0]);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the ORDERS and ORDER-LINE table.                         |
 +---------------------------------------------------------------*/

  if (do_A || do_o) {
    nrows = (eware - bware + 1) * ORDEFAC * DISTFAC;

    fprintf (stderr, "Loading/generating orders and order-line: w%d - w%d (%d ord, ~%d
ordl)\n   ",
```

```
        bware, eware, nrows, nrows * 10);

  begin_time = gettime ();
  begin_cpu = getcpu ();

  cid = 0;
  cdid = 1;
  cwid = bware;
  loopcount = 0;

  for (row = 0; row < nrows; ) {
    for (i = 0; i < ORDEARR; i++, row++) {
      cid++;
      if (cid > ORDEFAC) {       /* cycle cust id */
        cid = 1;                 /* cheap mod */
        cdid++;                  /* shift district cycle */
        if (cdid > DISTFAC) {
          cdid = 1;
          cwid++;                /* shift warehouse cycle */
        }
      }
      o_carrier_id[i] = rand () % 10 + 1;
      o_ol_cnt[i] = olcnt = rand () % 11 + 5;

      if (gen) {
        if (cid < 2101) {
          printf ("%d %d %d %d %s %d %d 1\n", cid, cdid, cwid,
                  randperm3000[cid - 1], sdate, o_carrier_id[i],
                  o_ol_cnt[i]);
        }
        else {
                    /* set carrierid to 11 instead of null 9/24 Saar */
          printf ("%d %d %d %d %s 11 %d 1\n", cid, cdid, cwid,
                  randperm3000[cid - 1], sdate, o_ol_cnt[i]);
        }
      }
      else {
        o_id[i] = cid;
        o_d_id[i] = cdid;
        o_w_id[i] = cwid;
        o_c_id[i] = randperm3000[cid - 1];
      }

      for (j = 0; j < o_ol_cnt[i]; j++) {
        ol_i_id[j] = sid = lrand48 () % 100000 + 1;
        if (cid < 2101)
          ol_amount[j] = 0;
        else
          ol_amount[j] = (lrand48 () % 999999 + 1) ;
        randstr (str24[j], 24, 24);

        if (gen) {
          if (cid < 2101) {
            fprintf (olfp, "%d %d %d %d %s %d %d %s\n", cid,
                     cdid, cwid, j + 1, sdate, ol_i_id[j], cwid,
                     ol_amount[j], str24[j]);
          }
          else {
                      /* change from null date, reorg columns 9/24 Saar */
            fprintf (olfp, "%d %d %d %d 01-Jan-1811 %d %d %s\n", cid,
                     cdid, cwid, j + 1, ol_i_id[j], cwid,
                     ol_amount[j], str24[j]);
          }
        }
        else {
          ol_o_id[j] = cid;
          ol_d_id[j] = cdid;
          ol_w_id[j] = cwid;
          ol_number[j] = j + 1;
          ol_supply_w_id[j] = cwid;
          strncpy (ol_dist_info[j], str24[j], 24);
        }
      }

      if (gen) {
        fflush (olfp);
      }
      else {
        if (cid < 2101) {
          if (oexn (&curol1, olcnt, 0)) {
            errrpt (&tpclda, &curol1);
            orol (&tpclda);
            fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                     cwid, cdid, cid);
            quit ();
            exit (1);
          }
          else if (ocom (&tpclda)) {
            errrpt (&tpclda, &tpclda);
            orol (&tpclda);
            fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                    cwid, cdid, cid);
            quit ();
            exit (1);
          }
        }
        else {
          if (oexn (&curol2, olcnt, 0)) {
            errrpt (&tpclda, &curol2);
            orol (&tpclda);
            fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                     cwid, cdid, cid);
            quit ();
            exit (1);
          }
          else if (ocom (&tpclda)) {
            errrpt (&tpclda, &tpclda);
            orol (&tpclda);
            fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                    cwid, cdid, cid);
            quit ();
            exit (1);
          }
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (cid < 2101) {
          if (oexn (&curo1, ORDEARR, 0)) {
            errrpt (&tpclda, &curo1);
            orol (&tpclda);
            fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n  ",
                     cwid, cdid, cid);
            quit ();
            exit (1);
          }
          else if (ocom (&tpclda)) {
            errrpt (&tpclda, &tpclda);
            orol (&tpclda);
            fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n  ",
                     cwid, cdid, cid);
            quit ();
            exit (1);
          }
        }
        else {
          if (oexn (&curo2, ORDEARR, 0)) {
            errrpt (&tpclda, &curo2);
            orol (&tpclda);
            fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n  ",
                     cwid, cdid, cid);
            quit ();
            exit (1);
          }
          else if (ocom (&tpclda)) {
            errrpt (&tpclda, &tpclda);
            orol (&tpclda);
            fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n  ",
                     cwid, cdid, cid);
            quit ();
            exit (1);
          }
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d orders committed\n  ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d orders loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
             nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

  /*-------------------------------------------------------------+
  | Load the NEW-ORDER table.                                    |
  +-------------------------------------------------------------*/

  if (do_A || do_n) {
    nrows = (eware - bware + 1) * NEWOFAC * DISTFAC;

    fprintf (stderr, "Loading/generating new-order: w%d - w%d (%d rows)\n  ",
             bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
```

```
      for (i = 0; i < NEWOARR; i++, row++) {
        cid++;
        if (cid > NEWOFAC) {
          cid = 1;
          cdid++;
          if (cdid > DISTFAC) {
            cdid = 1;
            cwid++;
          }
        }

        if (gen) {
          printf ("%d %d%d\n", cid + 2100, cdid, cwid);
        }
        else {
          no_o_id[i] = cid + 2100;
          no_d_id[i] = cdid;
          no_w_id[i] = cwid;
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curno, NEWOARR, 0)) {
          errrpt (&tpclda, &curno);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n  ",
                cwid, cdid, cid + 2100);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n  ",
                cwid, cdid, cid + 2100);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 45)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n  ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*-----------------------------------------------------------------+
 | clean up and exit.                                              |
 +-----------------------------------------------------------------*/

  if (olfp)
    fclose (olfp);
  if (!gen)
    quit ();
  exit (0);

}



initperm ()

{

  int i;
  int pos;
  int temp;

  /* init randperm3000 */

  for (i = 0; i < 3000; i++)
    randperm3000[i] = i + 1;
  for (i = 3000; i > 0; i--) {
    pos = rand () % i;
    temp = randperm3000[i - 1];
    randperm3000[i - 1] = randperm3000[pos];
    randperm3000[pos] = temp;
  }

}



randstr (str, x, y)

char *str;
int x;
int y;

{

  int i, j;
  int len;

  len = (rand () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = rand () % 62;
    if (j < 26)
      str[i] = (char) (j + 'a');
    else if (j < 52)
      str[i] = (char) (j - 26 + 'A');
    else
      str[i] = (char) (j - 52 + '0');
  }
  str[len] = '\0';

}



randdatastr (str, x, y)

char *str;
int x;
int y;

{

  int i, j;
  int len;
  int pos;

  len = (rand () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = rand () % 62;
    if (j < 26)
      str[i] = (char) (j + 'a');
    else if (j < 52)
      str[i] = (char) (j - 26 + 'A');
    else
      str[i] = (char) (j - 52 + '0');
  }
  str[len] = '\0';
  if ((rand () % 10) == 0) {
    pos = (rand () %(len - 8));
    str[pos] = 'O';
    str[pos + 1] = 'R';
    str[pos + 2] = 'I';
    str[pos + 3] = 'G';
    str[pos + 4] = 'I';
    str[pos + 5] = 'N';
    str[pos + 6] = 'A';
    str[pos + 7] = 'L';
  }

}



randnum (str, len)

char *str;
int len;

{

  int i;

  for (i = 0; i < len; i++)
    str[i] = (char) (rand () % 10 + '0');
  str[len] = '\0';

}



randlastname (str, id)

char *str;
int id;

{

  id = id % 1000;
  strcpy (str, lastname[id / 100]);
  strcat (str, lastname[(id / 10) % 10]);
  strcat (str, lastname[id % 10]);

}
```

```
NURand (A, x, y, cnum)

int A, x, y, cnum;

{

   int a, b;

   a = lrand48 () % (A + 1);
   b = (lrand48 () % (y - x + 1)) + x;
   return ((((a | b) + cnum) % (y - x + 1)) + x);

}



sysdate (sdate)

char *sdate;

{

   time_t tp;
   struct tm *tmptr;

   time (&tp);
   tmptr = localtime (&tp);
   strftime (sdate, 29, "%d-%b-%Y",tmptr);

}
```

Appendix D. RTE Scripts

D.1 RTE Parameters

```
/* For Oracle in the tpccload program C_LAST =1. C-Delta be the difference */
/*between C-LOAD and C-Run. C-Delta must be a value between 65..119
including the */
/* values of 65 and 119 and excluding the value of 96 and 112 */

/*-------------------------------------------------------------------------*/
/* This rteparams file is for doing 5 node runs.  One and only one of the    */
/* MASTER_NUM1,2,3 parameters must be set to 1 and the others must be set
to 0*/
/*-------------------------------------------------------------------------*/
#define MASTER_NUM1 1
#define MASTER_NUM2 0
#define MASTER_NUM3 0
MASTER "labperf1.austin.ibm.com"
/* MASTER "labperf2.austin.ibm.com" */
/* MASTER "labperf3.austin.ibm.com" */

/*---- SUT ----------------------------------------------------------------*/
#if MASTER_NUM1
  SUT="node1_en1 node4_en1"
#endif
#if MASTER_NUM2
  SUT="node2_en1 node3_en1"
#endif
#if MASTER_NUM3
  SUT="node5_en1"
#endif
/*-------------------------------------------------------------------------*/

LASTC=86
MEASUREMENT="1"

WAREHOUSES=11000

/*---- SLAVES -------------------------------------------------------------*/
#if MASTER_NUM1
/* SLAVES
driver1,driver1a,driver1b,driver1c,driver1d,driver1e,driver1f,driver1g,driver2,dri
ver2a,driver2b,driver2c,driver2d,driver2e,driver2f,driver2g,driver3,driver3a,drive
r3b,driver3c,driver3d,driver3e,driver3f,driver3g  */
SLAVES
driver1,driver1a,driver1b,driver1c,driver1d,driver1e,driver1f,driver1g,driver2,dri
ver2a,driver2b,driver2c,driver2d,driver2e,driver2f,driver2g,driver3,driver3a,drive
r3b,driver3c,driver3d,driver3e,driver3f,driver3g,driver10,driver10a,driver10b,dri
ver10c,driver10d,driver10e,driver10f,driver10g,driver11,driver11a,driver11b,driv
er11c,driver11d,driver11e,driver11f,driver11g,driver12,driver12a,driver12b,drive
r12c,driver12d,driver12e,driver12f,driver12g
/* SLAVES
driver10,driver10a,driver10b,driver10c,driver10d,driver10e,driver10f,driver10g,d
river11,driver11a,driver11b,driver11c,driver11d,driver11e,driver11f,driver11g,dri
ver12,driver12a,driver12b,driver12c,driver12d,driver12e,driver12f,driver12g   */
#endif
#if MASTER_NUM2
SLAVES
driver4,driver4a,driver4b,driver4c,driver4d,driver4e,driver4f,driver4g,driver5,dri
ver5a,driver5b,driver5c,driver5d,driver5e,driver5f,driver5g,driver6,driver6a,drive
r6b,driver6c,driver6d,driver6e,driver6f,driver6g,driver7,driver7a,driver7b,driver7
c,driver7d,driver7e,driver7f,driver7g,driver8,driver8a,driver8b,driver8c,driver8d,
driver8e,driver8f,driver8g,driver9,driver9a,driver9b,driver9c,driver9d,driver9e,dr
iver9f,driver9g
#endif
#if MASTER_NUM3
SLAVES
driver13,driver13a,driver13b,driver13c,driver13d,driver13e,driver13f,driver13g,d
river14,driver14a,driver14b,driver14c,driver14d,driver14e,driver14f,driver14g,dri
ver15,driver15a,driver15b,driver15c,driver15d,driver15e,driver15f,driver15g
#endif
/*-------------------------------------------------------------------------*/

/*---- CLIENTS ------------------------------------------------------------*/
#if MASTER_NUM1
MAIN_CLIENT   = client1
/*
CLIENT_REAL  = "client1 client2 client3 client4 client5 client6
            client7 client8 client9 client10 client11 client12"

*/

CLIENT_REAL  = "client1 client2 client3 client4 client5 client6
            client7 client8 client9 client10 client11 client12
            client37 client38 client39 client40 client41 client42
            client43 client44 client45 client46 client47 client48"
/*
CLIENT_REAL  = "client37 client38 client39 client40 client41 client42
            client43 client44 client45 client46 client47 client48"
*/
#endif
#if MASTER_NUM2
MAIN_CLIENT   = client13
CLIENT_REAL  = "client13 client14 client15 client16 client17 client18
            client19 client20 client21 client22 client23 client24
            client25 client26 client27 client28 client29 client30
            client31 client32 client33 client34 client35 client36"
#endif
#if MASTER_NUM3
MAIN_CLIENT   = client49
CLIENT_REAL  = "client49 client50 client51 client52 client53 client54
            client55 client56 client57 client58 client59 client60"
#endif
/*-------------------------------------------------------------------------*/

/*---- more client stuff --------------------------------------------------*/
#if MASTER_NUM1
CLIENT client1x1 oracle orif1db
CLIENT client2x1 oracle orif1db
CLIENT client3x1 oracle orif1db
CLIENT client4x1 oracle orif1db
CLIENT client1y1 oracle orif1db
CLIENT client2y1 oracle orif1db
CLIENT client3y1 oracle orif1db
CLIENT client4y1 oracle orif1db

CLIENT client5x2 oracle orif1db
CLIENT client6x2 oracle orif1db
CLIENT client7x2 oracle orif1db
CLIENT client8x2 oracle orif1db
CLIENT client5y2 oracle orif1db
CLIENT client6y2 oracle orif1db
CLIENT client7y2 oracle orif1db
CLIENT client8y2 oracle orif1db

CLIENT client9x3 oracle orif1db
CLIENT client10x3 oracle orif1db
CLIENT client11x3 oracle orif1db
CLIENT client12x3 oracle orif1db
CLIENT client9y3 oracle orif1db
CLIENT client10y3 oracle orif1db
CLIENT client11y3 oracle orif1db
CLIENT client12y3 oracle orif1db
#endif
#if MASTER_NUM2
CLIENT client13x4 oracle orif1db
CLIENT client14x4 oracle orif1db
CLIENT client15x4 oracle orif1db
CLIENT client16x4 oracle orif1db
CLIENT client13y4 oracle orif1db
CLIENT client14y4 oracle orif1db
CLIENT client15y4 oracle orif1db
CLIENT client16y4 oracle orif1db

CLIENT client17x5 oracle orif1db
CLIENT client18x5 oracle orif1db
CLIENT client19x5 oracle orif1db
CLIENT client20x5 oracle orif1db
CLIENT client17y5 oracle orif1db
CLIENT client18y5 oracle orif1db
CLIENT client19y5 oracle orif1db
CLIENT client20y5 oracle orif1db

CLIENT client21x6 oracle orif1db
CLIENT client22x6 oracle orif1db
CLIENT client23x6 oracle orif1db
CLIENT client24x6 oracle orif1db
CLIENT client21y6 oracle orif1db
CLIENT client22y6 oracle orif1db
CLIENT client23y6 oracle orif1db
CLIENT client24y6 oracle orif1db

CLIENT client25x7 oracle orif1db
CLIENT client26x7 oracle orif1db
CLIENT client27x7 oracle orif1db
CLIENT client28x7 oracle orif1db
CLIENT client25y7 oracle orif1db
CLIENT client26y7 oracle orif1db
CLIENT client27y7 oracle orif1db
CLIENT client28y7 oracle orif1db

CLIENT client29x8 oracle orif1db
CLIENT client30x8 oracle orif1db
CLIENT client31x8 oracle orif1db
CLIENT client32x8 oracle orif1db
```

```
CLIENT client29y8 oracle orif1db
CLIENT client30y8 oracle orif1db
CLIENT client31y8 oracle orif1db
CLIENT client32y8 oracle orif1db

CLIENT client33x9 oracle orif1db
CLIENT client34x9 oracle orif1db
CLIENT client35x9 oracle orif1db
CLIENT client36x9 oracle orif1db
CLIENT client33y9 oracle orif1db
CLIENT client34y9 oracle orif1db
CLIENT client35y9 oracle orif1db
CLIENT client36y9 oracle orif1db
#endif
#if MASTER_NUM1
CLIENT client37x10 oracle orif1db
CLIENT client38x10 oracle orif1db
CLIENT client39x10 oracle orif1db
CLIENT client40x10 oracle orif1db
CLIENT client37y10 oracle orif1db
CLIENT client38y10 oracle orif1db
CLIENT client39y10 oracle orif1db
CLIENT client40y10 oracle orif1db

CLIENT client41x11 oracle orif1db
CLIENT client42x11 oracle orif1db
CLIENT client43x11 oracle orif1db
CLIENT client44x11 oracle orif1db
CLIENT client41y11 oracle orif1db
CLIENT client42y11 oracle orif1db
CLIENT client43y11 oracle orif1db
CLIENT client44y11 oracle orif1db

CLIENT client45x12 oracle orif1db
CLIENT client46x12 oracle orif1db
CLIENT client47x12 oracle orif1db
CLIENT client48x12 oracle orif1db
CLIENT client45y12 oracle orif1db
CLIENT client46y12 oracle orif1db
CLIENT client47y12 oracle orif1db
CLIENT client48y12 oracle orif1db
#endif
#if MASTER_NUM3
CLIENT client49x13 oracle orif1db
CLIENT client50x13 oracle orif1db
CLIENT client51x13 oracle orif1db
CLIENT client52x13 oracle orif1db
CLIENT client49y13 oracle orif1db
CLIENT client50y13 oracle orif1db
CLIENT client51y13 oracle orif1db
CLIENT client52y13 oracle orif1db

CLIENT client53x14 oracle orif1db
CLIENT client54x14 oracle orif1db
CLIENT client55x14 oracle orif1db
CLIENT client56x14 oracle orif1db
CLIENT client53y14 oracle orif1db
CLIENT client54y14 oracle orif1db
CLIENT client55y14 oracle orif1db
CLIENT client56y14 oracle orif1db

CLIENT client57x15 oracle orif1db
CLIENT client58x15 oracle orif1db
CLIENT client59x15 oracle orif1db
CLIENT client60x15 oracle orif1db
CLIENT client57y15 oracle orif1db
CLIENT client58y15 oracle orif1db
CLIENT client59y15 oracle orif1db
CLIENT client60y15 oracle orif1db
#endif
/*-------------------------------------------------------------------------------*/

TELNET telnet 23
SOCKET socket 199703

/*---- Sockets ----------------------------------------------------------------*/
#if MASTER_NUM1
SOCKET_NETWORK socket1 6700 driver1
SOCKET_NETWORK socket2 6701 driver1a
SOCKET_NETWORK socket3 6702 driver1b
SOCKET_NETWORK socket4 6703 driver1c
SOCKET_NETWORK socket5 6704 driver1d
SOCKET_NETWORK socket6 6705 driver1e
SOCKET_NETWORK socket7 6706 driver1f
SOCKET_NETWORK socket8 6707 driver1g
SOCKET_NETWORK socket9 6700 driver1
SOCKET_NETWORK socket10 6701 driver1a
SOCKET_NETWORK socket11 6702 driver1b
SOCKET_NETWORK socket12 6703 driver1c
SOCKET_NETWORK socket13 6704 driver1d
SOCKET_NETWORK socket14 6705 driver1e
SOCKET_NETWORK socket15 6706 driver1f
SOCKET_NETWORK socket16 6707 driver1g
SOCKET_NETWORK socket17 6700 driver1
SOCKET_NETWORK socket18 6701 driver1a
SOCKET_NETWORK socket19 6702 driver1b
SOCKET_NETWORK socket20 6703 driver1c
SOCKET_NETWORK socket21 6704 driver1d
SOCKET_NETWORK socket22 6705 driver1e
SOCKET_NETWORK socket23 6706 driver1f
SOCKET_NETWORK socket24 6707 driver1g
SOCKET_NETWORK socket25 6700 driver1
SOCKET_NETWORK socket26 6701 driver1a
SOCKET_NETWORK socket27 6702 driver1b
SOCKET_NETWORK socket28 6703 driver1c
SOCKET_NETWORK socket29 6704 driver1d
SOCKET_NETWORK socket30 6705 driver1e
SOCKET_NETWORK socket31 6706 driver1f
SOCKET_NETWORK socket32 6707 driver1g
SOCKET_NETWORK socket33 6700 driver2
SOCKET_NETWORK socket34 6701 driver2a
SOCKET_NETWORK socket35 6702 driver2b
SOCKET_NETWORK socket36 6703 driver2c
SOCKET_NETWORK socket37 6704 driver2d
SOCKET_NETWORK socket38 6705 driver2e
SOCKET_NETWORK socket39 6706 driver2f
SOCKET_NETWORK socket40 6707 driver2g
SOCKET_NETWORK socket41 6700 driver2
SOCKET_NETWORK socket42 6701 driver2a
SOCKET_NETWORK socket43 6702 driver2b
SOCKET_NETWORK socket44 6703 driver2c
SOCKET_NETWORK socket45 6704 driver2d
SOCKET_NETWORK socket46 6705 driver2e
SOCKET_NETWORK socket47 6706 driver2f
SOCKET_NETWORK socket48 6707 driver2g
SOCKET_NETWORK socket49 6700 driver2
SOCKET_NETWORK socket50 6701 driver2a
SOCKET_NETWORK socket51 6702 driver2b
SOCKET_NETWORK socket52 6703 driver2c
SOCKET_NETWORK socket53 6704 driver2d
SOCKET_NETWORK socket54 6705 driver2e
SOCKET_NETWORK socket55 6706 driver2f
SOCKET_NETWORK socket56 6707 driver2g
SOCKET_NETWORK socket57 6700 driver2
SOCKET_NETWORK socket58 6701 driver2a
SOCKET_NETWORK socket59 6702 driver2b
SOCKET_NETWORK socket60 6703 driver2c
SOCKET_NETWORK socket61 6704 driver2d
SOCKET_NETWORK socket62 6705 driver2e
SOCKET_NETWORK socket63 6706 driver2f
SOCKET_NETWORK socket64 6707 driver2g
SOCKET_NETWORK socket65 6700 driver3
SOCKET_NETWORK socket66 6701 driver3a
SOCKET_NETWORK socket67 6702 driver3b
SOCKET_NETWORK socket68 6703 driver3c
SOCKET_NETWORK socket69 6704 driver3d
SOCKET_NETWORK socket70 6705 driver3e
SOCKET_NETWORK socket71 6706 driver3f
SOCKET_NETWORK socket72 6707 driver3g
SOCKET_NETWORK socket73 6700 driver3
SOCKET_NETWORK socket74 6701 driver3a
SOCKET_NETWORK socket75 6702 driver3b
SOCKET_NETWORK socket76 6703 driver3c
SOCKET_NETWORK socket77 6704 driver3d
SOCKET_NETWORK socket78 6705 driver3e
SOCKET_NETWORK socket79 6706 driver3f
SOCKET_NETWORK socket80 6707 driver3g
SOCKET_NETWORK socket81 6700 driver3
SOCKET_NETWORK socket82 6701 driver3a
SOCKET_NETWORK socket83 6702 driver3b
SOCKET_NETWORK socket84 6703 driver3c
SOCKET_NETWORK socket85 6704 driver3d
SOCKET_NETWORK socket86 6705 driver3e
SOCKET_NETWORK socket87 6706 driver3f
SOCKET_NETWORK socket88 6707 driver3g
SOCKET_NETWORK socket89 6700 driver3
SOCKET_NETWORK socket90 6701 driver3a
SOCKET_NETWORK socket91 6702 driver3b
SOCKET_NETWORK socket92 6703 driver3c
SOCKET_NETWORK socket93 6704 driver3d
SOCKET_NETWORK socket94 6705 driver3e
SOCKET_NETWORK socket95 6706 driver3f
SOCKET_NETWORK socket96 6707 driver3g
#endif
#if MASTER_NUM2
SOCKET_NETWORK socket97 6700 driver4
SOCKET_NETWORK socket98 6701 driver4a
SOCKET_NETWORK socket99 6702 driver4b
SOCKET_NETWORK socket100 6703 driver4c
SOCKET_NETWORK socket101 6704 driver4d
```

SOCKET_NETWORK socket102 6705 driver4e
SOCKET_NETWORK socket103 6706 driver4f
SOCKET_NETWORK socket104 6707 driver4g
SOCKET_NETWORK socket105 6700 driver4
SOCKET_NETWORK socket106 6701 driver4a
SOCKET_NETWORK socket107 6702 driver4b
SOCKET_NETWORK socket108 6703 driver4c
SOCKET_NETWORK socket109 6704 driver4d
SOCKET_NETWORK socket110 6705 driver4e
SOCKET_NETWORK socket111 6706 driver4f
SOCKET_NETWORK socket112 6707 driver4g
SOCKET_NETWORK socket113 6700 driver4
SOCKET_NETWORK socket114 6701 driver4a
SOCKET_NETWORK socket115 6702 driver4b
SOCKET_NETWORK socket116 6703 driver4c
SOCKET_NETWORK socket117 6704 driver4d
SOCKET_NETWORK socket118 6705 driver4e
SOCKET_NETWORK socket119 6706 driver4f
SOCKET_NETWORK socket120 6707 driver4g
SOCKET_NETWORK socket121 6700 driver4
SOCKET_NETWORK socket122 6701 driver4a
SOCKET_NETWORK socket123 6702 driver4b
SOCKET_NETWORK socket124 6703 driver4c
SOCKET_NETWORK socket125 6704 driver4d
SOCKET_NETWORK socket126 6705 driver4e
SOCKET_NETWORK socket127 6706 driver4f
SOCKET_NETWORK socket128 6707 driver4g
SOCKET_NETWORK socket129 6700 driver5
SOCKET_NETWORK socket130 6701 driver5a
SOCKET_NETWORK socket131 6702 driver5b
SOCKET_NETWORK socket132 6703 driver5c
SOCKET_NETWORK socket133 6704 driver5d
SOCKET_NETWORK socket134 6705 driver5e
SOCKET_NETWORK socket135 6706 driver5f
SOCKET_NETWORK socket136 6707 driver5g
SOCKET_NETWORK socket137 6700 driver5
SOCKET_NETWORK socket138 6701 driver5a
SOCKET_NETWORK socket139 6702 driver5b
SOCKET_NETWORK socket140 6703 driver5c
SOCKET_NETWORK socket141 6704 driver5d
SOCKET_NETWORK socket142 6705 driver5e
SOCKET_NETWORK socket143 6706 driver5f
SOCKET_NETWORK socket144 6707 driver5g
SOCKET_NETWORK socket145 6700 driver5
SOCKET_NETWORK socket146 6701 driver5a
SOCKET_NETWORK socket147 6702 driver5b
SOCKET_NETWORK socket148 6703 driver5c
SOCKET_NETWORK socket149 6704 driver5d
SOCKET_NETWORK socket150 6705 driver5e
SOCKET_NETWORK socket151 6706 driver5f
SOCKET_NETWORK socket152 6707 driver5g
SOCKET_NETWORK socket153 6700 driver5
SOCKET_NETWORK socket154 6701 driver5a
SOCKET_NETWORK socket155 6702 driver5b
SOCKET_NETWORK socket156 6703 driver5c
SOCKET_NETWORK socket157 6704 driver5d
SOCKET_NETWORK socket158 6705 driver5e
SOCKET_NETWORK socket159 6706 driver5f
SOCKET_NETWORK socket160 6707 driver5g
SOCKET_NETWORK socket161 6700 driver6
SOCKET_NETWORK socket162 6701 driver6a
SOCKET_NETWORK socket163 6702 driver6b
SOCKET_NETWORK socket164 6703 driver6c
SOCKET_NETWORK socket165 6704 driver6d
SOCKET_NETWORK socket166 6705 driver6e
SOCKET_NETWORK socket167 6706 driver6f
SOCKET_NETWORK socket168 6707 driver6g
SOCKET_NETWORK socket169 6700 driver6
SOCKET_NETWORK socket170 6701 driver6a
SOCKET_NETWORK socket171 6702 driver6b
SOCKET_NETWORK socket172 6703 driver6c
SOCKET_NETWORK socket173 6704 driver6d
SOCKET_NETWORK socket174 6705 driver6e
SOCKET_NETWORK socket175 6706 driver6f
SOCKET_NETWORK socket176 6707 driver6g
SOCKET_NETWORK socket177 6700 driver6
SOCKET_NETWORK socket178 6701 driver6a
SOCKET_NETWORK socket179 6702 driver6b
SOCKET_NETWORK socket180 6703 driver6c
SOCKET_NETWORK socket181 6704 driver6d
SOCKET_NETWORK socket182 6705 driver6e
SOCKET_NETWORK socket183 6706 driver6f
SOCKET_NETWORK socket184 6707 driver6g
SOCKET_NETWORK socket185 6700 driver6
SOCKET_NETWORK socket186 6701 driver6a
SOCKET_NETWORK socket187 6702 driver6b
SOCKET_NETWORK socket188 6703 driver6c
SOCKET_NETWORK socket189 6704 driver6d
SOCKET_NETWORK socket190 6705 driver6e
SOCKET_NETWORK socket191 6706 driver6f
SOCKET_NETWORK socket192 6707 driver6g

SOCKET_NETWORK socket193 6700 driver7
SOCKET_NETWORK socket194 6701 driver7a
SOCKET_NETWORK socket195 6702 driver7b
SOCKET_NETWORK socket196 6703 driver7c
SOCKET_NETWORK socket197 6704 driver7d
SOCKET_NETWORK socket198 6705 driver7e
SOCKET_NETWORK socket199 6706 driver7f
SOCKET_NETWORK socket200 6707 driver7g
SOCKET_NETWORK socket201 6700 driver7
SOCKET_NETWORK socket202 6701 driver7a
SOCKET_NETWORK socket203 6702 driver7b
SOCKET_NETWORK socket204 6703 driver7c
SOCKET_NETWORK socket205 6704 driver7d
SOCKET_NETWORK socket206 6705 driver7e
SOCKET_NETWORK socket207 6706 driver7f
SOCKET_NETWORK socket208 6707 driver7g
SOCKET_NETWORK socket209 6700 driver7
SOCKET_NETWORK socket210 6701 driver7a
SOCKET_NETWORK socket211 6702 driver7b
SOCKET_NETWORK socket212 6703 driver7c
SOCKET_NETWORK socket213 6704 driver7d
SOCKET_NETWORK socket214 6705 driver7e
SOCKET_NETWORK socket215 6706 driver7f
SOCKET_NETWORK socket216 6707 driver7g
SOCKET_NETWORK socket217 6700 driver7
SOCKET_NETWORK socket218 6701 driver7a
SOCKET_NETWORK socket219 6702 driver7b
SOCKET_NETWORK socket220 6703 driver7c
SOCKET_NETWORK socket221 6704 driver7d
SOCKET_NETWORK socket222 6705 driver7e
SOCKET_NETWORK socket223 6706 driver7f
SOCKET_NETWORK socket224 6707 driver7g
SOCKET_NETWORK socket225 6700 driver8
SOCKET_NETWORK socket226 6701 driver8a
SOCKET_NETWORK socket227 6702 driver8b
SOCKET_NETWORK socket228 6703 driver8c
SOCKET_NETWORK socket229 6704 driver8d
SOCKET_NETWORK socket230 6705 driver8e
SOCKET_NETWORK socket231 6706 driver8f
SOCKET_NETWORK socket232 6707 driver8g
SOCKET_NETWORK socket233 6700 driver8
SOCKET_NETWORK socket234 6701 driver8a
SOCKET_NETWORK socket235 6702 driver8b
SOCKET_NETWORK socket236 6703 driver8c
SOCKET_NETWORK socket237 6704 driver8d
SOCKET_NETWORK socket238 6705 driver8e
SOCKET_NETWORK socket239 6706 driver8f
SOCKET_NETWORK socket240 6707 driver8g
SOCKET_NETWORK socket241 6700 driver8
SOCKET_NETWORK socket242 6701 driver8a
SOCKET_NETWORK socket243 6702 driver8b
SOCKET_NETWORK socket244 6703 driver8c
SOCKET_NETWORK socket245 6704 driver8d
SOCKET_NETWORK socket246 6705 driver8e
SOCKET_NETWORK socket247 6706 driver8f
SOCKET_NETWORK socket248 6707 driver8g
SOCKET_NETWORK socket249 6700 driver8
SOCKET_NETWORK socket250 6701 driver8a
SOCKET_NETWORK socket251 6702 driver8b
SOCKET_NETWORK socket252 6703 driver8c
SOCKET_NETWORK socket253 6704 driver8d
SOCKET_NETWORK socket254 6705 driver8e
SOCKET_NETWORK socket255 6706 driver8f
SOCKET_NETWORK socket256 6707 driver8g
SOCKET_NETWORK socket257 6700 driver9
SOCKET_NETWORK socket258 6701 driver9a
SOCKET_NETWORK socket259 6702 driver9b
SOCKET_NETWORK socket260 6703 driver9c
SOCKET_NETWORK socket261 6704 driver9d
SOCKET_NETWORK socket262 6705 driver9e
SOCKET_NETWORK socket263 6706 driver9f
SOCKET_NETWORK socket264 6707 driver9g
SOCKET_NETWORK socket265 6700 driver9
SOCKET_NETWORK socket266 6701 driver9a
SOCKET_NETWORK socket267 6702 driver9b
SOCKET_NETWORK socket268 6703 driver9c
SOCKET_NETWORK socket269 6704 driver9d
SOCKET_NETWORK socket270 6705 driver9e
SOCKET_NETWORK socket271 6706 driver9f
SOCKET_NETWORK socket272 6707 driver9g
SOCKET_NETWORK socket273 6700 driver9
SOCKET_NETWORK socket274 6701 driver9a
SOCKET_NETWORK socket275 6702 driver9b
SOCKET_NETWORK socket276 6703 driver9c
SOCKET_NETWORK socket277 6704 driver9d
SOCKET_NETWORK socket278 6705 driver9e
SOCKET_NETWORK socket279 6706 driver9f
SOCKET_NETWORK socket280 6707 driver9g

```
SOCKET_NETWORK socket281 6700 driver9
SOCKET_NETWORK socket282 6701 driver9a
SOCKET_NETWORK socket283 6702 driver9b
SOCKET_NETWORK socket284 6703 driver9c
SOCKET_NETWORK socket285 6704 driver9d
SOCKET_NETWORK socket286 6705 driver9e
SOCKET_NETWORK socket287 6706 driver9f
SOCKET_NETWORK socket288 6707 driver9g
#endif
#if MASTER_NUM1
SOCKET_NETWORK socket289 6700 driver10
SOCKET_NETWORK socket290 6701 driver10a
SOCKET_NETWORK socket291 6702 driver10b
SOCKET_NETWORK socket292 6703 driver10c
SOCKET_NETWORK socket293 6704 driver10d
SOCKET_NETWORK socket294 6705 driver10e
SOCKET_NETWORK socket295 6706 driver10f
SOCKET_NETWORK socket296 6707 driver10g
SOCKET_NETWORK socket297 6700 driver10
SOCKET_NETWORK socket298 6701 driver10a
SOCKET_NETWORK socket299 6702 driver10b
SOCKET_NETWORK socket300 6703 driver10c
SOCKET_NETWORK socket301 6704 driver10d
SOCKET_NETWORK socket302 6705 driver10e
SOCKET_NETWORK socket303 6706 driver10f
SOCKET_NETWORK socket304 6707 driver10g
SOCKET_NETWORK socket305 6700 driver10
SOCKET_NETWORK socket306 6701 driver10a
SOCKET_NETWORK socket307 6702 driver10b
SOCKET_NETWORK socket308 6703 driver10c
SOCKET_NETWORK socket309 6704 driver10d
SOCKET_NETWORK socket310 6705 driver10e
SOCKET_NETWORK socket311 6706 driver10f
SOCKET_NETWORK socket312 6707 driver10g
SOCKET_NETWORK socket313 6700 driver10
SOCKET_NETWORK socket314 6701 driver10a
SOCKET_NETWORK socket315 6702 driver10b
SOCKET_NETWORK socket316 6703 driver10c
SOCKET_NETWORK socket317 6704 driver10d
SOCKET_NETWORK socket318 6705 driver10e
SOCKET_NETWORK socket319 6706 driver10f
SOCKET_NETWORK socket320 6707 driver10g
SOCKET_NETWORK socket321 6700 driver11
SOCKET_NETWORK socket322 6701 driver11a
SOCKET_NETWORK socket323 6702 driver11b
SOCKET_NETWORK socket324 6703 driver11c
SOCKET_NETWORK socket325 6704 driver11d
SOCKET_NETWORK socket326 6705 driver11e
SOCKET_NETWORK socket327 6706 driver11f
SOCKET_NETWORK socket328 6707 driver11g
SOCKET_NETWORK socket329 6700 driver11
SOCKET_NETWORK socket330 6701 driver11a
SOCKET_NETWORK socket331 6702 driver11b
SOCKET_NETWORK socket332 6703 driver11c
SOCKET_NETWORK socket333 6704 driver11d
SOCKET_NETWORK socket334 6705 driver11e
SOCKET_NETWORK socket335 6706 driver11f
SOCKET_NETWORK socket336 6707 driver11g
SOCKET_NETWORK socket337 6700 driver11
SOCKET_NETWORK socket338 6701 driver11a
SOCKET_NETWORK socket339 6702 driver11b
SOCKET_NETWORK socket340 6703 driver11c
SOCKET_NETWORK socket341 6704 driver11d
SOCKET_NETWORK socket342 6705 driver11e
SOCKET_NETWORK socket343 6706 driver11f
SOCKET_NETWORK socket344 6707 driver11g
SOCKET_NETWORK socket345 6700 driver11
SOCKET_NETWORK socket346 6701 driver11a
SOCKET_NETWORK socket347 6702 driver11b
SOCKET_NETWORK socket348 6703 driver11c
SOCKET_NETWORK socket349 6704 driver11d
SOCKET_NETWORK socket350 6705 driver11e
SOCKET_NETWORK socket351 6706 driver11f
SOCKET_NETWORK socket352 6707 driver11g
SOCKET_NETWORK socket353 6700 driver12
SOCKET_NETWORK socket354 6701 driver12a
SOCKET_NETWORK socket355 6702 driver12b
SOCKET_NETWORK socket356 6703 driver12c
SOCKET_NETWORK socket357 6704 driver12d
SOCKET_NETWORK socket358 6705 driver12e
SOCKET_NETWORK socket359 6706 driver12f
SOCKET_NETWORK socket360 6707 driver12g
SOCKET_NETWORK socket361 6700 driver12
SOCKET_NETWORK socket362 6701 driver12a
SOCKET_NETWORK socket363 6702 driver12b
SOCKET_NETWORK socket364 6703 driver12c
SOCKET_NETWORK socket365 6704 driver12d
SOCKET_NETWORK socket366 6705 driver12e
SOCKET_NETWORK socket367 6706 driver12f
SOCKET_NETWORK socket368 6707 driver12g
SOCKET_NETWORK socket369 6700 driver12
SOCKET_NETWORK socket370 6701 driver12a
SOCKET_NETWORK socket371 6702 driver12b
SOCKET_NETWORK socket372 6703 driver12c
SOCKET_NETWORK socket373 6704 driver12d
SOCKET_NETWORK socket374 6705 driver12e
SOCKET_NETWORK socket375 6706 driver12f
SOCKET_NETWORK socket376 6707 driver12g
SOCKET_NETWORK socket377 6700 driver12
SOCKET_NETWORK socket378 6701 driver12a
SOCKET_NETWORK socket379 6702 driver12b
SOCKET_NETWORK socket380 6703 driver12c
SOCKET_NETWORK socket381 6704 driver12d
SOCKET_NETWORK socket382 6705 driver12e
SOCKET_NETWORK socket383 6706 driver12f
SOCKET_NETWORK socket384 6707 driver12g
#endif
#if MASTER_NUM3
SOCKET_NETWORK socket385 6700 driver13
SOCKET_NETWORK socket386 6701 driver13a
SOCKET_NETWORK socket387 6702 driver13b
SOCKET_NETWORK socket388 6703 driver13c
SOCKET_NETWORK socket389 6704 driver13d
SOCKET_NETWORK socket390 6705 driver13e
SOCKET_NETWORK socket391 6706 driver13f
SOCKET_NETWORK socket392 6707 driver13g
SOCKET_NETWORK socket393 6700 driver13
SOCKET_NETWORK socket394 6701 driver13a
SOCKET_NETWORK socket395 6702 driver13b
SOCKET_NETWORK socket396 6703 driver13c
SOCKET_NETWORK socket397 6704 driver13d
SOCKET_NETWORK socket398 6705 driver13e
SOCKET_NETWORK socket399 6706 driver13f
SOCKET_NETWORK socket400 6707 driver13g
SOCKET_NETWORK socket401 6700 driver13
SOCKET_NETWORK socket402 6701 driver13a
SOCKET_NETWORK socket403 6702 driver13b
SOCKET_NETWORK socket404 6703 driver13c
SOCKET_NETWORK socket405 6704 driver13d
SOCKET_NETWORK socket406 6705 driver13e
SOCKET_NETWORK socket407 6706 driver13f
SOCKET_NETWORK socket408 6707 driver13g
SOCKET_NETWORK socket409 6700 driver13
SOCKET_NETWORK socket410 6701 driver13a
SOCKET_NETWORK socket411 6702 driver13b
SOCKET_NETWORK socket412 6703 driver13c
SOCKET_NETWORK socket413 6704 driver13d
SOCKET_NETWORK socket414 6705 driver13e
SOCKET_NETWORK socket415 6706 driver13f
SOCKET_NETWORK socket416 6707 driver13g
SOCKET_NETWORK socket417 6700 driver14
SOCKET_NETWORK socket418 6701 driver14a
SOCKET_NETWORK socket419 6702 driver14b
SOCKET_NETWORK socket420 6703 driver14c
SOCKET_NETWORK socket421 6704 driver14d
SOCKET_NETWORK socket422 6705 driver14e
SOCKET_NETWORK socket423 6706 driver14f
SOCKET_NETWORK socket424 6707 driver14g
SOCKET_NETWORK socket425 6700 driver14
SOCKET_NETWORK socket426 6701 driver14a
SOCKET_NETWORK socket427 6702 driver14b
SOCKET_NETWORK socket428 6703 driver14c
SOCKET_NETWORK socket429 6704 driver14d
SOCKET_NETWORK socket430 6705 driver14e
SOCKET_NETWORK socket431 6706 driver14f
SOCKET_NETWORK socket432 6707 driver14g
SOCKET_NETWORK socket433 6700 driver14
SOCKET_NETWORK socket434 6701 driver14a
SOCKET_NETWORK socket435 6702 driver14b
SOCKET_NETWORK socket436 6703 driver14c
SOCKET_NETWORK socket437 6704 driver14d
SOCKET_NETWORK socket438 6705 driver14e
SOCKET_NETWORK socket439 6706 driver14f
SOCKET_NETWORK socket440 6707 driver14g
SOCKET_NETWORK socket441 6700 driver14
SOCKET_NETWORK socket442 6701 driver14a
SOCKET_NETWORK socket443 6702 driver14b
SOCKET_NETWORK socket444 6703 driver14c
SOCKET_NETWORK socket445 6704 driver14d
SOCKET_NETWORK socket446 6705 driver14e
SOCKET_NETWORK socket447 6706 driver14f
SOCKET_NETWORK socket448 6707 driver14g
SOCKET_NETWORK socket449 6700 driver15
SOCKET_NETWORK socket450 6701 driver15a
SOCKET_NETWORK socket451 6702 driver15b
SOCKET_NETWORK socket452 6703 driver15c
SOCKET_NETWORK socket453 6704 driver15d
SOCKET_NETWORK socket454 6705 driver15e
SOCKET_NETWORK socket455 6706 driver15f
SOCKET_NETWORK socket456 6707 driver15g
```

```
SOCKET_NETWORK socket457 6700 driver15
SOCKET_NETWORK socket458 6701 driver15a
SOCKET_NETWORK socket459 6702 driver15b
SOCKET_NETWORK socket460 6703 driver15c
SOCKET_NETWORK socket461 6704 driver15d
SOCKET_NETWORK socket462 6705 driver15e
SOCKET_NETWORK socket463 6706 driver15f
SOCKET_NETWORK socket464 6707 driver15g
SOCKET_NETWORK socket465 6700 driver15
SOCKET_NETWORK socket466 6701 driver15a
SOCKET_NETWORK socket467 6702 driver15b
SOCKET_NETWORK socket468 6703 driver15c
SOCKET_NETWORK socket469 6704 driver15d
SOCKET_NETWORK socket470 6705 driver15e
SOCKET_NETWORK socket471 6706 driver15f
SOCKET_NETWORK socket472 6707 driver15g
SOCKET_NETWORK socket473 6700 driver15
SOCKET_NETWORK socket474 6701 driver15a
SOCKET_NETWORK socket475 6702 driver15b
SOCKET_NETWORK socket476 6703 driver15c
SOCKET_NETWORK socket477 6704 driver15d
SOCKET_NETWORK socket478 6705 driver15e
SOCKET_NETWORK socket479 6706 driver15f
SOCKET_NETWORK socket480 6707 driver15g
#endif
/*-------------------------------------------------------------------------*/

OUTPUTNAME="../runs/labperf1"

CPU=12

#if 0
BEGIN_WAIT=20:00
RAMPUP=25:00
RUNTIME=15:00
RAMPDOWN_WAIT=1:00
RAMPDOWN=10:00
#else
BEGIN_WAIT=25:00
RAMPUP=75:00
RUNTIME=2:05:00
RAMPDOWN_WAIT=5:00
RAMPDOWN=20:00
#endif
INTERVAL=1:00   /* Interval to calculate mix from */

LOGIN_MAX_LOAD      = 4
LOGIN_BEGIN         = 0  /* skip login state if set to 1 */
NOBEGIN             = 1
KEYSTROKE_PACKET_SIZE = 0

MAX_CONCURRENT_SPAWN  = 8
SPAWN_COUNT         = 4

MIN_PORT            = 8088
MAX_PORT            = 8089

/* User variables.  Think, Emulex Delay, %desired, %min, %max*/
#if 0 /* Original */
NEWORDER = "12.03, 0, 0"
PAYMENT  = "12.03, 0, 0, 43.07, 43.07, 43.07 "
ORDSTAT  = "10.03, 0, 0,  4.05,  4.05,  4.05 "
DELIVERY = "05.03, 0, 0,  4.05,  4.05,  4.05 "
STOCKLEV = "05.03, 0, 0,  4.05,  4.05,  4.05 "
#elif 0
#elif 0 /* From rteparams.null */
NEWORDER = "12.25, 0.42, 0.38"
PAYMENT  = "12.25, 0.19, 0.23, 43.2, 41.1, 45.3 "
ORDSTAT  = "10.50, 0.39, 0.21,  4.1,  3.9,  4.3 "
DELIVERY = "05.5,  0.19, 0.15,  4.1,  3.9,  4.3 "
STOCKLEV = "05.5,  0.25, 0.18,  4.1,  3.9,  4.3 "
#elif 0 /* From Pookeepsie */
NEWORDER = "16.25, 0.42, 0.38"
PAYMENT  = "16.25, 0.19, 0.23, 43.15, 43.15, 43.15 "
ORDSTAT  = "14.50, 0.39, 0.21,  4.03,  4.03,  4.03 "
DELIVERY = "09.50, 0.19, 0.15,  4.03,  4.03,  4.03 "
STOCKLEV = "09.50, 0.25, 0.18,  4.03,  4.03,  4.03 "
#elif 1 /* Testing */
NEWORDER = "15.32, 0, 0"
PAYMENT  = "15.32, 0, 0, 43.07, 43.07, 43.07 "
ORDSTAT  = "13.32, 0, 0,  4.05,  4.05,  4.05 "
DELIVERY = "08.32, 0, 0,  4.05,  4.05,  4.05 "
STOCKLEV = "08.32, 0, 0,  4.05,  4.05,  4.05 "
#endif

/*---- Starting users on sockets ------------------------------------------*/
#if MASTER_NUM1
START_RANGE client1x1 socket1 230 0-23
START_RANGE client1y1 socket2 230 23-46
START_RANGE client1x1 socket3 230 46-69
START_RANGE client1y1 socket4 230 69-92
START_RANGE client1x1 socket5 230 92-115
START_RANGE client1y1 socket6 230 115-138
START_RANGE client1x1 socket7 230 138-161
START_RANGE client1y1 socket8 230 161-184
START_RANGE client2x1 socket9 230 184-207
START_RANGE client2y1 socket10 230 207-230
START_RANGE client2x1 socket11 230 230-253
START_RANGE client2y1 socket12 220 253-275
START_RANGE client2x1 socket13 230 275-298
START_RANGE client2y1 socket14 230 298-321
START_RANGE client2x1 socket15 230 321-344
START_RANGE client2y1 socket16 230 344-367
START_RANGE client3x1 socket17 230 367-390
START_RANGE client3y1 socket18 230 390-413
START_RANGE client3x1 socket19 230 413-436
START_RANGE client3y1 socket20 230 436-459
START_RANGE client3x1 socket21 230 459-482
START_RANGE client3y1 socket22 230 482-505
START_RANGE client3x1 socket23 230 505-528
START_RANGE client3y1 socket24 220 528-550
START_RANGE client4x1 socket25 230 550-573
START_RANGE client4y1 socket26 230 573-596
START_RANGE client4x1 socket27 230 596-619
START_RANGE client4y1 socket28 230 619-642
START_RANGE client4x1 socket29 230 642-665
START_RANGE client4y1 socket30 230 665-688
START_RANGE client4x1 socket31 230 688-711
START_RANGE client4y1 socket32 230 711-734
START_RANGE client5x2 socket33 230 734-757
START_RANGE client5y2 socket34 230 757-780
START_RANGE client5x2 socket35 230 780-803
START_RANGE client5y2 socket36 220 803-825
START_RANGE client5x2 socket37 230 825-848
START_RANGE client5y2 socket38 230 848-871
START_RANGE client5x2 socket39 230 871-894
START_RANGE client5y2 socket40 230 894-917
START_RANGE client6x2 socket41 230 917-940
START_RANGE client6y2 socket42 230 940-963
START_RANGE client6x2 socket43 230 963-986
START_RANGE client6y2 socket44 230 986-1009
START_RANGE client6x2 socket45 230 1009-1032
START_RANGE client6y2 socket46 230 1032-1055
START_RANGE client6x2 socket47 230 1055-1078
START_RANGE client6y2 socket48 220 1078-1100
START_RANGE client7x2 socket49 230 1100-1123
START_RANGE client7y2 socket50 230 1123-1146
START_RANGE client7x2 socket51 230 1146-1169
START_RANGE client7y2 socket52 230 1169-1192
START_RANGE client7x2 socket53 230 1192-1215
START_RANGE client7y2 socket54 230 1215-1238
START_RANGE client7x2 socket55 230 1238-1261
START_RANGE client7y2 socket56 230 1261-1284
START_RANGE client8x2 socket57 230 1284-1307
START_RANGE client8y2 socket58 230 1307-1330
START_RANGE client8x2 socket59 230 1330-1353
START_RANGE client8y2 socket60 220 1353-1375
START_RANGE client8x2 socket61 230 1375-1398
START_RANGE client8y2 socket62 230 1398-1421
START_RANGE client8x2 socket63 230 1421-1444
START_RANGE client8y2 socket64 230 1444-1467
START_RANGE client9x3 socket65 230 1467-1490
START_RANGE client9y3 socket66 230 1490-1513
START_RANGE client9x3 socket67 230 1513-1536
START_RANGE client9y3 socket68 230 1536-1559
START_RANGE client9x3 socket69 230 1559-1582
START_RANGE client9y3 socket70 230 1582-1605
START_RANGE client9x3 socket71 230 1605-1628
START_RANGE client9y3 socket72 220 1628-1650
START_RANGE client10x3 socket73 230 1650-1673
START_RANGE client10y3 socket74 230 1673-1696
START_RANGE client10x3 socket75 230 1696-1719
START_RANGE client10y3 socket76 230 1719-1742
START_RANGE client10x3 socket77 230 1742-1765
START_RANGE client10y3 socket78 230 1765-1788
START_RANGE client10x3 socket79 230 1788-1811
START_RANGE client10y3 socket80 230 1811-1834
START_RANGE client11x3 socket81 230 1834-1857
START_RANGE client11y3 socket82 230 1857-1880
START_RANGE client11x3 socket83 230 1880-1903
START_RANGE client11y3 socket84 220 1903-1925
START_RANGE client11x3 socket85 230 1925-1948
START_RANGE client11y3 socket86 230 1948-1971
START_RANGE client11x3 socket87 230 1971-1994
START_RANGE client11y3 socket88 230 1994-2017
START_RANGE client12x3 socket89 230 2017-2040
START_RANGE client12y3 socket90 230 2040-2063
START_RANGE client12x3 socket91 230 2063-2086
START_RANGE client12y3 socket92 230 2086-2109
START_RANGE client12x3 socket93 230 2109-2132
```

```
START_RANGE client12y3 socket94 230 2132-2155
START_RANGE client12x3 socket95 230 2155-2178
START_RANGE client12y3 socket96 220 2178-2200
#endif
#if MASTER_NUM2
START_RANGE client13x4 socket97 230 2200-2223
START_RANGE client13y4 socket98 230 2223-2246
START_RANGE client13x4 socket99 230 2246-2269
START_RANGE client13y4 socket100 230 2269-2292
START_RANGE client13x4 socket101 230 2292-2315
START_RANGE client13y4 socket102 230 2315-2338
START_RANGE client13x4 socket103 230 2338-2361
START_RANGE client13y4 socket104 230 2361-2384
START_RANGE client14x4 socket105 230 2384-2407
START_RANGE client14y4 socket106 230 2407-2430
START_RANGE client14x4 socket107 230 2430-2453
START_RANGE client14y4 socket108 220 2453-2475
START_RANGE client14x4 socket109 230 2475-2498
START_RANGE client14y4 socket110 230 2498-2521
START_RANGE client14x4 socket111 230 2521-2544
START_RANGE client14y4 socket112 230 2544-2567
START_RANGE client15x4 socket113 230 2567-2590
START_RANGE client15y4 socket114 230 2590-2613
START_RANGE client15x4 socket115 230 2613-2636
START_RANGE client15y4 socket116 230 2636-2659
START_RANGE client15x4 socket117 230 2659-2682
START_RANGE client15y4 socket118 230 2682-2705
START_RANGE client15x4 socket119 230 2705-2728
START_RANGE client15y4 socket120 220 2728-2750
START_RANGE client16x4 socket121 230 2750-2773
START_RANGE client16y4 socket122 230 2773-2796
START_RANGE client16x4 socket123 230 2796-2819
START_RANGE client16y4 socket124 230 2819-2842
START_RANGE client16x4 socket125 230 2842-2865
START_RANGE client16y4 socket126 230 2865-2888
START_RANGE client16x4 socket127 230 2888-2911
START_RANGE client16y4 socket128 230 2911-2934
START_RANGE client17x5 socket129 230 2934-2957
START_RANGE client17y5 socket130 230 2957-2980
START_RANGE client17x5 socket131 230 2980-3003
START_RANGE client17y5 socket132 220 3003-3025
START_RANGE client17x5 socket133 230 3025-3048
START_RANGE client17y5 socket134 230 3048-3071
START_RANGE client17x5 socket135 230 3071-3094
START_RANGE client17y5 socket136 230 3094-3117
START_RANGE client18x5 socket137 230 3117-3140
START_RANGE client18y5 socket138 230 3140-3163
START_RANGE client18x5 socket139 230 3163-3186
START_RANGE client18y5 socket140 230 3186-3209
START_RANGE client18x5 socket141 230 3209-3232
START_RANGE client18y5 socket142 230 3232-3255
START_RANGE client18x5 socket143 230 3255-3278
START_RANGE client18y5 socket144 220 3278-3300
START_RANGE client19x5 socket145 230 3300-3323
START_RANGE client19y5 socket146 230 3323-3346
START_RANGE client19x5 socket147 230 3346-3369
START_RANGE client19y5 socket148 230 3369-3392
START_RANGE client19x5 socket149 230 3392-3415
START_RANGE client19y5 socket150 230 3415-3438
START_RANGE client19x5 socket151 230 3438-3461
START_RANGE client19y5 socket152 230 3461-3484
START_RANGE client20x5 socket153 230 3484-3507
START_RANGE client20y5 socket154 230 3507-3530
START_RANGE client20x5 socket155 230 3530-3553
START_RANGE client20y5 socket156 220 3553-3575
START_RANGE client20x5 socket157 230 3575-3598
START_RANGE client20y5 socket158 230 3598-3621
START_RANGE client20x5 socket159 230 3621-3644
START_RANGE client20y5 socket160 230 3644-3667
START_RANGE client21x6 socket161 230 3667-3690
START_RANGE client21y6 socket162 230 3690-3713
START_RANGE client21x6 socket163 230 3713-3736
START_RANGE client21y6 socket164 230 3736-3759
START_RANGE client21x6 socket165 230 3759-3782
START_RANGE client21y6 socket166 230 3782-3805
START_RANGE client21x6 socket167 230 3805-3828
START_RANGE client21y6 socket168 220 3828-3850
START_RANGE client22x6 socket169 230 3850-3873
START_RANGE client22y6 socket170 230 3873-3896
START_RANGE client22x6 socket171 230 3896-3919
START_RANGE client22y6 socket172 230 3919-3942
START_RANGE client22x6 socket173 230 3942-3965
START_RANGE client22y6 socket174 230 3965-3988
START_RANGE client22x6 socket175 230 3988-4011
START_RANGE client22y6 socket176 230 4011-4034
START_RANGE client23x6 socket177 230 4034-4057
START_RANGE client23y6 socket178 230 4057-4080
START_RANGE client23x6 socket179 230 4080-4103
START_RANGE client23y6 socket180 220 4103-4125
START_RANGE client23x6 socket181 230 4125-4148
START_RANGE client23y6 socket182 230 4148-4171
START_RANGE client23x6 socket183 230 4171-4194
START_RANGE client23y6 socket184 230 4194-4217
START_RANGE client24x6 socket185 230 4217-4240
START_RANGE client24y6 socket186 230 4240-4263
START_RANGE client24x6 socket187 230 4263-4286
START_RANGE client24y6 socket188 230 4286-4309
START_RANGE client24x6 socket189 230 4309-4332
START_RANGE client24y6 socket190 230 4332-4355
START_RANGE client24x6 socket191 230 4355-4378
START_RANGE client24y6 socket192 220 4378-4400

START_RANGE client25x7 socket193 230 4400-4423
START_RANGE client25y7 socket194 230 4423-4446
START_RANGE client25x7 socket195 230 4446-4469
START_RANGE client25y7 socket196 230 4469-4492
START_RANGE client25x7 socket197 230 4492-4515
START_RANGE client25y7 socket198 230 4515-4538
START_RANGE client25x7 socket199 230 4538-4561
START_RANGE client25y7 socket200 230 4561-4584
START_RANGE client26x7 socket201 230 4584-4607
START_RANGE client26y7 socket202 230 4607-4630
START_RANGE client26x7 socket203 230 4630-4653
START_RANGE client26y7 socket204 220 4653-4675
START_RANGE client26x7 socket205 230 4675-4698
START_RANGE client26y7 socket206 230 4698-4721
START_RANGE client26x7 socket207 230 4721-4744
START_RANGE client26y7 socket208 230 4744-4767
START_RANGE client27x7 socket209 230 4767-4790
START_RANGE client27y7 socket210 230 4790-4813
START_RANGE client27x7 socket211 230 4813-4836
START_RANGE client27y7 socket212 230 4836-4859
START_RANGE client27x7 socket213 230 4859-4882
START_RANGE client27y7 socket214 230 4882-4905
START_RANGE client27x7 socket215 230 4905-4928
START_RANGE client27y7 socket216 220 4928-4950
START_RANGE client28x7 socket217 230 4950-4973
START_RANGE client28y7 socket218 230 4973-4996
START_RANGE client28x7 socket219 230 4996-5019
START_RANGE client28y7 socket220 230 5019-5042
START_RANGE client28x7 socket221 230 5042-5065
START_RANGE client28y7 socket222 230 5065-5088
START_RANGE client28x7 socket223 230 5088-5111
START_RANGE client28y7 socket224 230 5111-5134
START_RANGE client29x8 socket225 230 5134-5157
START_RANGE client29y8 socket226 230 5157-5180
START_RANGE client29x8 socket227 230 5180-5203
START_RANGE client29y8 socket228 220 5203-5225
START_RANGE client29x8 socket229 230 5225-5248
START_RANGE client29y8 socket230 230 5248-5271
START_RANGE client29x8 socket231 230 5271-5294
START_RANGE client29y8 socket232 230 5294-5317
START_RANGE client30x8 socket233 230 5317-5340
START_RANGE client30y8 socket234 230 5340-5363
START_RANGE client30x8 socket235 230 5363-5386
START_RANGE client30y8 socket236 230 5386-5409
START_RANGE client30x8 socket237 230 5409-5432
START_RANGE client30y8 socket238 230 5432-5455
START_RANGE client30x8 socket239 230 5455-5478
START_RANGE client30y8 socket240 220 5478-5500
START_RANGE client31x8 socket241 230 5500-5523
START_RANGE client31y8 socket242 230 5523-5546
START_RANGE client31x8 socket243 230 5546-5569
START_RANGE client31y8 socket244 230 5569-5592
START_RANGE client31x8 socket245 230 5592-5615
START_RANGE client31y8 socket246 230 5615-5638
START_RANGE client31x8 socket247 230 5638-5661
START_RANGE client31y8 socket248 230 5661-5684
START_RANGE client32x8 socket249 230 5684-5707
START_RANGE client32y8 socket250 230 5707-5730
START_RANGE client32x8 socket251 230 5730-5753
START_RANGE client32y8 socket252 220 5753-5775
START_RANGE client32x8 socket253 230 5775-5798
START_RANGE client32y8 socket254 230 5798-5821
START_RANGE client32x8 socket255 230 5821-5844
START_RANGE client32y8 socket256 230 5844-5867
START_RANGE client33x9 socket257 230 5867-5890
START_RANGE client33y9 socket258 230 5890-5913
START_RANGE client33x9 socket259 230 5913-5936
START_RANGE client33y9 socket260 230 5936-5959
START_RANGE client33x9 socket261 230 5959-5982
START_RANGE client33y9 socket262 230 5982-6005
START_RANGE client33x9 socket263 230 6005-6028
START_RANGE client33y9 socket264 220 6028-6050
START_RANGE client34x9 socket265 230 6050-6073
START_RANGE client34y9 socket266 230 6073-6096
START_RANGE client34x9 socket267 230 6096-6119
START_RANGE client34y9 socket268 230 6119-6142
START_RANGE client34x9 socket269 230 6142-6165
START_RANGE client34y9 socket270 230 6165-6188
```

```
START_RANGE client34x9 socket271 230 6188-6211
START_RANGE client34y9 socket272 230 6211-6234
START_RANGE client35x9 socket273 230 6234-6257
START_RANGE client35y9 socket274 230 6257-6280
START_RANGE client35x9 socket275 230 6280-6303
START_RANGE client35y9 socket276 220 6303-6325
START_RANGE client35x9 socket277 230 6325-6348
START_RANGE client35y9 socket278 230 6348-6371
START_RANGE client35x9 socket279 230 6371-6394
START_RANGE client35y9 socket280 230 6394-6417
START_RANGE client36x9 socket281 230 6417-6440
START_RANGE client36y9 socket282 230 6440-6463
START_RANGE client36x9 socket283 230 6463-6486
START_RANGE client36y9 socket284 230 6486-6509
START_RANGE client36x9 socket285 230 6509-6532
START_RANGE client36y9 socket286 230 6532-6555
START_RANGE client36x9 socket287 230 6555-6578
START_RANGE client36y9 socket288 220 6578-6600
#endif
#if MASTER_NUM1
START_RANGE client37x10 socket289 230 6600-6623
START_RANGE client37y10 socket290 230 6623-6646
START_RANGE client37x10 socket291 230 6646-6669
START_RANGE client37y10 socket292 230 6669-6692
START_RANGE client37x10 socket293 230 6692-6715
START_RANGE client37y10 socket294 230 6715-6738
START_RANGE client37x10 socket295 230 6738-6761
START_RANGE client37y10 socket296 230 6761-6784
START_RANGE client38x10 socket297 230 6784-6807
START_RANGE client38y10 socket298 230 6807-6830
START_RANGE client38x10 socket299 230 6830-6853
START_RANGE client38y10 socket300 220 6853-6875
START_RANGE client38x10 socket301 230 6875-6898
START_RANGE client38y10 socket302 230 6898-6921
START_RANGE client38x10 socket303 230 6921-6944
START_RANGE client38y10 socket304 230 6944-6967
START_RANGE client39x10 socket305 230 6967-6990
START_RANGE client39y10 socket306 230 6990-7013
START_RANGE client39x10 socket307 230 7013-7036
START_RANGE client39y10 socket308 230 7036-7059
START_RANGE client39x10 socket309 230 7059-7082
START_RANGE client39y10 socket310 230 7082-7105
START_RANGE client39x10 socket311 230 7105-7128
START_RANGE client39y10 socket312 220 7128-7150
START_RANGE client40x10 socket313 230 7150-7173
START_RANGE client40y10 socket314 230 7173-7196
START_RANGE client40x10 socket315 230 7196-7219
START_RANGE client40y10 socket316 230 7219-7242
START_RANGE client40x10 socket317 230 7242-7265
START_RANGE client40y10 socket318 230 7265-7288
START_RANGE client40x10 socket319 230 7288-7311
START_RANGE client40y10 socket320 230 7311-7334
START_RANGE client41x11 socket321 230 7334-7357
START_RANGE client41y11 socket322 230 7357-7380
START_RANGE client41x11 socket323 230 7380-7403
START_RANGE client41y11 socket324 220 7403-7425
START_RANGE client41x11 socket325 230 7425-7448
START_RANGE client41y11 socket326 230 7448-7471
START_RANGE client41x11 socket327 230 7471-7494
START_RANGE client41y11 socket328 230 7494-7517
START_RANGE client42x11 socket329 230 7517-7540
START_RANGE client42y11 socket330 230 7540-7563
START_RANGE client42x11 socket331 230 7563-7586
START_RANGE client42y11 socket332 230 7586-7609
START_RANGE client42x11 socket333 230 7609-7632
START_RANGE client42y11 socket334 230 7632-7655
START_RANGE client42x11 socket335 230 7655-7678
START_RANGE client42y11 socket336 220 7678-7700
START_RANGE client43x11 socket337 230 7700-7723
START_RANGE client43y11 socket338 230 7723-7746
START_RANGE client43x11 socket339 230 7746-7769
START_RANGE client43y11 socket340 230 7769-7792
START_RANGE client43x11 socket341 230 7792-7815
START_RANGE client43y11 socket342 230 7815-7838
START_RANGE client43x11 socket343 230 7838-7861
START_RANGE client43y11 socket344 230 7861-7884
START_RANGE client44x11 socket345 230 7884-7907
START_RANGE client44y11 socket346 230 7907-7930
START_RANGE client44x11 socket347 230 7930-7953
START_RANGE client44y11 socket348 220 7953-7975
START_RANGE client44x11 socket349 230 7975-7998
START_RANGE client44y11 socket350 230 7998-8021
START_RANGE client44x11 socket351 230 8021-8044
START_RANGE client44y11 socket352 230 8044-8067
START_RANGE client45x12 socket353 230 8067-8090
START_RANGE client45y12 socket354 230 8090-8113
START_RANGE client45x12 socket355 230 8113-8136
START_RANGE client45y12 socket356 230 8136-8159
START_RANGE client45x12 socket357 230 8159-8182
START_RANGE client45y12 socket358 230 8182-8205
START_RANGE client45x12 socket359 230 8205-8228
START_RANGE client45y12 socket360 220 8228-8250
START_RANGE client46x12 socket361 230 8250-8273
START_RANGE client46y12 socket362 230 8273-8296
START_RANGE client46x12 socket363 230 8296-8319
START_RANGE client46y12 socket364 230 8319-8342
START_RANGE client46x12 socket365 230 8342-8365
START_RANGE client46y12 socket366 230 8365-8388
START_RANGE client46x12 socket367 230 8388-8411
START_RANGE client46y12 socket368 230 8411-8434
START_RANGE client47x12 socket369 230 8434-8457
START_RANGE client47y12 socket370 230 8457-8480
START_RANGE client47x12 socket371 230 8480-8503
START_RANGE client47y12 socket372 220 8503-8525
START_RANGE client47x12 socket373 230 8525-8548
START_RANGE client47y12 socket374 230 8548-8571
START_RANGE client47x12 socket375 230 8571-8594
START_RANGE client47y12 socket376 230 8594-8617
START_RANGE client48x12 socket377 230 8617-8640
START_RANGE client48y12 socket378 230 8640-8663
START_RANGE client48x12 socket379 230 8663-8686
START_RANGE client48y12 socket380 230 8686-8709
START_RANGE client48x12 socket381 230 8709-8732
START_RANGE client48y12 socket382 230 8732-8755
START_RANGE client48x12 socket383 230 8755-8778
START_RANGE client48y12 socket384 220 8778-8800
#endif
#if MASTER_NUM3
START_RANGE client49x13 socket385 230 8800-8823
START_RANGE client49y13 socket386 230 8823-8846
START_RANGE client49x13 socket387 230 8846-8869
START_RANGE client49y13 socket388 230 8869-8892
START_RANGE client49x13 socket389 230 8892-8915
START_RANGE client49y13 socket390 230 8915-8938
START_RANGE client49x13 socket391 230 8938-8961
START_RANGE client49y13 socket392 230 8961-8984
START_RANGE client50x13 socket393 230 8984-9007
START_RANGE client50y13 socket394 230 9007-9030
START_RANGE client50x13 socket395 230 9030-9053
START_RANGE client50y13 socket396 220 9053-9075
START_RANGE client50x13 socket397 230 9075-9098
START_RANGE client50y13 socket398 230 9098-9121
START_RANGE client50x13 socket399 230 9121-9144
START_RANGE client50y13 socket400 230 9144-9167
START_RANGE client51x13 socket401 230 9167-9190
START_RANGE client51y13 socket402 230 9190-9213
START_RANGE client51x13 socket403 230 9213-9236
START_RANGE client51y13 socket404 230 9236-9259
START_RANGE client51x13 socket405 230 9259-9282
START_RANGE client51y13 socket406 230 9282-9305
START_RANGE client51x13 socket407 230 9305-9328
START_RANGE client51y13 socket408 220 9328-9350
START_RANGE client52x13 socket409 230 9350-9373
START_RANGE client52y13 socket410 230 9373-9396
START_RANGE client52x13 socket411 230 9396-9419
START_RANGE client52y13 socket412 230 9419-9442
START_RANGE client52x13 socket413 230 9442-9465
START_RANGE client52y13 socket414 230 9465-9488
START_RANGE client52x13 socket415 230 9488-9511
START_RANGE client52y13 socket416 230 9511-9534
START_RANGE client53x14 socket417 230 9534-9557
START_RANGE client53y14 socket418 230 9557-9580
START_RANGE client53x14 socket419 230 9580-9603
START_RANGE client53y14 socket420 220 9603-9625
START_RANGE client53x14 socket421 230 9625-9648
START_RANGE client53y14 socket422 230 9648-9671
START_RANGE client53x14 socket423 230 9671-9694
START_RANGE client53y14 socket424 230 9694-9717
START_RANGE client54x14 socket425 230 9717-9740
START_RANGE client54y14 socket426 230 9740-9763
START_RANGE client54x14 socket427 230 9763-9786
START_RANGE client54y14 socket428 230 9786-9809
START_RANGE client54x14 socket429 230 9809-9832
START_RANGE client54y14 socket430 230 9832-9855
START_RANGE client54x14 socket431 230 9855-9878
START_RANGE client54y14 socket432 220 9878-9900
START_RANGE client55x14 socket433 230 9900-9923
START_RANGE client55y14 socket434 230 9923-9946
START_RANGE client55x14 socket435 230 9946-9969
START_RANGE client55y14 socket436 230 9969-9992
START_RANGE client55x14 socket437 230 9992-10015
START_RANGE client55y14 socket438 230 10015-10038
START_RANGE client55x14 socket439 230 10038-10061
START_RANGE client55y14 socket440 230 10061-10084
START_RANGE client56x14 socket441 230 10084-10107
START_RANGE client56y14 socket442 230 10107-10130
START_RANGE client56x14 socket443 230 10130-10153
START_RANGE client56y14 socket444 220 10153-10175
START_RANGE client56x14 socket445 230 10175-10198
START_RANGE client56y14 socket446 230 10198-10221
```

```
START_RANGE client56x14 socket447 230 10221-10244
START_RANGE client56y14 socket448 230 10244-10267
START_RANGE client57x15 socket449 230 10267-10290
START_RANGE client57y15 socket450 230 10290-10313
START_RANGE client57x15 socket451 230 10313-10336
START_RANGE client57y15 socket452 230 10336-10359
START_RANGE client57x15 socket453 230 10359-10382
START_RANGE client57y15 socket454 230 10382-10405
START_RANGE client57x15 socket455 230 10405-10428
START_RANGE client57y15 socket456 220 10428-10450
START_RANGE client58x15 socket457 230 10450-10473
START_RANGE client58y15 socket458 230 10473-10496
START_RANGE client58x15 socket459 230 10496-10519
START_RANGE client58y15 socket460 230 10519-10542
START_RANGE client58x15 socket461 230 10542-10565
START_RANGE client58y15 socket462 230 10565-10588
START_RANGE client58x15 socket463 230 10588-10611
START_RANGE client58y15 socket464 230 10611-10634
START_RANGE client59x15 socket465 230 10634-10657
START_RANGE client59y15 socket466 230 10657-10680
START_RANGE client59x15 socket467 230 10680-10703
START_RANGE client59y15 socket468 220 10703-10725
START_RANGE client59x15 socket469 230 10725-10748
START_RANGE client59y15 socket470 230 10748-10771
START_RANGE client59x15 socket471 230 10771-10794
START_RANGE client59y15 socket472 230 10794-10817
START_RANGE client60x15 socket473 230 10817-10840
START_RANGE client60y15 socket474 230 10840-10863
START_RANGE client60x15 socket475 230 10863-10886
START_RANGE client60y15 socket476 230 10886-10909
START_RANGE client60x15 socket477 230 10909-10932
START_RANGE client60y15 socket478 230 10932-10955
START_RANGE client60x15 socket479 230 10955-10978
START_RANGE client60y15 socket480 220 10978-11000
#endif
/*----------------------------------------------------------------------------*/

#define TES_FLAG_TRACE         0x00000010
#define TES_FLAG_KEYSTROKE_TIME 0x00000200
#define TES_FLAG_LOCAL_LOG     0x00000400
#define TES_FLAG_LOCAL_TRACE   0x00000800
#define TES_FLAG_LOCAL_IPRINT  0x00004000

#if 0
/* SETFLAG ALL TES_FLAG_TRACE */
SETFLAG ALL TES_FLAG_LOCAL_TRACE
SETFLAG ALL TES_FLAG_LOCAL_IPRINT
#endif

#if 0
SETFLAG client1x telnet 1 TES_FLAG_KEYSTROKE_TIME
#endif
```

D.2 user_master.C

```
/*****************************************************************************/
/* user_master.C                                  Audit: 05/30/96 */
/*****************************************************************************/

static char *rcsid="$Id: user_master.C,v 1.1 1999/02/22 06:31:05 channui Exp
$";

#include <iostream.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#define _H_CUR01
#include <cur00.h>
#undef _H_CUR01
extern "C" {
#include "data/cur01.h"
int wrefresh (WINDOW *);
int wclrtoeol(WINDOW *);
int setupterm(char*,FILE*,int*);
int nodelay(int);
int keypad(int);
int wgetch(WINDOW *);
}
#include "data/rte.h"
#include "data/Stats.h"
#include "data/misc.h"
#include "user_tpcc.h"

struct header_s {
   int slave;
   int num;
   int type;
   int num_timestamps;
   int user_data_length;
   int data_type;
};


char *get_variable(char *name);
int get_variable(char *name, int *number);
int send_global_data(void);
int make_ratios (double *buffer);
extern int ramp_up_complete;
extern int interval_start_time, interval_stop_time;
extern "C" int strcasecmp(char *s1, char *s2);
extern "C" int strncasecmp(char *s1, char *s2, int n);

struct UserSpawnData {
   int Warehouse;
   int District;
};

/* user_master.C */
int user_statistics_print(void);
// int user_spawn(int *length, char *buffer);
int user_spawn(int min, int max, int number, int *length, char *buffer);
int user_finished(int length, char *buffer);

extern SlaveStatus slave_status[MAX_SLAVES];

extern Stats status[MAX_TRAN_TYPE][MAX_TIMES];
extern WINDOW *statistics_win;
extern UserGlobal *shmglobal;

/* Transaction mix parameters */
double ratio_desired[6], ratio_min[6], ratio_max[6], ratio_range[6];
char *ratio_names[] = { "RTE", "NEWORDER", "PAYMENT", "ORDSTAT",
"DELIVERY",
                "STOCKLEV", NULL };
char *Status_Names[] = {"Menu", "Keying", "Response", "Think"};


char *transaction_names[] = { "RTE", "New Order", "Payment", "Order Stat",
                "Delivery", "Stock Level", NULL };

static int current_status = 2, status_needs_refresh = 1;

int user_statistics_print(void) {
   int i;
   static int count = 0;
   double ratios[6];
   if (status_needs_refresh) {
                count = 0;
                status_needs_refresh = 0;
                wmove   (statistics_win, 0, 0);
                wprintw (statistics_win, "%11s %8s %8s %8s %8s %8s %6s %6s
%6s",
                                Status_Names[current_status], "90%", "Avg", "Min",
"Max",
                                "Samples", "Ratio", "Mix", "Think");
   }
   make_ratios(ratios);


   for (i = 1; i <= 5; i++) {
                /* The reason we do this is because calculating the percentiles
                   is expensive */
                if (count % 10 == 0) {
                   wmove   (statistics_win, i, 0);
                   wprintw (statistics_win, "%11s %8.2f",
                                transaction_names[i],
status[i][current_status].ninety()/1000.0);
                   count = 0;
                }
                wmove   (statistics_win, i, 21);
                wprintw (statistics_win, "%8.2f %8.2f %8.2f %8d %6.2f %6.2f %6.2f",
                                status[i][current_status].average()/1000.0,
                                status[i][current_status].min()/1000.0,
                                status[i][current_status].max()/1000.0,
                                status[i][current_status].samples(),
                                ratios[i], shmglobal->chances[i],
                                status[i][3].average()/1000.0);
   }
   wmove   (statistics_win, 7, 0);
```

```
    extern int runtime_counts[MAX_TRAN_TYPE];
    extern int begin_time, ramp_up, run_time;
    int start = interval_start_time;
    int stop  = interval_stop_time;
    double interval = ((double)(stop-start) / (1000*60));
    double samples = status[1][2].samples();
    if (interval <= 0 || samples <= 0) {
                wprintw (statistics_win, "TPM-C: %7s / ", "-------");
    } else {
                wprintw (statistics_win, "TPM-C: %7.2f / ", samples/interval);
    }
    samples = runtime_counts[1];
    if (samples > 0) {
                start = begin_time+((ramp_up>=0)?ramp_up:0);
                if (run_time > 0 && stop > begin_time + ramp_up + run_time) {
                   stop = begin_time + ramp_up + run_time;
                }
                interval = (double)(stop - start)/(1000.0*60.0);
                wprintw (statistics_win, "%7.2f", samples/interval);
    } else {
                wprintw (statistics_win, "-------");
    }

    count++;
    return RTE_OK;
}

extern int login_begin;
int login_max_load;

const int MAX_WAREHOUSES=20000;
/* All of this 10 stuff is district size.  Should be a constant.
   Maybe fix that later */
int num_warehouses = -1;
int warehouses[MAX_WAREHOUSES*10];
int user_spawn(int min, int max, int number, int *length, char *buffer) {
//int user_spawn(int number, int *length, char *buffer) {
    int i, min_index;
    int adj_wh = num_warehouses;                        // adjusted warehouse
number
    UserSpawnData *ptr = (UserSpawnData *)buffer;
    *length = sizeof(*ptr);

//    min_index = 0;
//    for (i = 1; i < (num_warehouses)*10 && i < MAX_WAREHOUSES*10; i++) {
//
//  if both min and max are zero, running START, otherwise running
//                START_RANGE.  Must also determine what the ending warehouse
number
//                will be for said range
//

                if (min ==0 && max == 0) {
                            min++;
                            min_index =  0 ;
                } else {
                            adj_wh = max;   // inclusive range of wh-s
                            min = min * 10;
                            min_index = min;
                }
    for (i = min ; i < (adj_wh)*10 && i < MAX_WAREHOUSES*10; i++) {
                if (warehouses[i] < warehouses[min_index]) {
                   min_index = i;
                }
    }

    ptr->Warehouse = min_index / 10 + 1;
    ptr->District  = min_index % 10 + 1;
    warehouses[min_index]++;
/*    iprint (IPRINT_INFO, "Driver for Warehouse %d, District %d started.
warehouses[%d]++= %d\n",
          ptr->Warehouse, ptr->District, min_index, warehouses[min_index]);*/
    return RTE_OK;
}

int user_finished(int length, char *buffer) {
    UserSpawnData *ptr = (UserSpawnData *)buffer;
    int temp = (ptr->Warehouse-1)*10+ptr->District-1;
    warehouses[temp]--;
/*    iprint (IPRINT_INFO, "Driver for Warehouse %d, District %d died.
warehouses[%d]--= %d\n",
          ptr->Warehouse, ptr->District, temp, warehouses[temp]);*/
    return RTE_OK;
}


double limit(double min, double max, double val) {
    if (val < min)
                return min;
    if (val > max)
                return max;
    return val;
}


int make_ratios (double *buffer) {
    int neword   = status[NEWORDER][0].samples();
    int payment  = status[PAYMENT][0].samples();
    int ordstat  = status[ORDSTAT][0].samples();
    int delivery = status[DELIVERY][0].samples();
    int stocklev = status[STOCKLEV][0].samples();
    int total    = neword + payment + ordstat + delivery + stocklev;
    int i;

    if (total == 0) {
                buffer[NEWORDER] = 100.0;
                for (i = 2; i < 6; i++) {
                   buffer[i] = ratio_desired[i];
                   buffer[NEWORDER] -= buffer[i];
                }
                return 0;
    }

    buffer[PAYMENT]  = (double)payment / (double)total * 100.0;
    buffer[ORDSTAT]  = (double)ordstat  / (double)total * 100.0;
    buffer[DELIVERY] = (double)delivery / (double)total * 100.0;
    buffer[STOCKLEV] = (double)stocklev / (double)total * 100.0;
    buffer[NEWORDER] = 100.0- buffer[PAYMENT] - buffer[ORDSTAT] -
                                                  buffer[DELIVERY] -
buffer[STOCKLEV];

    return total;
}

int user_global_update(int *length, char *buffer) {
    UserGlobal *shmglobal = (UserGlobal *)buffer;
    static double last[6];
    static last_test_state = 0;
    static int users_last=-1;
    double ratios[6];
    double current[6];
    int i, different = 0;
    int desired = 0;
    int host_busy, all_zero;

    *length = sizeof(*shmglobal);

    make_ratios(ratios);

    /* Calculate ratios we want for next time */
    /* Note: we just keep on with the desired values until ramp-up is complete
          this at least starts us out without any humps or spikes in the
                graph */
    if (ramp_up_complete){
                current[NEWORDER] = 100.0;
                for (i = 2; i < 6; i++) {
                   if (ratio_desired[i] > ratios[i]) {
                               current[i] = ratio_max[i];
                   } else {
                               current[i] = 2*ratio_desired[i] - ratios[i];
                               if (current[i] < ratio_min[i])
                                  current[i] = ratio_min[i];
                   }
                   current[NEWORDER] -= current[i];
                }
    } else {
                for (i = 1; i < 6; i++) {
                   current[i] = ratio_desired[i];
                }
    }

    /* Add up all the users */
    /* This needs to be changed to be more transparent*/
    shmglobal->total_users = 0;
    for (i = 0; i < MAX_SLAVES; i++) {
                shmglobal->total_users += slave_status[i].active;
                desired              += slave_status[i].desired;
    }
    /* Count up number of warehouses we WANT to have*/
    if (num_warehouses < 0) {
                num_warehouses = (desired-1)/10+1;
    }
    shmglobal->max_warehouses = num_warehouses;

    host_busy = 0;
    all_zero = 1;
    for (i = 1; i <= 5; i++) {
                if (status[i][current_status].average() != 0) {
                   all_zero = 0;
```

```
                }
                if ( status[i][current_status].average()/1000.0 > login_max_load ) {
                    host_busy = 1;
                }
    }
    if (shmglobal->host_busy && all_zero) {
                host_busy = 1;
    }

    if (host_busy != shmglobal->host_busy) {
                shmglobal->host_busy = host_busy;
                different = 1;
    }

    for (i = 2; i < 6; i++) {
                if (current[i] != last[i])
                    different = 1;
    }

    if (last_test_state != shmglobal->test_state) {
                different = 1;
                last_test_state = shmglobal->test_state;
    }

    // Don't send if it's the same as last time
    if ( !different && shmglobal->total_users == users_last ) {
                return RTE_ERROR;
    }

    users_last = shmglobal->total_users;
    for (i = 1; i < 6; i++) {
                shmglobal->chances[i] = last[i] = current[i];
    }

    return RTE_OK;
}

int user_isbusy() {
    return shmglobal->host_busy;
}


int parse_array(char *string, int max, int *buffer) {
    int i, rc;
    char *ptr;
    char *temp = strdup(string);
    ptr = strtok(temp, ",");
    for (i = 0; ptr && i < max; i++) {
                rc = sscanf(ptr, "%d", &buffer[i]);
                if (rc < 1) {
                    free(temp);
                    return i;
                }
                ptr = strtok(NULL, ",");
    }
    free(temp);
    return i;
}

int parse_array(char *string, int max, double *buffer) {
    int i, rc;
    char *ptr;
    char *temp = strdup(string);
    ptr = strtok(temp, ",");
    for (i = 0; ptr && i < max; i++) {
                rc = sscanf(ptr, "%lf", &buffer[i]);
                if (rc < 1) {
                    free(temp);
                    return i;
                }
                ptr = strtok(NULL, ",");
    }
    free(temp);
    return i;
}


int user_init() {
    double  dbuffer[32];
    int   rc, i;
    char *ptr;

    if (get_variable("KEYSTROKE_SLEEP",  &shmglobal->keystroke_sleep) !=
RTE_OK) {
                shmglobal->keystroke_sleep = 0;
    }
    if (get_variable("LOGIN_TIMEOUT", &shmglobal->login_timeout) != RTE_OK)
{
                shmglobal->login_timeout = 120; /* 2 minutes */
    }
    if (get_variable("KEYSTROKE_PACKET_SIZE",
&shmglobal->keystroke_packet_size) != RTE_OK) {
                shmglobal->keystroke_packet_size = 0;
    }
    shmglobal->login_timeout *= 1000;
    if (get_variable("LOGIN_MAX_LOAD", &login_max_load) != RTE_OK) {
                login_max_load = 2;
    }
    if (get_variable("WAREHOUSES", &num_warehouses) != RTE_OK) {
                num_warehouses = -1;
    }
    if (get_variable("LASTC", &shmglobal->lastc) != RTE_OK) {
                shmglobal->lastc = 193; /* 2 minutes */
    }
    iprint(IPRINT_INFO, "Login Timeout = %s\n",
mstoa(shmglobal->login_timeout,0));
    iprint(IPRINT_INFO, "Keystroke Sleep = %s\n",
mstoa(shmglobal->keystroke_sleep*1000, 0));
    iprint(IPRINT_INFO, "Keystroke Packet Size= %d\n",
shmglobal->keystroke_packet_size);
    if (num_warehouses >= 0) {
                iprint(IPRINT_INFO, "Fixed Warehouses to = %d\n",
num_warehouses);
    }

    if (!(ptr = get_variable("NEWORDER"))) {
                iprint_error ("Error. NEWORDER variable not found\n");
                exit (1);
    }
    if (parse_array(ptr, 3, dbuffer)!=3) {
                iprint_error ("Error. NEWORDER should be think, emulex_menu,
emulex_response");
                exit (1);
    }
    shmglobal->think         [NEWORDER] = dbuffer[0];
    shmglobal->emulex_menu   [NEWORDER] = dbuffer[1];
    shmglobal->emulex_response[NEWORDER] = dbuffer[2];
    shmglobal->test_state = 0;

    for (i = 2; i < 6; i++) {
                if (!(ptr = get_variable(ratio_names[i])) ||
                    (parse_array(ptr, 6, dbuffer)!=6)) {
                    iprint(__FILE__, __LINE__, IPRINT_ERROR,
                        "Error. %s should be think, emulex_menu,
emulex_response, desired, min, max",
                                ratio_names[i]);
                    exit (1);
                }
                shmglobal->think[i]         = dbuffer[0];
                shmglobal->emulex_menu[i]    = dbuffer[1];
                shmglobal->emulex_response[i] = dbuffer[2];
                ratio_desired[i]            = dbuffer[3];
                ratio_min[i]                = dbuffer[4];
                ratio_max[i]                = dbuffer[5];
                ratio_range[i]              = ratio_max[i]-ratio_min[i];
    }

    return RTE_OK;
}

int user_extra_data(header_s *header) {
    int i;
    int num_timestamps;

    if (header->data_type != RTE_ITEM_KEYSTROKE_TIMES)
                return RTE_OK;
    int *times = (int *)((char *)header+sizeof(struct header_s));
    num_timestamps = header->user_data_length / 4 - 1;

    iprint (IPRINT_TRACE, "Keystroke times = ");
    for (i = 0 ; i < num_timestamps;i++) {
                iprint (IPRINT_TRACE, "%d ", times[i]);
    }
    iprint (IPRINT_TRACE, "\n", times[i]);

    return RTE_OK;
}

int user_process_command(char *command) {
    char buffer[256], *ptr;
    int i, found, len;
    strncpy (buffer, command, 256);
    ptr = strtok (buffer, " \t");
    found = 0;
    printf ("user_process_command('%s')\n",ptr);
    if (!strcasecmp (ptr, "pause")) {
                shmglobal->test_state = 1;
    } else if (!strcasecmp (ptr, "warmup")) {
                shmglobal->test_state = 2;
    } else if (!strcasecmp (ptr, "notest")) {
```

```
                shmglobal->test_state = 0;
   } else if (!strcasecmp (ptr, "login_max_load?")) {
                iprint (IPRINT_WARNING, "Current LOGIN_MAX_LOAD = %d\n",
login_max_load);
   } else if (!strncasecmp (command, "login_max_load=",15)) {
                login_max_load=atoi(command+15);
                iprint (IPRINT_WARNING, "Set LOGIN_MAX_LOAD = %d\n",
login_max_load);
   } else if (!strcasecmp (ptr, "display")) {
      while (ptr && (ptr = strtok(NULL, " \t"))) {
         if (*ptr == '\0')
            continue;
         for (i = 0; i < 5; i++) {
            len = min(strlen(Status_Names[i]), strlen(ptr));
            if (!strncasecmp (ptr, Status_Names[i], len)) {
               status_needs_refresh = found = 1;
               current_status = i;
               return RTE_OK;
            }
         }
         iprint (IPRINT_WARNING, "Unknown type to display:%s\n", ptr);
      }
   } else {
                iprint (IPRINT_WARNING, "Unknown Command:'%s'\n",
command);
                return RTE_ERROR;
   }
   return RTE_OK;
}

int transaction_process () {
   return RTE_OK;
}

int user_begin() {
   return RTE_OK;
}

void user_make_header(char *buffer) {
   int i;
   struct user_data_header *data = (struct user_data_header *)buffer;
}
```

D.3 user_slave.C

```
/*****************************************************************************/
/* user_slave.C                                  Audit: 05/30/96 */
/*****************************************************************************/

static char *rcsid="$Id: user_slave.C,v 1.1 1999/02/22 06:31:06 channui Exp $";

/*****************************************************************************/
/***              TPCC FILE FOR ALL USERS                                  ***/
/*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/time.h>
#include "rte_slave.h"
#include "user_tpcc.h"

/* This MUST match the corresponding one in client's inout.h file! */
#define TRIGGER "\021"
//#define NOSLEEP
// Increased EXPECT_TIMEOUT from 600000 - oz 10/20/97
#define EXPECT_TIMEOUT 6000000
#define KEYWAIT_FUDGE  5000

extern SHM_Slave *shm;
extern TableEntrySlave *shmentry;
extern DriverStatus    *status;
extern echo_trace(char *);
extern echo_trace();
extern char *expect_save;

const char *SQL_TPERRNO_MESSAGE = "tperrno";
const char *SQL_RTN_MESSAGE =     "rtn:";
const char *SQL_FATAL_MESSAGE =   "SQL Fatal Error";
const char *ROLLBACK_MESSAGE =    "Item number is not valid";

int           WHSEID;                 /* warehouse number for each users */

/*****************************************************************************/
/* The "uniform()" function has range of the absolute value of the       */
/* difference between the min. and the max values upto 2147483647.      */
/*****************************************************************************/
/*---------------------------------------------------------*/
/* NURand                        */
/*---------------------------------------------------------*/
/* A: 255 for C_LAST, 1023 for C_ID, 8191 for OL_I_ID   */
/* x: 0 for C_LAST, 1 for C_ID and OL_I_ID              */
/* y: 999 for C_LAST, 3000 for C_ID, 100000 for OL_I_ID */
/*---------------------------------------------------------*/
long
NURand(int A, int x, int y, long cval)
{
   return ((((long) uniform((long) 0, (long) A) | (long) uniform((long) x, (long) y))
+ cval) % (y - x + 1)) + x;
}

/*---------------------------------------------------------*/
/* getname                       */
/*---------------------------------------------------------*/
/* generates a random number from 0 to 999 inclusive    */
/* a random name is generated by associating a random   */
/* string with each digit of the generated number       */
/* three strings are concatenated to generate lastname  */
/*---------------------------------------------------------*/
char          *
getname()
{
   char          *last_name_parts[] =
   {
    "BAR",
    "OUGHT",
    "ABLE",
    "PRI",
    "PRES",
    "ESE",
    "ANTI",
    "CALLY",
    "ATION",
    "EING"
   };
   static char     lastname[128];
   int             random_num;

#if 1
   random_num = NURand(255, 0, 999, shmglobal->lastc);
#else
   random_num = NURand(255, 0, 999, LASTC);
#endif
   strcpy(lastname, last_name_parts[random_num / 100]);
   random_num %= 100;
   strcat(lastname, last_name_parts[random_num / 10]);
   random_num %= 10;
   strcat(lastname, last_name_parts[random_num]);
   return (lastname);
}

typedef struct gen_tran_s {
   int invalid;
   void *data;
   long len;
   long keywait;
   long type;
   char *menu;
   char *request;
} gen_tran_t;

int generic_transaction( gen_tran_t *data ) {
   char buffer[2048];
   static int consecutive_errs = 0;
   int rc;
   set_typing_delay(0);
   iprint(IPRINT_TRACE, "> generic_transaction sleep (%d)\n",data->type);
#ifndef NOSLEEP
   if (shmglobal->test_state == 0)
                transaction_sleep_do();
#endif

#ifdef EXPECT_TIMEOUT
   int timeout = EXPECT_TIMEOUT;
#else
   int timeout = 0;
#endif

   // Start the transaction (MENU)
   iprint(IPRINT_TRACE, "> generic_transaction start (%d)\n",data->type);
   transaction_start(data->type, data->len, data->data);

   iprint(IPRINT_TRACE, "> transmit data->menu\n");
   transmit(data->menu);
   echo_trace ("Waiting for Menu (DELIVERY)");
   if (expect(TRIGGER, timeout) == ERROR) {
                iprint (IPRINT_ERROR, "Slave %d: Failed to receive %sscreen\n",
                     shmentry->num, data->menu);
```

```
                return (ERROR);
    }
#ifndef NOSLEEP
    usleep(shmglobal->emulex_menu[data->type]*1000000.0+0.9);
#endif

    // Send our request (KEYING)
    transaction_mark(WHERE_NOW);
    echo_trace ("Keying");

#ifndef NOSLEEP
    usleep(data->keywait*1000000+KEYWAIT_FUDGE); // Keying delay
#endif
    // Wait for response (RESPONSE)
    transaction_mark(WHERE_NOW);

    iprint(IPRINT_TRACE, "> transmit data->request\n");
    transmit(data->request);

    echo_trace ("Wait for Response");
    if (expect(TRIGGER, timeout) == ERROR) {
                iprint (IPRINT_ERROR, "Slave %d: Failed to receive %s
response\n",
                                shmentry->num, data->menu);
                return (ERROR);
    }
#ifndef NOSLEEP
    usleep(shmglobal->emulex_response[data->type]*1000000.0+0.9);
#endif

    // Look for errors and set our think time (THINK)
    transaction_mark(WHERE_NOW);
    if (expect_after_match ("ERROR: ")) {
                data->invalid = 1;
                iprint (IPRINT_ERROR, "Slave %d: %s found'%s'\n",
                        shmentry->num, data->menu, "ERROR:");
                // Very dangerous, keep going rather than exiting...
                return RTE_ERROR;
                // Check for consecutive errors and if there are more than
                // 4 of them exit - allow for transient errors to make
                // tuning and testing easier   -oz
                // In either case the transaction is marked as invalid and
                // will be reported as an error by the analyze program.
                // if (consecutive_errs++ > 4)
                //    return RTE_ERROR;
    } else {
                consecutive_errs = 0;
    }
    echo_trace ("Thinking");
    transaction_sleep_set(neg_exp_4(shmglobal->think[data->type])*1000.0);
    iprint(IPRINT_TRACE, "< generic_transaction finish\n");
    return (RTE_OK);
}

/*****************************************************************************/
/***                  Delivery Transaction                                 ***/
/*****************************************************************************/
int
Delivery()
{
    static struct delivery_struct delivery, delivery_new;
    int           rc;
    char          *ptr;
    char          buffer[256];
    gen_tran_t    tran;

    tran.invalid = 0;
    tran.data    = &delivery;
    tran.len     = sizeof(delivery);
    tran.keywait = 2;
    tran.type    = DELIVERY;
    tran.menu    = "4";
    tran.request = buffer;

    // Set up all data for new transactions
    delivery_new.carrier = uniform(1, 10);                // carrier # 1 to 10

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\n", delivery_new.carrier);

    // Go do the transaction
    rc = generic_transaction(&tran);
    delivery        = delivery_new;
    delivery.invalid = tran.invalid;

    return (rc);
}

/*****************************************************************************/
/***                  New Order Transaction                                ***/
/*****************************************************************************/
int NewOrder() {
    static struct neword_struct neword, neword_new;
    int           i, rc, whses, low_whse=1;
    char          buffer[2048];
    char          *ptr;
    const char    *ptr2;
    gen_tran_t    tran;

    tran.invalid = 0;
    tran.data    = &neword;
    tran.len     = sizeof(neword);
    tran.keywait = 18;
    tran.type    = NEWORDER;
    tran.menu    = "1";
    tran.request = buffer;

    neword_new.rollback=0;

    /*** SECTION TO DETERMINE ROLLBACK TRANSACTION FOR 1% OF NEW
ORDERS ***/
    neword_new.did = uniform(1, 10);              // district number
    neword_new.cid = NURand(1023, 1, 3000, CUSTC);    // customer # 1 to 3000
    neword_new.nloop = uniform(5, 15);                // number of items to order
(5-15)
    neword_new.olremote=0;              // find total number of remote order-lines

    whses = shmglobal->max_warehouses;

    for (i = 0; i < neword_new.nloop; i++) {
                // Warehouse Number
                neword_new.item[i].olswid = WHSEID;
                if (whses > 1 && (uniform(0.0, 100.0) < 1.0)) {
                   /* for 1% of items (if *uniform()==0) */
                   /* Generate a uniform whse number that's different from WHSEID
*/
                   neword_new.item[i].olswid =
                            (long) uniform((long) low_whse, (long)whses-1);
                   if (neword_new.item[i].olswid >= WHSEID)
                            neword_new.item[i].olswid++;
                   neword_new.olremote++;  // find total number of remote
order-lines
                }
                // Item number 1-100000
                neword_new.item[i].oliid = NURand(8191, 1, 100000, ITEMC);
                // Quantity 1-10
                neword_new.item[i].olquantity = uniform(1, 10);
    }                                                   /* end of for n_loop */
    // We occasionally force a transaction to have invalid data to force a
    // rollback
    if (uniform(1, 5000) <= 50)
                neword_new.item[neword_new.nloop-1].oliid= 999999;

    neword_new.oremote = (neword_new.olremote > 0);

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\t%d",neword_new.did, neword_new.cid);
    for (i = 0; i < neword_new.nloop; i++) {
                ptr += sprintf(ptr, "\t%d\t%d\t%d",
                                        neword_new.item[i].olswid,
                                        neword_new.item[i].oliid,
                                        neword_new.item[i].olquantity);
    }
    ptr += sprintf(ptr, "\n");

    // Go do the transaction
    rc = generic_transaction(&tran);
    neword        = neword_new;
    neword.invalid = tran.invalid;

    // Check for a rollback
    if (expect_after_match (ROLLBACK_MESSAGE)) {
                neword.rollback=1;
                echo_trace ("Found rollback!\n");
    }

    // Grab the orderID from the
    if (!(ptr2 = expect_after_match("\033[6;15H"))) {
                echo_trace ("Didn't find order-id for neworder");
                iprint (IPRINT_ERROR, "Neworder didn't have Order-ID");
                neword.oid = -1;
    } else {
                neword.oid = atoi(ptr2+8);
    }

    // This is really not useful since we aren't going to be sending individual
    // keystrokes anymore
    if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME) {
```

```
                log_data(RTE_ITEM_KEYSTROKE_TIMES,
keystroke_length*sizeof(int),keystroke_times);
    }

    return (rc);
}


/*********************************************************************/
/***                Order Status Transaction                       ***/
/*********************************************************************/
int OrderStatus() {
    static struct ordstat_struct ordstat, ordstat_new;
    char            buffer[2048];
    int             rc;
    char           *ptr;
    gen_tran_t      tran;

    tran.invalid = 0;
    tran.data    = &ordstat;
    tran.len     = sizeof(ordstat);
    tran.keywait = 2;
    tran.type    = ORDSTAT;
    tran.menu    = "3";
    tran.request = buffer;

    // Set up all data for new transactions
    ordstat_new.did = uniform(1, 10);     /* district number 1 to 10 */
    if (uniform(1, 100) <= 60) {/* for 60% of transactions*/
                char *tmp = getname();
                strcpy(ordstat_new.clast, tmp);          /* by customer last name */
                if (ordstat_new.clast[0] < 'A' || ordstat_new.clast[0] > 'Z') {
                    iprint (IPRINT_ERROR,
                        "ASSERTION: OrderStatus getname() returns invalid name!
'%s'\n",
                                ordstat_new.clast);
                    return RTE_ERROR;
                }
                ordstat_new.byname = 1;
                ordstat_new.cid = 0;
    } else {
                ordstat_new.cid = NURand(1023, 1, 3000, CUSTC); /* cust. # 1 to
3000 */
                ordstat_new.byname = 0;
                ordstat_new.clast[0] = (char) NULL;
    }

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\t", ordstat_new.did);
    if (ordstat_new.byname){
                ptr += sprintf(ptr, "\t%s\n", ordstat_new.clast);
    } else {
                ptr += sprintf(ptr, "%d\n", ordstat_new.cid);
    }

    // Go do the transaction
    rc = generic_transaction(&tran);
    ordstat        = ordstat_new;
    ordstat.invalid = tran.invalid;

    return (rc);
}

/*********************************************************************/
/***                Payment Transaction                            ***/
/*********************************************************************/
int
Payment()
{
    static struct payment_struct payment, payment_new;
    int             dollars, cents, rc, whses, low_whse = 1;
    char            buffer[2048];
    char           *ptr;
    gen_tran_t      tran;

    tran.invalid = 0;
    tran.data    = &payment;
    tran.len     = sizeof(payment);
    tran.keywait = 3;
    tran.type    = PAYMENT;
    tran.menu    = "2";
    tran.request = buffer;

    payment_new.did = uniform(1, 10);  /* district number 1 to 10 */
    if (uniform(1, 100) <= 60) {/* for 60% of transactions*/
                strncpy(payment_new.clast, getname(), 17);    // by customer last
name
                if (payment_new.clast[0] < 'A' || payment_new.clast[0] > 'Z') {
                    iprint (IPRINT_ERROR,
                        "ASSERTION: payment_new getname() returns invalid name!
'%s'\n",
                                payment_new.clast);
                    return RTE_ERROR;
                }
                payment_new.byname = 1;
                payment_new.cid = 0;
    } else {
                payment_new.cid = NURand(1023, 1, 3000, CUSTC);                 /*
cust. # 1 to 3000 */
                payment_new.byname = 0;
                payment_new.clast[0] = (char) NULL;
    }

    whses = shmglobal->max_warehouses;

    if (whses < 2 || uniform(1, 100) <= 85) {/* for 85 % of transactions*/
                payment_new.cwid = WHSEID;
                payment_new.cdid = payment_new.did;
                payment_new.remote = 0;
    } else {                                    /* for 15 % of transactions */
                payment_new.cwid = (long) uniform((long)low_whse, (long)
whses-1);
                if (payment_new.cwid >= WHSEID)
                    payment_new.cwid++;

                payment_new.remote = 1;
                payment_new.cdid = uniform(1, 10);    /* district 1 to 10 */
    }

    dollars = uniform(1, 5000);             /* dollar amt = 1 to 5000 */
    if (dollars == 5000)
                cents = 0;
    else
                cents = uniform(0, 99);

    payment_new.amount= ((double) dollars) + ((double) cents) / 100.0;

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\t", payment_new.did);
    if (payment_new.byname){
                ptr += sprintf(ptr, "\t%s\t", payment_new.clast);
    } else {
                ptr += sprintf(ptr, "%d\t\t", payment_new.cid);
    }
    ptr += sprintf(ptr, "%d\t%d\t",payment_new.cwid, payment_new.cdid);
    ptr += sprintf(ptr, "%d.%02.2d\n",dollars, cents);

    // Go do the transaction
    rc = generic_transaction(&tran);
    payment        = payment_new;
    payment.invalid = tran.invalid;

    return (rc);
}

/*********************************************************************/
/***                Stock Level Transaction                        ***/
/*********************************************************************/
int
StockLevel()
{
    static struct stocklev_struct stocklevel, stocklevel_new;
    char            buffer[2048];
    int             rc;
    char           *ptr;
    gen_tran_t      tran;

    tran.invalid = 0;
    tran.data    = &stocklevel;
    tran.len     = sizeof(stocklevel);
    tran.keywait = 2;
    tran.type    = STOCKLEV;
    tran.menu    = "5";
    tran.request = buffer;

    stocklevel_new.invalid = 0;
    stocklevel_new.threshold = uniform(10, 20);       /* uniform no. between 10
and                                                                          *
20 */
    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\n", stocklevel_new.threshold);

    // Go do the transaction
    rc = generic_transaction(&tran);
    stocklevel        = stocklevel_new;
    stocklevel.invalid = tran.invalid;
```

```
    return (rc);
}


/*****************************************************************************/
/***  MAIN()                                                               ***/
/*****************************************************************************/
int
user_transaction()
{
    char          logout[32];
    double        ntask;
    int           resp;
    static int task = 0;

    if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME) {
                int rc;
                /* Wait for specified period of time */
                sleep (shmglobal->keystroke_sleep);
                /* Quit after one transaction */
                shm->lock(shmentry->pid);
                shmentry->flags |= TES_FLAG_DIE;
                shm->unlock(shmentry->pid);
                rc = NewOrder();
                iprint (IPRINT_INFO, "Slave %d: Keystroke timing setting die
flag\n", shmentry->num);
                return rc;
    }

#if 1
    switch (shmglobal->test_state) {
    case 0: // Normal
                break;
    case 1: // pause
                sleep (1);
                return RTE_OK;
    case 2: // warmup
                switch(task++) {
                case 0: return Delivery();
                case 1: return OrderStatus();
                case 2: return Payment();
                case 3: return StockLevel();
                case 4: task = 0; return NewOrder();
                }
    }
    /*****************************************************************************/
    /***  CHOOSE ONE OF THE TRANSACTIONS                                       ***/
    /*****************************************************************************/
    ntask = (double) uniform(0.0, 100.0);
    if (ntask <= shmglobal->chances[DELIVERY]) {
                return Delivery();
    }
    ntask -= shmglobal->chances[DELIVERY];
    if (ntask <= shmglobal->chances[ORDSTAT]) {
                return OrderStatus();
    }
    ntask -= shmglobal->chances[ORDSTAT];
    if (ntask <= shmglobal->chances[PAYMENT]) {
                return Payment();
    }
    ntask -= shmglobal->chances[PAYMENT];
    if (ntask <= shmglobal->chances[STOCKLEV]) {
                return StockLevel();
    }
    return NewOrder();
#else
    // this code should be shared between all of the users on a slave
    // int the best case it should be shared between all of the slaves,
    // but that would be too costly.
    // for now it is done on a per user basis.  If this thing is ever
    // modified to be threaded then it will probably go to the per-process
    // basis.  Although with shared memory, it would be possible to go to
    // per-slave.  Actually, before this code is put into use it must be
    // fixed up to share across processes.  Right now it will take, on average,
    // 22 minutes for one user to just key in the 100 entries.

    // use a card deck with no replacement to fulfill the requirements
    {
                int deck[100], count=-1, i, size=1, tmp;
                // lock deck
                if (count < 0) {
                   // deck is empty fill it up
                   count = 0;
                   for (i = 0; i < 43 * size; i++) {
                               deck[count++] = Payment;
                   }
                   for (i = 0; i < 4 * size; i++) {
                               deck[count++] = StockLevel;
                   }
                   for (i = 0; i < 4 * size; i++) {
                               deck[count++] = OrderStatus;
                   }
                   for (i = 0; i < 4 * size; i++) {
                               deck[count++] = Delivery;
                   }
                   for (; count < 100 * size; i++) {
                               deck[count++] = NewOrder;
                   }
                   // randomize the deck
                   for (i = 0; i < 100 * size; i++) {
                               int tmp;
                               int pick = uniform(i+1, 100);
                               tmp = deck[i];
                               deck[i] = deck[pick];
                               deck[pick] = tmp;
                   }
                }
                tmp = deck[count--];
                // unlock deck
                switch(tmp) {
                case Delivery:    return Delivery();
                case OrderStatus: return OrderStatus();
                case Payment:     return Payment();
                case StockLevel:  return StockLevel();
                case NewOrder:    return NewOrder();
                }
    }
#endif

#if 0
    if (resp != RTE_OK) {                   /* logoff if response is not correct */
                strcpy(logout, "9\n");      /* menu option 9 */
                transmit(logout);
                resp = expect("tpcc_cstux_inf:");
                return (ERROR);
    } else
                return (RTE_OK);
#endif
}                                                       /* end of main */

int user_parameter_change(void) {
    #if 0
    int i;
    iprint(IPRINT_TRACE, "Slave %d: total_users = %d\n",shmentry->num);
    iprint(IPRINT_TRACE, "Slave %d: chances     = ", shmentry->num);
    for (i = 0; i < MAX_TRAN_TYPE; i++)
                iprint(IPRINT_TRACE, "%6.2f ",shmglobal->chances[i]);
    iprint(IPRINT_TRACE, "\nSlave %d: think       = ",shmentry->num);
    for (i = 0; i < MAX_TRAN_TYPE; i++)
                iprint(IPRINT_TRACE, "%6.2f ",shmglobal->think[i]);
    iprint(IPRINT_TRACE, "\n");
    #endif
    return RTE_OK;
}

int user_login(char *user, char *password, void *data) {
    UserLocal *localdata = (UserLocal *)data;
    int rc;
    int timeout_value = shmglobal->login_timeout;
    char buffer[32];
    set_typing_delay(0);

    rc = expect   (TRIGGER, timeout_value);
    if (rc == RTE_ERROR) {
                iprint (IPRINT_ERROR, "Slave %d: didn't find Warehouse
prompt\n", shmentry->num);
    }
    sprintf(buffer, "%d\t%d\n",localdata->Warehouse, localdata->District);
    transmit(buffer);
    iprint (IPRINT_TRACE, "Slave %d: Warehouse=%d, District=%d, pid=%d\n",
shmentry->num, localdata->Warehouse, localdata->District, getpid());

    rc = expect (TRIGGER, timeout_value);
    if (rc != RTE_OK) {
       iprint (IPRINT_ERROR, "Slave %d: Failed logging in\n", shmentry->num);
                return RTE_ERROR;
    }
    return RTE_OK;
}

int user_init () {
    extern int expect_save_active;
    WHSEID = shmlocal->Warehouse;

    status->max_transmit =  shmglobal->keystroke_packet_size;
    expect_save_active = 1;
    return RTE_OK;
}
```

```
int user_logout () {
   transmit("9");
   iprint (IPRINT_TRACE, "Slave %d: Warehouse=%d, District=%d\n",
shmentry->num, shmlocal->Warehouse, shmlocal->District);
   return RTE_OK;
}

int user_cleanup () {
   transaction_sleep_do();
   transaction_start(0, 0, NULL); // Just something to clear out the buffer...
   return RTE_OK;
}

int user_spawn_ok() {
   int rc, hb;
   hb = ((UserGlobal *)(shm->global_data))->host_busy;
   rc = hb?RTE_ERROR:RTE_OK;
   return rc;
}
```

D.4 user_tpcc.h

```
/*************************************************************************/
/* user_tpcc.h                                    Audit: 05/30/96 */
/*************************************************************************/

/* $Id: user_tpcc.h,v 1.1 1999/02/22 06:31:06 channui Exp $ */

#ifndef USER_TPCC_H
#define USER_TPCC_H
/*************************************************************************/
/***   run-time constant for customer last name from 0 to 255,        ***/
/***   run-time constant for customer id from 0 to 1023,              ***/
/***   run-time constant for item id from 0 to 8191.                  ***/
/*************************************************************************/
/* #define     LASTC        117 */
/* Change for 3.1 */
#define        LASTC        193
#define        CUSTC        319
#define        ITEMC        3849

/*************************************************************************/
/***   response type
                ***/
/*************************************************************************/
/* #define OK                1  */
/* #define ERROR             -1 */

/*************************************************************************/
/***   transaction type
                ***/
/*************************************************************************/
#define        NEWORDER1
#define        PAYMENT                 2
#define        ORDSTAT                 3
#define        DELIVERY   4
#define        STOCKLEV  5

/*************************************************************************/
/***   transaction structures
                ***/
/*************************************************************************/
struct neword_struct {
   char          invalid;     /* transaction completed sucessfully */
   long          did;
   long          cid;
   long          oid;      /* Order-ID returned from client */
   long          nloop;       /* number of order line, avg = 15 */
   char          oremote;   /* 1 for remote order, 10%*/
   long          olremote;  /* number of remote order line, 1%*/
   char          rollback;   /* actually saw rollback text on screen */
   struct items_struct {
                 long          olswid;
                 long          oliid;
                 long          olquantity;
   } item[15];
};

struct payment_struct {
   char          invalid;     /* transaction completed sucessfully */
   long did;
   long cid;
   long cwid;
   long cdid;
   char clast[17];
   double amount;
   char byname;                                 /* 1 for by last name, 0 for by
id */
   char remote;                                 /* 1 for remote wardhouse, 0
otherwise */
};

struct ordstat_struct {
   char          invalid;     /* transaction completed sucessfully */
   long did;
   long cid;
   char clast[17];
   char byname;                                 /* 1 for by last name, 0 for by
id */
};

struct delivery_struct {
   char          invalid;     /* transaction completed sucessfully */
   char carrier;
};

struct stocklev_struct {
   char          invalid;     /* transaction completed sucessfully */
   long threshold;
};

struct generic_struct {
   char          invalid;     /* transaction completed sucessfully */
};

union transaction_info {
   char   invalid;
   struct generic_struct  generic;
   struct neword_struct   neword;
   struct payment_struct  payment;
   struct ordstat_struct  ordstat;
   struct delivery_struct delivery;
   struct stocklev_struct stocklev;
};

struct UserGlobal {
   int total_users;
   int max_warehouses;
   int keystroke_sleep;
   int login_timeout;
   int keystroke_packet_size;
   int lastc;
   int test_state;
   int host_busy;
   double chances[MAX_TRAN_TYPE];
   double think[MAX_TRAN_TYPE];
   double emulex_response[MAX_TRAN_TYPE];
   double emulex_menu    [MAX_TRAN_TYPE];
};

struct UserLocal {
   int Warehouse;
   int District;
};

struct user_data_header {
};

extern UserGlobal *shmglobal;
extern UserLocal  *shmlocal;
```

Appendix E: Third Party Quotes

June 24, 1999

Mr. Dee Prewit
IBM Corporation
11400 Burnet Road
Austin, TX 78758

Dear Mr. Prewit,

Thank you for the opportunity to quote the following RS/6000 S70 server systems. Attached is the hardware systems, software, and maintenance pricing for all items that we have been discussing.

Let's review the quote at your earliest convenience. The attached quotation is valid for sixty days from June 24, 1999.

Thank you for your time, and I look forward to speaking with you soon.

Sincerely,

Russ Turco

KeyLink Systems,
A Pioneer-Standard Company

RS/6000 Product Manager
800-448-6177 x7638
770-625-7644 (fax)
turcor@pios.com

KeyLink Systems Quote

Product	Description	Qty	Purchase Price	Maintenance (5 Years)
Server Hardware				
7017-S7A	RS/6000 Server Model S7A	5	75,180	244,800
2913	4.5 GB 1" Ultra SCSI Hot Swap	15	21,000	
2934	Async/Terminal Cable	5	225	
2968	10/100 Mbs Ethernet PCI Adapter	20	5,500	
4179	4 x 2048 MB R1 Memory	20	1,966,080	
5312	4way RS64 II 262 Mhz Proc, 8MB L2	10	750,000	806,400
5313	4way RS64 II 262 Mhz Proc, 8MB L2	5	375,000	201,600
6225	Advanced SerialRAID Adapter	130	390,000	
6320	I/O Drawer	20	475,320	
6321	Primary I/O Drawer Group	5	3,000	
6322	Support Processor Group	15	40,500	
6323	Secondary I/O Drawer Group	15	11,010	
3126	I/O Drawer Rack-Rack, Drawer Cables	5	15,875	
7000	I/O Rack	10	35,000	
7015-R00	System Rack Model R00	25	77,750	37,200
6171	Additional Power Distribution Unit	25	25,000	
7133-D40	SSA Disk Subsystem, Black Cover	144	1,872,000	1,382,400
8022	50/60 Hz AC, 300 VDC Power Supply	144	288,000	
8204	4.5 GB Disk Drive Modules	73	138,700	
8209	9.1 GB Disk Drive Modules	2,114	5,834,640	
8218	18 GB Disk Drive Modules	88	396,000	
8801	SSA Cables	576	54,720	
Client Hardware				
7043-140	RS/6000 Model 43P-140, 332 MHz	60	437,700	201,600
2900	4.5 GB SCSI Internal Disk	1	1,300	
4102	128MB DIMM Memory	60	26,880	
4115	128MB DIMM Memory Expansion	240	215,040	
2934	Async Terminal/Printer Cable	60	2,700	
2986	10/100 Mbps Ethernet Adapter, PCI	180	49,500	
User Connectivity				
3153-BG3	IBM ASCII Terminal, Keyboard	65	37,505	
	Hardware Subtotal		13,621,125	2,874,000
Server Software				
5765-C34	AIX 4.3 for S7A	5	1,500	
5765-C64	AIX C compiler	1	1,578	
5765-654	Performance Toolbox	1	1,300	
5765-A86	HACMP Conc Resource Mgr	5	100,000	
Client Software				
5756-C34	AIX 4.3.2 Unlimited Users	60	285,000	
5765-C64	AIX C compiler	1	1,578	
5697-D17	IBM TXSeries 4.2 for AIX	60	201,156	70,500
	Software Subtotal		592,112	70,500
	System Totals			
	RS/6000		14,213,237	2,944,500
	Oracle Version 8		1,850,480	1,850,480
	NBase 16port Ethernet Switches	9	14,040	2,808
	CentreCom Ethernet 8port Hubs w 10% Spares	15180	546,480	
	System Total		16,624,237	4,797,788
	System Total with Support			21,422,025
	KeyLink Systems' Solution excluding tax/ship			13,521,883

This price is valid for 60 days from June 24, 1999. Thank you for your consideration.

June 21, 1999

DataComm Warehouse

1720 OAK ST.
LAKEWOOD NJ., 08701
800-328-2261
(Ext 20286)
FAX: 732-363-4823

Attn: Dee Prewit

PLEASE ACCEPT OUR QUOTATION FOR THE FOLLOWING ITEMS.

ITEM#	QTY	DESCRIPTION	UNIT	TOTAL
DEH2924	15,180	Compex 8 port10baset hub	36	546,480

Please note, prices are valid for 60 days (subject to availability). Thank you for your consideration.

Sincerely,

Claudia Moore
Datacomm Warehouse
800-328-2261 ext20286
732-363-4823 FAX

Quotation No.
Terms: **Net 30 days**
Quote Duration: **Sixty (60) days**
FOB: **FACTORY**
Date: **6/24/1999**

To:
D Prewit
IBM
,

From:
NBase-Xyplex Rep: **Tom Sherer**
Phone: **713-840-6086**
Fax: **713-840-6087**

Product Quotation

Product Number	Product Description	Unit Price	Qty	Disc	Total Price
NH2032	Powergroup Switch with 16 10/100Base-TX ports with 4 expansion slots.	$2,600.00	9	40%	$14,040.00
				Total Configuration Cost	**$14,040.00**
			Note: If we are favored with your order it should be made out to: Xyplex, Inc 3200 Southwest Frwy Suite 300 FAX: 713-840-6087		

Note: This quotation is submitted for your convenience, and is subject to acceptance by NBase-Xyplex Corporate. Prices apply to these quantities only. NBase-Xyplex Deliveries will be quoted upon receipt of order. All NBase-Xyplex products are warranteed for 12 months from receipt of delivery.

By_____
Tom Sherer