IBM RISC System/6000 Enterprise Server H70 c/s

using

Oracle8 Enterprise Edition 8.1.5

IBM TXSeries 4.2 for AIX

TPC Benchmark™ C

Full Disclosure Report

IBM System Performance and Evaluation Center

Submitted For Review
May 27, 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 May 27, 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 H70 c/s

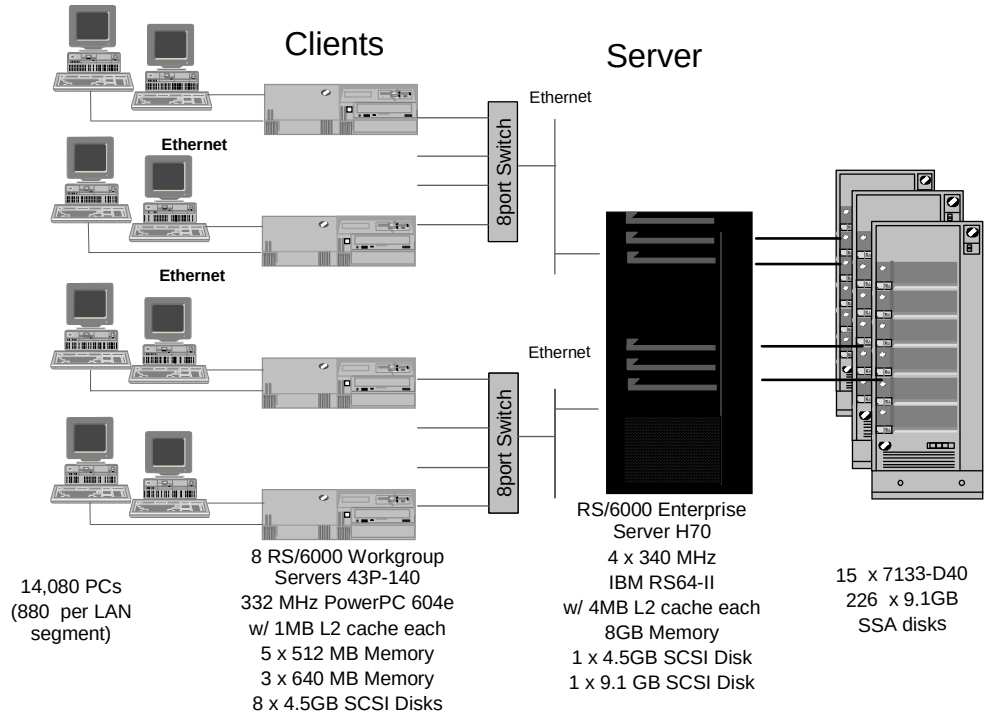
TPC-C Rev. 3.4

Report Date: May 27, 1999

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$1,343,526	**17,133.73 tpmC**	**$78.50/tpmC**	**Nov 19, 1999**

Processors	Database Manager	Operating System	Other Software	No. Users
4 x IBM RS64-II	**Oracle Version 8.1.5**	**AIX 4.3.2**	**TXSeries 4.2 for AIX**	**14,080**

RS/6000 Enterprise Server H70



System Components	Clients		Server	
	Quantity	Description	Quantity	Description
Processor	**8**	**332 MHz PowerPC 604e w/ 1MB L2 cache each**	**4**	**340 MHz IBM RS64-II w/ 4MB L2 cache each**
Memory	**5 x 512, 3x640 MB**		**8 GB**	
Disk Controllers	**8**	**SCSI-2 Adapters**	**2** **4**	**SCSI-2 Adapters** **SSA Adapters**
Disk Drives	**8**	**4.5 GB SCSI**	**226** **1** **1**	**9.1 GB SSA Disks** **4.5 GB SCSI Disk** **9.1 GB SCSI Disk**
Total Storage	**4.5 GB each client**		**1,884.41 GB**	
Terminals	**8**	**System Console**	**1**	**System Console**
Term. Connect	**1,936**	**CentreCOM 8-Port Ethernet Hubs + 10% spares**		

IBM RS/6000 Enterprise Server H70 c/s

TPC-C Rev. 3.4

Report Date:May 27, 1999

Description	Part Number	"Source	Unit Price	Qty	Extended Price	5 yr. Maint Price
Server Hardware						
RS/6000 Server Model H70	7026-H70	1	17,322	1	17,322	7,872
2 SCSI Adapters, Intg Enet, Rack, PDU, CD-ROM, SCSI Hot Swap 6-pack						
4.5 GB 1" SCSI Disk	2900	1	1,300	1	1,300	0
9.1 GB 1" Ultra SCSI Hot Swap	2913	1	1,750	1	1,750	0
Async/Terminal Cable	2934	1	45	1	45	0
10/100 Mbs Ethernet PCI Adapter	2968	1	275	2	550	0
Memory Expansion Feature	4098	1	1,038	2	2,076	0
512 MB SDRAM DIMMs	4119	1	6,400	16	102,400	0
2way RS64 II 340 Mhz Processor	4319	1	20,000	2	40,000	38,400
Advanced SerialRAID Adapter	6225	1	3,000	4	12,000	0
32 MB Fast-Write Cache Option	6235	1	575	1	575	0
Prestige EXT UPS, Rack Mount	9910-U33	1	3,822	2	7,644	0
System Rack Model S00	7015-S00	1	3,500	3	10,500	5,040
Additional Power Distribution Unit	6171	1	1,000	3	3,000	0
SSA Disk Subsystem, Black Cover	7133-D40	1	13,000	15	195,000	144,000
50/60 Hz AC, 300 VDC Power Supply	8022	1	2,000	15	30,000	0
9.1 GB Disk Drive Modules	8209	1	2,760	226	623,760	0
SSA Cables	8801	1	60	30	1,800	0
			Subtotal		1,049,722	195,312
Server Software						
AIX 4.3 for H70	5765-C34	1	175	1	175	0
AIX C compiler	5765-C64	1	1,578	1	1,578	0
Performance Toolbox	5765-654	1	1,300	1	1,300	0
Oracle Version 8.1.5		2	153,520	1	153,520	153,520
			Subtotal		156,573	153,520
Client Hardware						
RS/6000 Model 43P-140, 332 MHz	7043-140	1	7,295	8	58,360	26,880
4.5gb Disk, Intg SCSI-2 FW, Intg Enet, 1MB L2 cache						
128MB DIMM Memory	4102	1	448	8	3,584	0
128MB DIMM Memory Expansion	4115	1	896	27	24,192	0
Async Terminal/Printer Cable	2934	1	45	8	360	0
10/100 Mbps Ethernet Adapter, PCI	2986	1	275	16	4,400	0
IBM ASCII Terminal, Keyboard	3153-BG3	1	577	9	5,193	5,940
			Subtotal		96,089	32,820
Client Software						
AIX 4.3 Unlimited Users	5756-C34	1	4,750	8	38,000	0
AIX C compiler	5765-C64	1	1,578	1	1,578	0
IBM TXSeries 4.2 for AIX	5697-D17	1	3,353	8	26,821	9,400
			Subtotal		66,399	9,400
User Connectivity						
8port Ethernet Switch+ 2 Spare	Nbase	4	2,166.50	4	8,666	1,486
Ethernet Hub (8port) + 10% Spares	DEH2924	3	32.00	1936	61,952	0
			Subtotal		70,618	1,486
			Discounts		(415,955)	(70,972)
				Total	1,023,446	321,566

Notes:

Pricing Sources:
1=IBM, 2=Oracle, 3=Data Comm Warehouse, 4=NBase-Xyplex
Audited by: Francois Raab , Information Sizing

Five-Year Cost of Ownership:

	1,345,012
tpmC	17,133.73
$/tpmC	78.50

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 H70

MQTH, computed Maximum Qualified Throughput: 17,133.73 tpmC

Repeatability Run: 17,036.4 tpmC 0.57% difference

Response Times (in seconds)	90th %	Average	Maximum
New Order	2.46	1.14	36.85
Payment	2.24	1.03	36.54
Order-Status	2.26	1.06	36.72
Delivery (interactive portion)	0.20	0.12	0.65
Delivery (deferred portion)	0.45	0.28	2.93
Stock-Level	3.10	1.69	37.43
Menu	0.01	0.00	0.40

Transaction Mix, in percent of total transactions	Percent
New Order	44.78%
Payment	43.07%
Order-Status	4.06%
Delivery	4.02%
Stock-Level	4.05%

Keying/Think Times (in seconds)	Min.	Average	Max.
New Order	18.00/0.01	18.01/12.04	18.12/120.31
Payment	3.00/0.01	3.01/12.06	3.11/120.31
Order-Status	2.00/0.01	2.01/10.00	2.11/100.30
Delivery	2.00/0.01	2.01/5.01	2.09/50.30
Stock-Level	2.00/0.01	2.01/5.07	2.10/50.30

Test Duration	
Ramp-up Time	42 min 06 sec
Measurement interval	30 minutes
Transactions during measurement interval (all types)	1,147,627
Ramp-down time	10 minutes
Checkpointing	
Number of checkpoints	1
Checkpoint interval	30

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 IBM RISC System/6000 Enterprise Server Model H70. The phrase RS/6000 will be substituted for RISC System/6000 for the remainder of this document.

The software used on the RS/6000 Enterprise Server H70 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 H70

Company Name	System Name	Data Base Software	Operating System Software
IBM Corporation Oracle Corporation	RS/6000 Enterprise Server H70	Oracle8 Enterprise Edition 8.1.5	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 in transactions per minute	Total system cost/tpmC
$1,343,526	**17,133.73 tpm-C**	**$78.50 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 the IBM RS/6000 Enterprise server H70 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 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.*

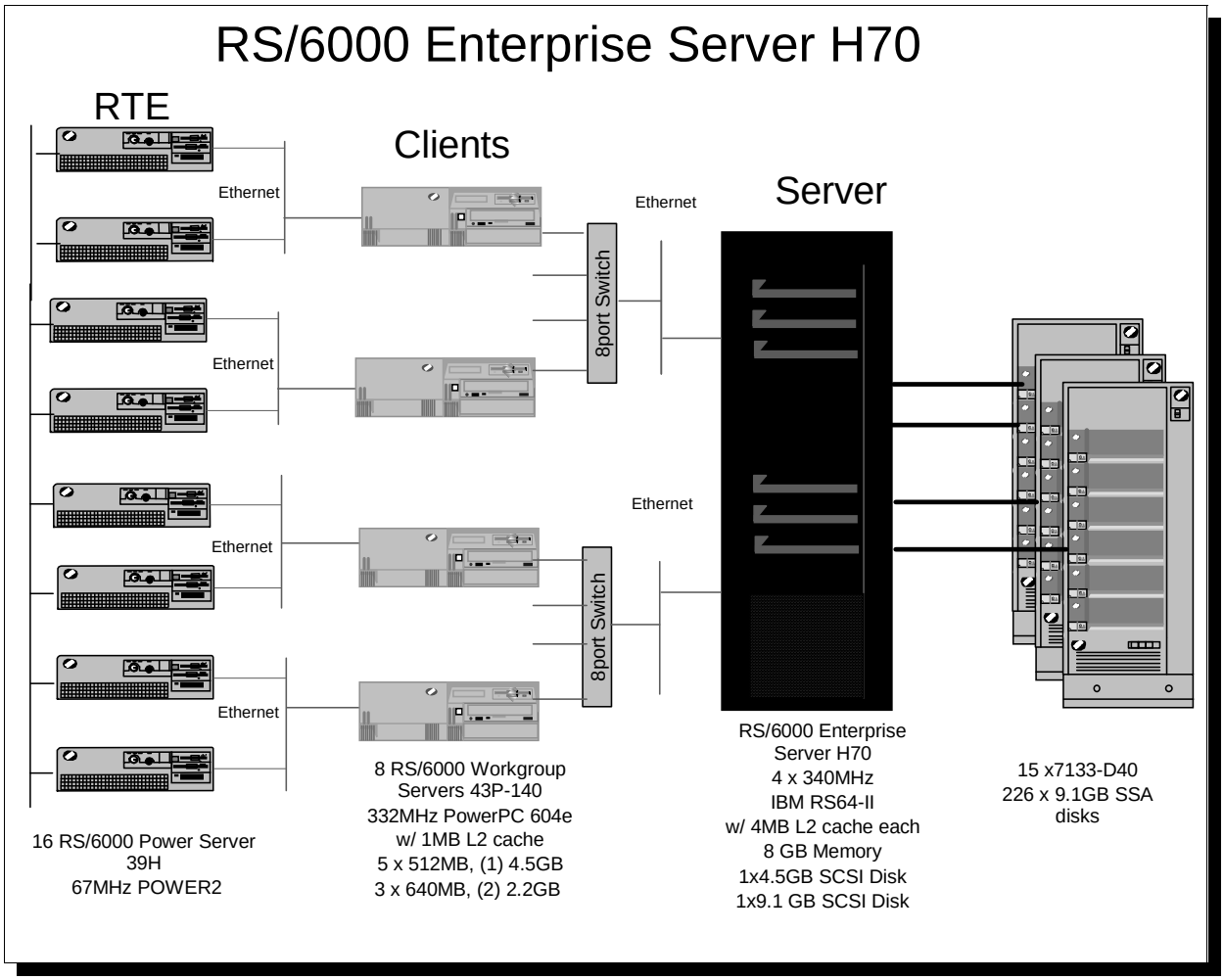
Appendix B contains the system, data base, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

1.4 Configuration Diagrams

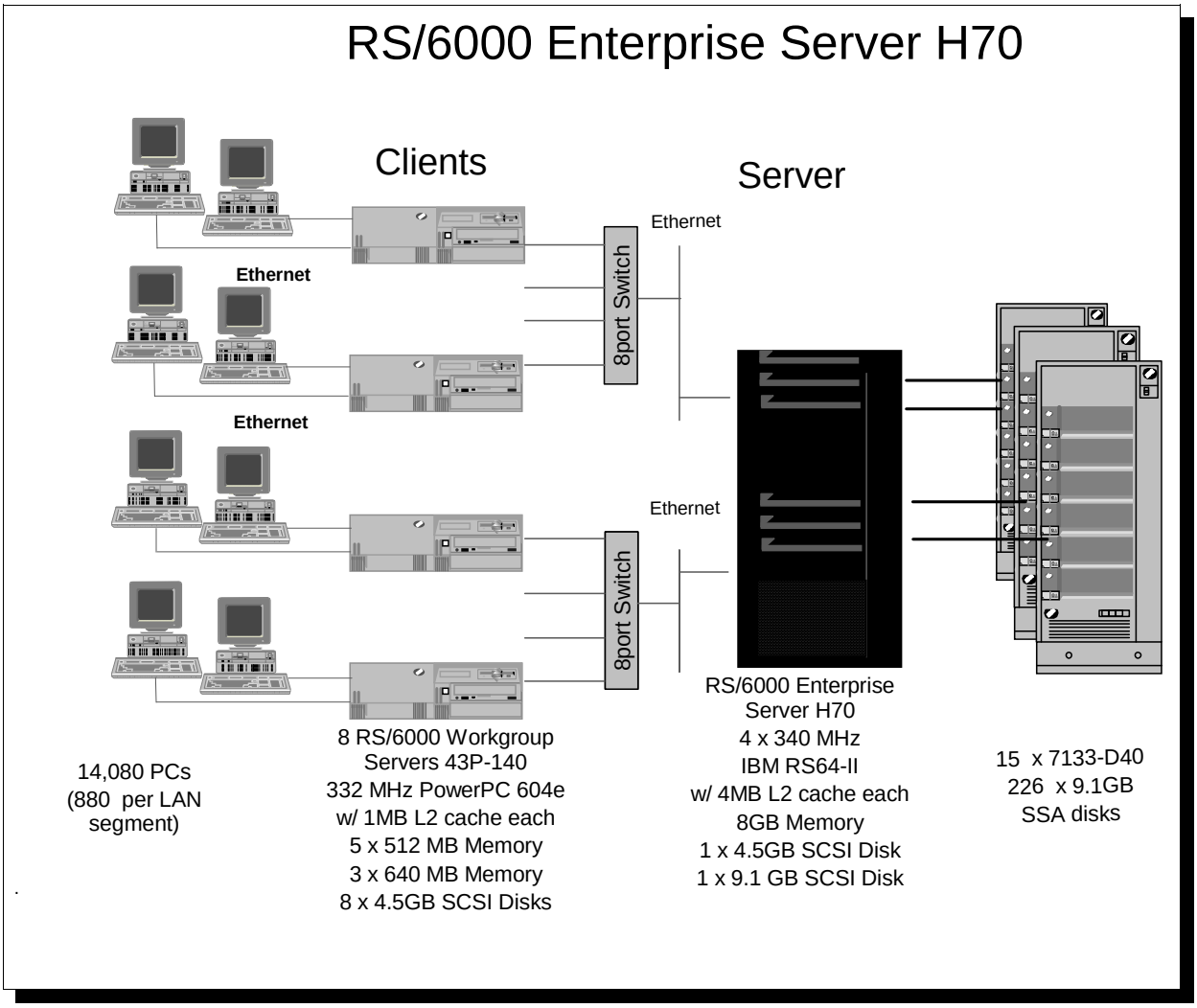
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *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 H70 Benchmark Configuration



RISC System/6000 Enterprise server H70 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.

Partitioning was not used for any of the measurement reported in this full disclosure.

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-100s, 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-100 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 H70
Percentage of Home order lines	99.0%
Percentage of Remote order lines	1.0%
Percentage of Rolled Back Transactions	0.99%
Average Number of Items per order	10
Payment	
Percentage of Home transactions	84.98%
Percentage of Remote transactions	15.02%
Non-Primary Key Access	
Percentage of Payment using C_LAST	59.97%
Percentage of Order-Status using C_LAST	59.92%
Delivery	
Delivery transactions skipped	0
Transaction Mix	
New-Order	44.78%
Payment	43.07%
Order-Status	4.06%
Delivery	4.02%
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 at full load for approximately sixty minutes 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.

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 raid disk, Oracle8 continued to process the transactions successfully.
4. The test continued for another 1 1/2 minutes.
5. The system was immediately shut down by switching the Emergency Power Off , thereby removing system power.
6. The disk from step 3 was powered back on.
7. The system was powered back on and rebooted.
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.

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. The disk containing the TPCC table was powered off.
5. The run was stopped.
6. The disk from step 4 was powered back on and was restored from the backup copy in step 1.

7. Oracle8 was restarted and its transaction log was used to roll forward through the transactions that had completed since the run had started.
8. 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.
9. Consistency condition 3 was verified.

Failure of Durable Fast Write Cache on SSA Adapter for Redo Logs

The following steps were successfully performed to pass the Durability test for failure of a durable medium that contains transient redo log transactions:

1. The SSA adapter for the Redo logs contains a cache that is powered by an onboard battery which will retain its contents if the adapter fails, or system power goes off. This test was performed in two parts:

   A) failure of the adapter/system power

   B) failure of the onboard battery
2. Test (A) was performed with the power-off test of the log above. After the system was powered off, the cache was removed from its current adapter and re-inserted into the system before rebooting.
3. Data on the cache was recovered successfully by meeting the requirements listed in the power-off test of the logs above.
4. Test (B). 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.
5. Test (B) was conducted by inducing a battery failure during a test run. Five minutes into the measurement interval, the battery failed. The system recorded the failure, flushed out its vram contents and quit using the cache. Error notices were posted into the system error log.
6. The run continued without the cache.
7. Step 4 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.
8. 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. While 1,450 warehouses were built initially, only 1,408 warehouses were used during the tests. The unused warehouses were deleted.

Table 5-1 Initial Cardinality of Tables (RS/6000 H70)

Table Name	Number of Rows
Warehouse	1,408
District	14,500
Customer	43,500,000
History	43,500,000
Order	43,500,000
New Order	13,050,000
Order Line	434,965,466
Stock	145,000,000
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 H70 Data Distribution Benchmark Configuration

Controller	Disk	Contents	Capacity
scsi	hdisk0	OS	4.5 GB
scsi	hdisk1	Oracle	9.1 GB
ssa	hdisk2-3	Paging	9.1 GB
ssa	hdisk4-5	i-Order	9.1 GB
ssa	hdisk6-25	Customer	9.1GB
ssa	hdisk26-57	Stock	9.1GB
ssa	hdisk58	TPC-C App Controls	9.1 GB
ssa	hdisk59-60	New Order	9.1GB
ssa	hdisk61-62	History	9.1GB
ssa	hdisk63-64	Orders	9.1GB
ssa	hdisk65	Rollbacks	9.1GB
ssa	hdisk71-76	Order Line	9.1GB
ssa	hdisk90-91	Order Index	9.1GB
ssa	hdisk98-116	Customer	9.1GB

Controller	Disk	Contents	Capacity
ssa	hdisk117-148	Stock	9.1GB
ssa	hdisk149-154	Customer, Order Indexes	9.1GB
ssa	hdisk155-164	Order Line	9.1GB
ssa	hdisk165-166	History	9.1GB
ssa	hdisk167-168	New Orders	9.1GB
ssa	hdisk194-212	Customer	9.1GB
ssa	hdisk213-244	Stock	9.1GB
ssa	hdisk245-254	i-Order	9.1GB
ssa	hdisk255-256	District, Warehouse, Item, Item Index	9.1GB
ssa	hdisk258	Log RAID	34.69GB
ssa	hdisk262	Log RAID	34.69GB
ssa	hdisk268	Log RAID	34.69GB
ssa	hdisk270	Log RAID	34.69GB
ssa	hdisk286	Log RAID	34.69GB

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 Oracle8i Enterprise Edition 8.1 from Oracle Inc. Oracle8 Enterprise Edition 8.1 is a relational DBMS.

5.4 Partitions/Replications Mapping

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

IBM did not implement horizontal or vertical partitioning for this TPC-C test.

5.5 180 day space calculations

RS/6000 Enterprise Server H70

TPM	17,133.73							
Warehouses	1,450.00							
SEGMENT	**TYPE**	**TSPACE**	**BLOCKS**	**FIVE_PCT**	**DAILY_GROW**	**TOTAL**		
CUSTOMER	TABLE	CUST	10,875,003.00	543,750.15	0.00	11,418,753.15		
DISTRICT	TABLE	MISC	7,252.00	362.60	0.00	7,614.60		
HISTORY	TABLE	HIST	742,359.00	0.00	140,351.76	882,710.76		
ICUSTOMER	INDEX	UNUSED	256,000.00	12,800.00	0.00	268,800.00		
ICUSTOMER2	INDEX	ICUST2	512,000.00	25,600.00	0.00	537,600.00		
IDISTRICT	INDEX	UNUSED	25,600.00	1,280.00	0.00	26,880.00		
IITEM	INDEX	MISC	25,600.00	1,280.00	0.00	26,880.00		
INORD	INDEX	NORD	51,200.00	2,560.00	0.00	53,760.00		
IORDERS	INDEX	IORD1	409,600.00	20,480.00	0.00	430,080.00		
IORDERS2	INDEX	IORD2	512,000.00	25,600.00	0.00	537,600.00		
IORDL	INDEX	ORDL	7,910,400.00	0.00	1,495,554.84	9,405,954.84		
ISTOCK	INDEX	UNUSED	1,024,000.00	51,200.00	0.00	1,075,200.00		
ITEM	TABLE	MISC	3,031.00	151.55	0.00	3,182.55		
IWAREHOUSE	INDEX	UNUSED	25,600.00	1,280.00	0.00	26,880.00		
ORDERS	TABLE	ORDR	429,319.00	0.00	81,167.84	510,486.84		
ROLL_SEG	SYS	ROLL	24,320.00	0.00	0.00	24,320.00		
STOCK	TABLE	STOCK	13,181,822.00	659,091.10	0.00	13,840,913.10		
SYSTEM	SYS	SYSTEM	98,048.00	0.00	0.00	98,048.00		
WAREHOUSE	TABLE	MISC	726.00	36.30	0.00	762.30		
Total			36,113,880.00	1,345,471.70	1,717,074.45	39,176,426.15		
Dynamic space		9,082,078						
Static space		28,377,274						
Free space		1,717,074						
Daily growth		1,717,074						
Daily spread		0	Oracle may be configured such that daily spread is 0					
180-day space (blk.)		337,450,675						
Block size (bytes)		4,096						
180-day (GB)		1,287.27						
			new_order	1,128,755.00				
Log block size		512	Redo blocks writte	34,992,617.00				
Log blocks/tpmC		31.00	Number of log blocks used in one tpmC					
8-hour log (GB)		121.57						
Disk	**Disk**	**SUT**	**SUT**	**Priced**	**Priced**		Space usage (GB)	
Type	**Formatted Capa**	**# of disks**	**Capacity(GB)**	**# of disks**	**Capacity(GB)**		180-day	**1,287.27**
							RAID	**127.19**
4.5	4,296.00	1.00	4.20	**1.00**	**4.20**		os+paging	**3.00**
9.1	8,672.00	207.00	1,753.03	**207.00**	**1,753.03**		Paging	**16.00**
RAID(4-9.1GB)	26,048.00	5.00	127.19	**20.00**	**127.19**		Total Space	**1,433.46**
				228.00	**1,884.41**			

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 H70 Response, Think and Keying Times

Response Times	New Order	Payment	Order Status	Delivery (int./def.)	Stock Level	Menus
90 %	2.46	2.24	2.26	0.20/0.45	3.1	0.01
Average	1.14	1.03	1.06	0.12/0.28	1.69	0
Maximum	36.85	36.54	36.72	0.65/2.93	37.43	0.4
			Think Times			
Minimum	0.01	0.01	0.01	0.01	0.01	N/A
Average	12.04	12.06	10	5.01	5.07	N/A
Maximum	120.31	120.31	100.3	50.3	50.3	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	18.12	3.11	2.11	2.09	2.1	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 H70 New-Order Response Time Distribution

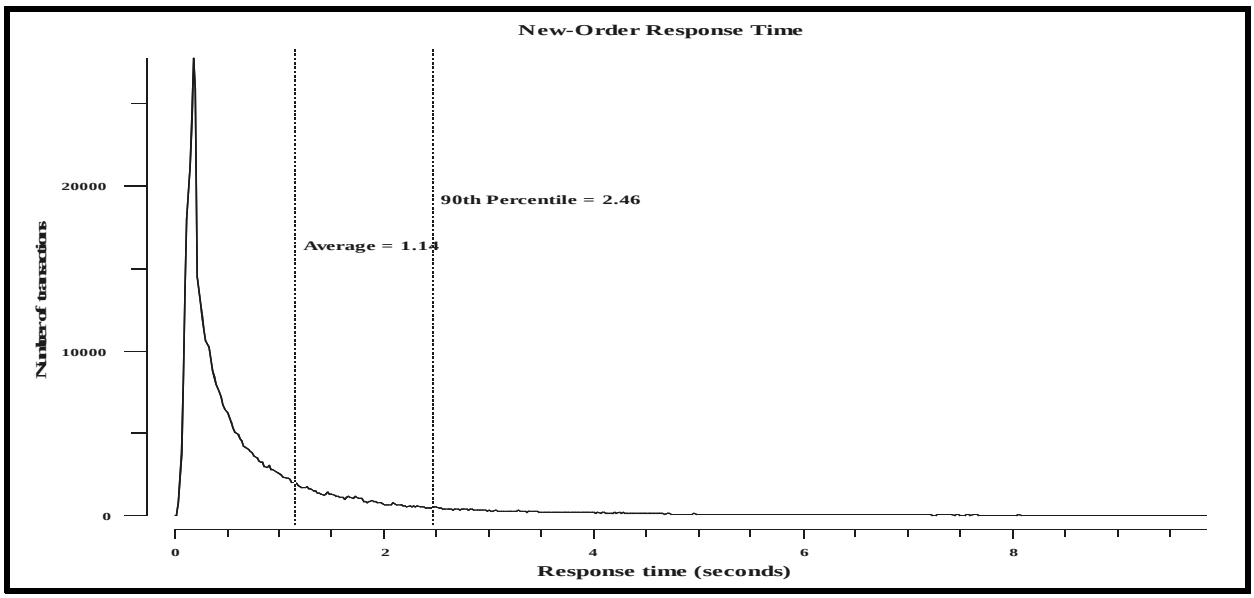


Figure 6-3-2. RS/6000 H70 Payment Response Time Distribution

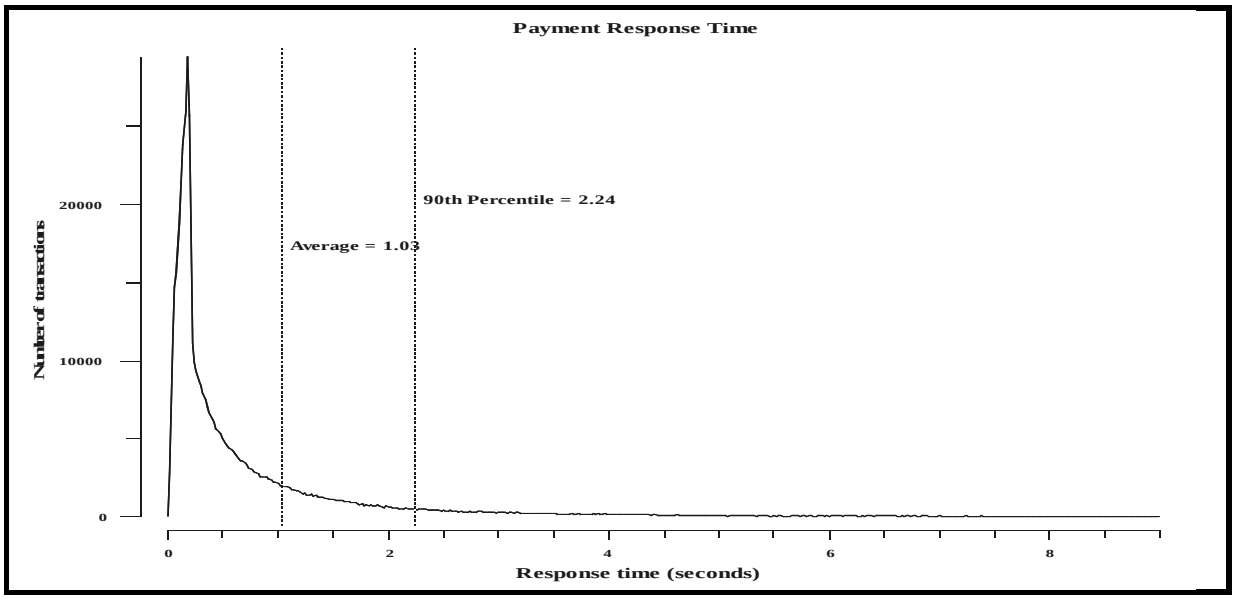


Figure 6-3-3. RS/6000 H70 Order-Status Response Time Distribution

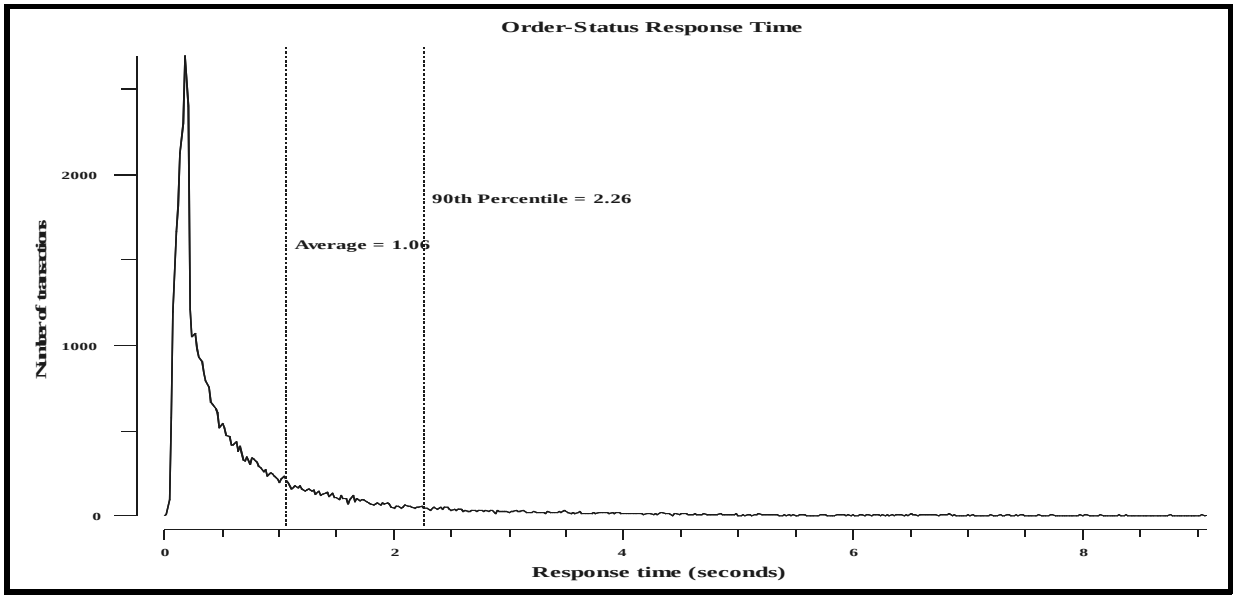


Figure 6-3-4. RS/6000 H70 Delivery (Interactive) Response Time Distribution

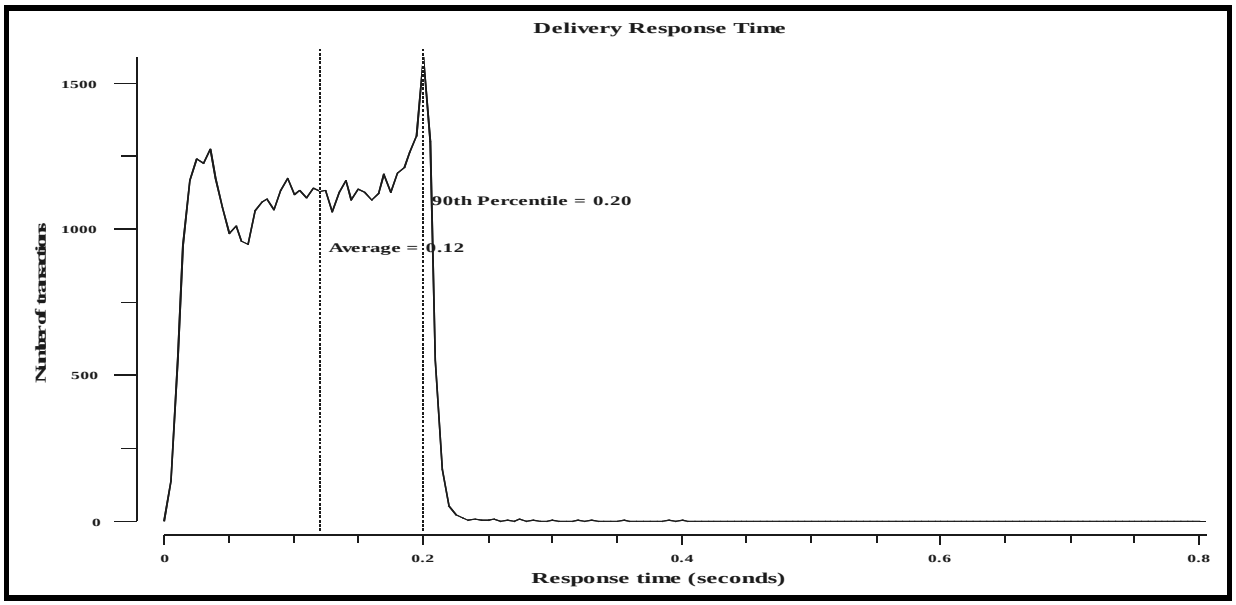


Figure 6-3-5. RS/6000 H70 Delivery (Deferred) Response Time Distribution

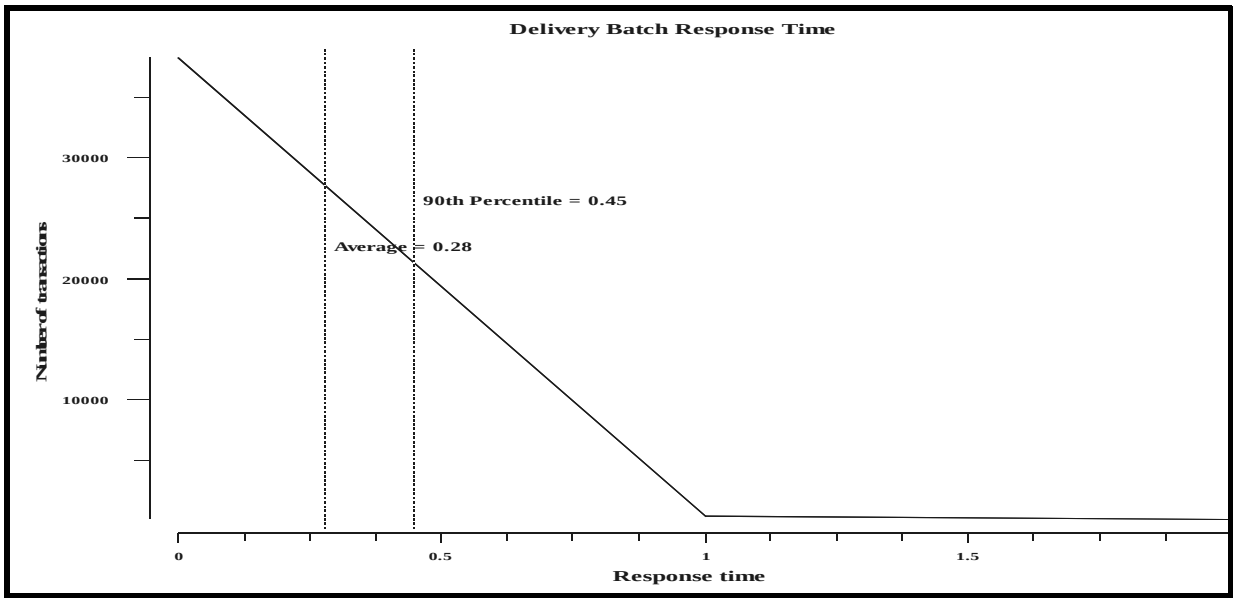
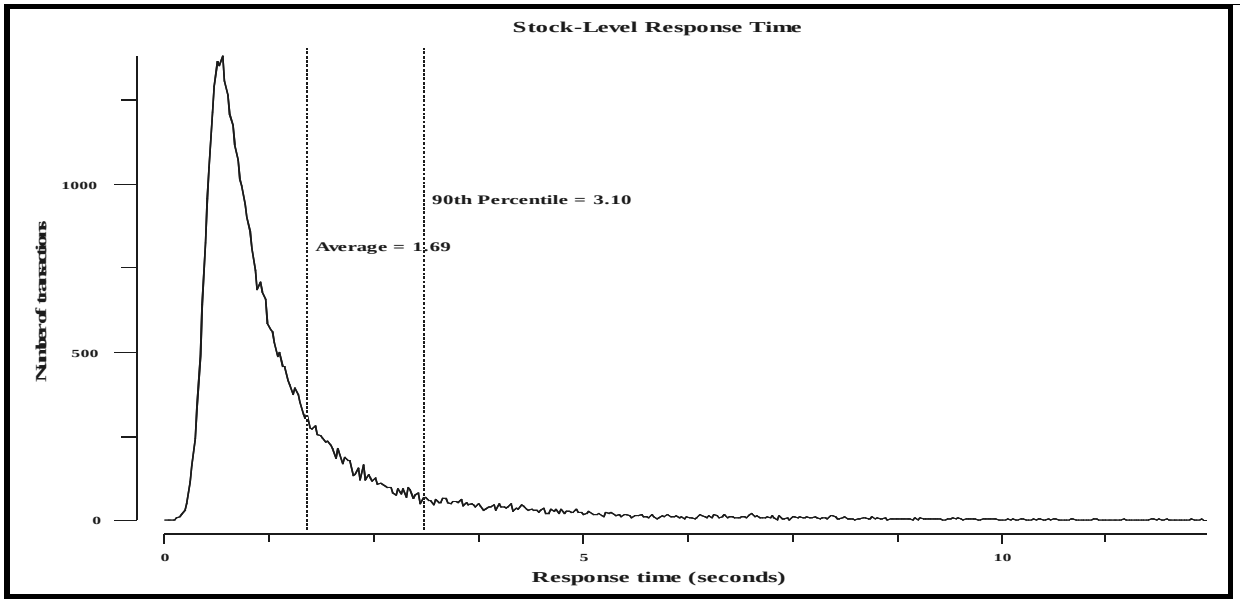


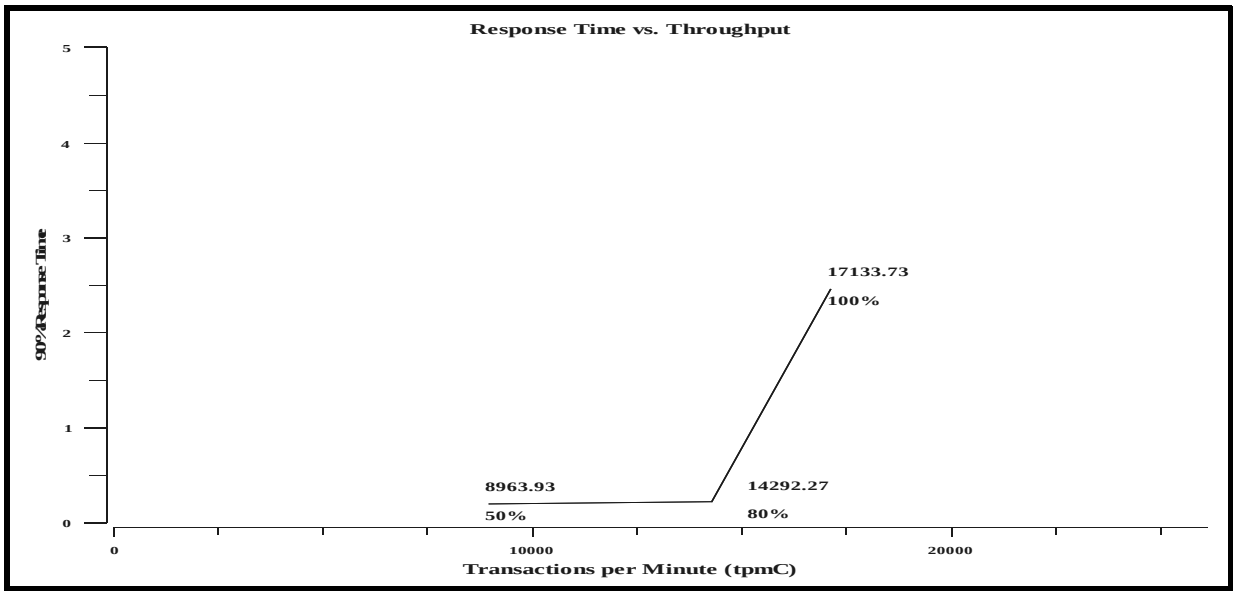
Figure 6-3-6. RS/6000 H70 Stock Level Response Time Distribution



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.

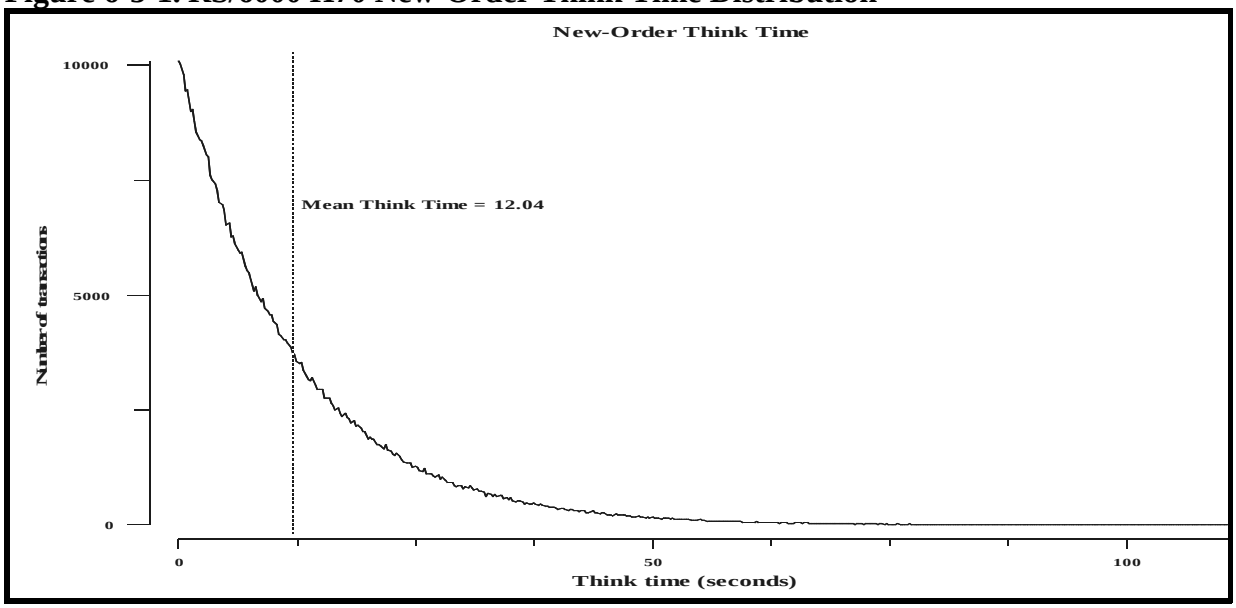
Figure 6-4-1. RS/6000 H70 New-Order Response Time vs. Throughput



6.5 Think Time Frequency Distribution

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

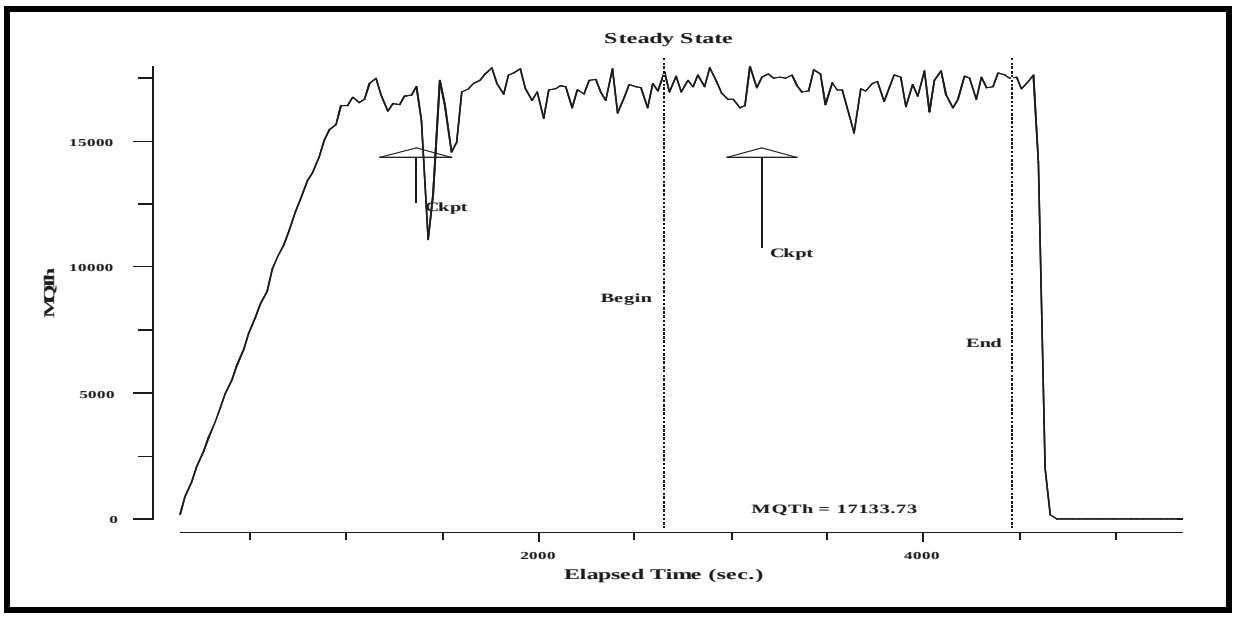
Figure 6-5-1. RS/6000 H70 New-Order Think Time Distribution



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 several 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.

- Each client machine is a node in an Encina Cell.
- Two servers are configured in each.node: one processes the delivery transactions and one all other transaction.
- The delivery server is configured with one processing agent with 2 server manager DCE threads, and 2 background threads to process deferred deliveries. Each background thread has one connection to the database.
- The other server is configured with 5 processing agents. Each processing agent has 2 server manager DCE threads. Each thread has one connection to the database.
- When the Encina clients are started, they connect to Encina cell.
- 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 terminal.
- 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 interface.
- 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 output 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 automatically checkpoint every 30 minutes. One checkpoint occurs during the rampup period, with another occurring during the measurement interval.

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 17,036.40 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 thirty minute Measurement Interval was used. Further, the measurement interval is a multiple of the checkpoint interval, and the checkpoints fall outside the protected zones of either edge of the measurement interval (as required by 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 39H emulates a network of 880 RS/6000 Model 43P-140 workstations. The communications mechanism used in the benchmark and priced configurations are the same. In the benchmark configuration a separate Ethernet LAN was used to connect two driver systems to a client system. In other words, there was a separate LAN segment every two drivers to a client. Each LAN segment in the priced configuration is used to connect 880 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 H70 configuration connected 880 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 a Midrange System.

Pricing for 8 port Ethernet Hubs was quoted by Data Comm Warehouse. These have a 5-year return-to-factory warranty.

Pricing for the 8 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 30% 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.

- IBM 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 35% for 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 except the following:

Product	Availability Date
Oracle 8.1.5	Nov. 19, 1999

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 H70	17,133.73	$1,343,526	$78.50	All HS/SW available as shown in Section 8.3

RS/6000 Enterprise Server H70 Five Year System Price Configuration

Description	Part Number	Source	Unit Price	Qty	Extended Price	5 yr. Maint Price
Server Hardware						
RS/6000 Server Model H70	7026-H70	1	17,322	1	17,322	7,872
2 SCSI Adapters, Intg Enet, Rack, PDU,						
CD-ROM, SCSI Hot Swap 6-pack						
4.5 GB 1" SCSI Disk	2900	1	1,300	1	1,300	0
9.1 GB 1" Ultra SCSI Hot Swap	2913	1	1,750	1	1,750	0
Async/Terminal Cable	2934	1	45	1	45	0
10/100 Mbs Ethernet PCI Adapter	2968	1	275	2	550	0
Memory Expansion Feature	4098	1	1,038	2	2,076	0
512 MB SDRAM DIMMs	4119	1	6,400	16	102,400	0
2way RS64 II 340 Mhz Processor	4319	1	20,000	2	40,000	38,400
Advanced SerialRAID Adapter	6225	1	3,000	4	12,000	0
32 MB Fast-Write Cache Option	6235	1	575	1	575	0
Prestige EXT UPS, Rack Mount	9910-U33	1	3,822	2	7,644	0
System Rack Model S00	7015-S00	1	3,500	3	10,500	5,040
Additional Power Distribution Unit	6171	1	1,000	3	3,000	0
SSA Disk Subsystem, Black Cover	7133-D40	1	13,000	15	195,000	144,000
50/60 Hz AC, 300 VDC Power Supply	8022	1	2,000	15	30,000	0
9.1 GB Disk Drive Modules	8209	1	2,760	226	623,760	0
SSA Cables	8801	1	60	30	1,800	0
			Subtotal		1,049,722	195,312
Server Software						
AIX 4.3 for H70	5765-C34	1	175	1	175	0
AIX C compiler	5765-C64	1	1,578	1	1,578	0
Performance Toolbox	5765-654	1	1,300	1	1,300	0
Oracle Version 8.1.5		2	153,520	1	153,520	153,520
			Subtotal		156,573	153,520
Client Hardware						
RS/6000 Model 43P-140, 332 MHz	7043-140	1	7,295	8	58,360	26,880
4.5gb Disk, Intg SCSI-2 FW, Intg Enet, 1MB L2 cache						
128MB DIMM Memory	4102	1	448	8	3,584	0
128MB DIMM Memory Expansion	4115	1	896	27	24,192	0
Async Terminal/Printer Cable	2934	1	45	8	360	0
10/100 Mbps Ethernet Adapter, PCI	2986	1	275	16	4,400	0
IBM ASCII Terminal, Keyboard	3153-BG3	1	577	9	5,193	5,940
			Subtotal		96,089	32,820
Client Software						
AIX 4.3 Unlimited Users	5756-C34	1	4,750	8	38,000	0
AIX C compiler	5765-C64	1	1,578	1	1,578	0
IBM TXSeries 4.2 for AIX	5697-D17	1	3,353	8	26,821	9,400
			Subtotal		66,399	9,400
User Connectivity						
8port Ethernet Switch+ 2 Spare	Nbase	4	2,166.50	4	8,666	1,486
Ethernet Hub (8port) + 10% Spares	DEH2924	3	32.00	1936	61,952	0
			Subtotal		70,618	1,486
			Discounts		(415,955)	(70,972)
				Total	1,023,446	321,566
Notes:						
Pricing Sources:			Five-Year Cost of Ownership:			1,345,012
1=IBM, 2=Oracle, 3=Data Comm Warehouse, 4=NBase-Xyplex					tpmC	17,133.73
Audited by: Francois Raab , Information Sizing					$/tpmC	78.50

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 at the end of the appendix section of this report.

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Sponsors:

Herve Lejeune
Oracle
System products
500 Oracle Parkway
Redwood Shores, CA 94065

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

May 20, 1998

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

Platform: RISC System/6000 Enterprise Server H70 c/s
Operating system: AIX 4.3.2
Database Manager: Oracle8 Enterprise Edition 8.1.5
Transaction Manager: IBM TX Series 4.2 for AIX

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: RISC System/6000 Enterprise Server H70 c/s				
4 x PowerPC RS64-II (340 MHz)	8.0 GB (4 MB L2 Cache per processor)	1 x 4.5 GB 227 x 9.1 GB	2.46 Seconds	17,133.73
Eight Clients: RISC System/6000 Workgroup Server 43P-140 (Specification for each)				
1 x PowerPC 604e (332 MHz)	3 with 640 MB 5 with 512 MB (1 MB L2 Cache per processor)	3 with 2 x 2.1 GB 5 with 1 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:

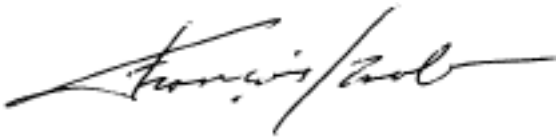
- The transactions were correctly implemented
- The database records were the proper size
- The database was properly scaled and populated
- The ACID properties were met
- Input data was generated according to the specified percentages

- 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 30 minutes.
- One checkpoint was 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:

The (6) 2.2 GB disks used in three of the clients were substituted with (3) 4.5 GB disks in the priced configuration. Based on the specifications of the disks and on measurement data collected, it is my opinion that this substitution would have no negative effect on the reported performance.

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, %f
qtime, %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++) {
     err_printf("OS: Item %d id %d amt %f\n",
                                i,
                                oraStructP->ordout.ol_i_id[i],
                                oraStructP->ordout.ol_amount[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 (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 special logins
* - 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>
#include "delivery.h"
#include "tpcc_trans.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 %.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], 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 special logins
 * - 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, "12345678901234567890");
   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, "12345678901234567890");
   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,
"12345678901234567890123456789012345678901234567890123456789012345678901234567890");
   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

```
}
/*
*               .c terminal
*
* $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 to to
 *              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 special logins
 * - 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 collecting stats 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 of timestamps
 *  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.tacf

```
/*
 * Copyright (C) 1991, 1990 Transarc Corporation
 * All Rights Reserved
 */
/*
 *  neworder.tacf -- attribute configuration file for tpcc 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 for tpccserver.
 *
 * $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
                expTPCCDelivery([in]  trpc_handle_t       handle,
                                                [in,out] delivery_data_t *dataP,
                                                [out] trpc_status_t  * trpcStatus);

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

[nontransactional] void
        expTPCCDvryInfo([in]  trpc_handle_t       handle,
                [in,out] dbInfo_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 any order\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 and PA\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 (%d 0x%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 */
```

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 collect timestamps 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 special logins
* - 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 collecting stats in the client
*   and dumping them before exit.
*
* - Removed pre_rpc_stats and post_rpc_stats
* - Removed code to write the stats 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>
#include "delivery.h"
#include "tpcc_trans.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 are active).\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 in info_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");
}
```

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%4d", POS(12,4), user_dataP->warehouse);
#else
   session_data    = static_session_data;
   sprintf(session_data, "%s%4d", 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, 4, &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_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 (last>=0) {
                                pos=last;
                                message(threadP, "Warehouse ID is a required field");
                                return 0;
                   }
                   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;
                   }
                } else if (last < 0) {
                   last = i;
                }
   }
   data->s_O_OL_CNT = (last < 0)?MAX_ITEMS:last/3;
   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;

   }

   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, 5, data->item[i].s_OL_SUPPLY_W_ID);
threadP->write( buf, 4 );
                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%4d", POS(12,6), user_dataP->warehouse);
#else
   session_data    = static_session_data;
   sprintf(session_data, "%s%4d", 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%4d", POS(12,4), user_dataP->warehouse);
#else
   session_data    = static_session_data;
   sprintf(session_data, "%s%4d", 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, 5, data->item[i].s_OL_SUPPLY_W_ID);
                threadP->write(buf, 4);

                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%4d", POS(12,4), user_dataP->warehouse);
#else
  session_data  = static_session_data;
  sprintf(session_data, "%s%4d", 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%4d%s%2d", POS(12,4), user_dataP->warehouse,
                                                        POS(29,4),
user_dataP->district);
#else
  session_data  = static_session_data;
  sprintf(session_data, "%s%4d%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,  4, &(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;
  void swap_fields(int i, int j);
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

```
#define F_INT 1
#define F_STR 2
#define F_MON 3

#define MAXVSZ 17
#define FSTR "               "  /* Must be MAXVSZ in length! */

typedef char byte;

#define REQ_MASK 1
#define DP_MASK 2

struct field {
        byte r,c;       /* row, col of 1st input position in field */
        byte nu,nd,nl,nr;   /* Indices of neigbors: up, down, left, right */
        byte sz;        /* Number of input postions in field */
        byte ty;        /* type of input value: integer, money, string */
        byte fl;        /* flags-- 1="yes"
                                                b0: required field?
                                        b1: contains decimal pt?  */
        byte ln;        /* if positive, index of linked field */
        char val[MAXVSZ+1];   /* field buffer */
        };

struct field NO_screen[47]={
        2 ,29,46,1 ,46,1 ,2 ,F_INT,1, 0,FSTR,
        3 ,12,0 ,2 ,0 ,2 ,4 ,F_INT,1, 1,FSTR,
        7 , 3,1 ,5 ,1 ,3 ,4 ,F_INT,1, 2,FSTR,
        7 ,10,1 ,6 ,2 ,4 ,6 ,F_INT,1, 3,FSTR,
        7 ,45,1 ,7 ,3 ,5 ,2 ,F_INT,1, 4,FSTR,
        8 , 3,2 ,8 ,4 ,6 ,4 ,F_INT,1, 5,FSTR,
        8 ,10,3 ,9 ,5 ,7 ,6 ,F_INT,1, 6,FSTR,
        8 ,45,4 ,10,6 ,8 ,2 ,F_INT,1, 7,FSTR,
        9 , 3,5 ,11,7 ,9 ,4 ,F_INT,1, 8,FSTR,
        9 ,10,6 ,12,8 ,10,6 ,F_INT,1, 9,FSTR,
        9 ,45,7 ,13,9 ,11,2 ,F_INT,1,10,FSTR,
        10, 3,8 ,14,10,12,4 ,F_INT,1,11,FSTR,
        10,10,9 ,15,11,13,6 ,F_INT,1,12,FSTR,
        10,45,10,16,12,14,2 ,F_INT,1,13,FSTR,
        11, 3,11,17,13,15,4 ,F_INT,1,14,FSTR,
        11,10,12,18,14,16,6 ,F_INT,1,15,FSTR,
        11,45,13,19,15,17,2 ,F_INT,1,16,FSTR,
        12, 3,14,20,16,18,4 ,F_INT,0,17,FSTR,
        12,10,15,21,17,19,6 ,F_INT,0,18,FSTR,
        12,45,16,22,18,20,2 ,F_INT,0,19,FSTR,
        13, 3,17,23,19,21,4 ,F_INT,0,20,FSTR,
        13,10,18,24,20,22,6 ,F_INT,0,21,FSTR,
        13,45,19,25,21,23,2 ,F_INT,0,22,FSTR,
```

```
                14, 3,20,26,22,24,4 ,F_INT,0,23,FSTR,
                14,10,21,27,23,25,6 ,F_INT,0,24,FSTR,
                14,45,22,28,24,26,2 ,F_INT,0,25,FSTR,
                15, 3,23,29,25,27,4 ,F_INT,0,26,FSTR,
                15,10,24,30,26,28,6 ,F_INT,0,27,FSTR,
                15,45,25,31,27,29,2 ,F_INT,0,28,FSTR,
                16, 3,26,32,28,30,4 ,F_INT,0,29,FSTR,
                16,10,27,33,29,31,6 ,F_INT,0,30,FSTR,
                16,45,28,34,30,32,2 ,F_INT,0,31,FSTR,
                17, 3,29,35,31,33,4 ,F_INT,0,32,FSTR,
                17,10,30,36,32,34,6 ,F_INT,0,33,FSTR,
                17,45,31,37,33,35,2 ,F_INT,0,34,FSTR,
                18, 3,32,38,34,36,4 ,F_INT,0,35,FSTR,
                18,10,33,39,35,37,6 ,F_INT,0,36,FSTR,
                18,45,34,40,36,38,2 ,F_INT,0,37,FSTR,
                19, 3,35,41,37,39,4 ,F_INT,0,38,FSTR,
                19,10,36,42,38,40,6 ,F_INT,0,39,FSTR,
                19,45,37,43,39,41,2 ,F_INT,0,40,FSTR,
                20, 3,38,44,40,42,4 ,F_INT,0,41,FSTR,
                20,10,39,45,41,43,6 ,F_INT,0,42,FSTR,
                20,45,40,46,42,44,2 ,F_INT,0,43,FSTR,
                21, 3,41,0 ,43,45,4 ,F_INT,0,44,FSTR,
                21,10,42,0 ,44,46,6 ,F_INT,0,45,FSTR,
                21,45,43,0 ,45, 0,2 ,F_INT,0,46,FSTR,
                };
struct field PA_screen[6]={
                4 ,52,5 ,1 ,5 ,1, 2 ,F_INT,1, 0,FSTR,
                9 ,11,0, 2 ,0 ,2 ,4 ,F_INT,1, 4,FSTR,
                9 ,33,0 ,4 ,1 ,3 ,4 ,F_INT,1, 2,FSTR,
                9 ,54,0 ,4 ,2 ,4 ,2 ,F_INT,1, 3,FSTR,
                10,29,2 ,5 ,3 ,5 ,16,F_STR,1, 1,FSTR,
                15,24,4 ,0 ,4 ,0 ,5 ,F_MON,1, 5,FSTR, /*Not counting 2 decimal places*/
                };
struct field OS_screen[3]={
                2 ,29,2 ,1 ,2 ,1 ,2 ,F_INT,1, 0,FSTR,
                3 ,11,0 ,2 ,0 ,2 ,4 ,F_INT,1, 2,FSTR,
                3 ,44,1 ,0 ,1 ,0 ,16,F_STR,1, 1,FSTR,
                };
struct field ID_screen[1]={
                4 ,17,0 ,0 ,0 ,0 ,2 ,F_INT,0, 0,FSTR,
                };
struct field SL_screen[1]={
                4 ,24,0 ,0 ,0 ,0 ,2 ,F_INT,0, 0,FSTR,
                };
struct field UI_screen[1]={
                24,10,0 ,0 ,0 ,0 ,4 ,F_INT,0, 0,FSTR,
                };
```

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 CURRENT file.\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

```
/*
 * 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"
#include "tpcc_trans.h"
#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 = 3000;
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;

   if (interface_type & ONLINE_INTERFACES) {
     ENCINA_CALL("mon_InitServerInterface",

mon_InitServerInterface(MON_SERVER_INTERFACE(tpccTrans,1,0)));
     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;
     }
   }

   if (interface_type & DELIVERY_INTERFACE) {
     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);
}
```

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 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, 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 each client.\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 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 reservered 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 */
}
```

tpccpl.c

```
#ifdef RCSID
static char *RCSid =
  "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RCS/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, %f qtime, %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 to oracle\n", num_connections);
  if (cn_array == NULL) {
              err_printf("Failed to allocated %d entried 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 new orders.\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 new orders.\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 */
```

A.2 Client Transaction Code

pldel.c

```
#ifdef RCSID
static char *RCSid =
   "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/pldel.c,v 1.1
1999/04/14 19:03:05 wenjian 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 "plora.h"
#ifdef TUX
#include <userlog.h>
#endif

#include "tpccflags.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

#ifdef PLSQLDEL
#define SQLTXT "BEGIN delivery.deliver (:w_id, :carrier_id, :order_id,\
 :retry); END;"
#else
# ifdef DMLRETDEL
#define SQLTXT1 "DELETE FROM new_order WHERE no_d_id = :d_id \
              AND  no_w_id = :w_id and rownum <= 1 \
             RETURNING  no_o_id into :o_id "
# else
#define SQLTXT1A "\
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 \
"

#define SQLTXT1B "\
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 \
"

#define SQLTXT1C "\
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 \
"

#define SQLTXT1D "\
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 \
"

#define SQLTXT1E "\
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 \
"

#define SQLTXT1F "\
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 \
"

#define SQLTXT1G "\
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 \
"

#define SQLTXT1H "\
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 \
"

#define SQLTXT1I "\
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 \
"

#define SQLTXT1J "\
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 SQLTXT2 "DELETE FROM new_order WHERE rowid = :no_rowid"
#endif

#ifdef DMLRETDEL
#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 \
             returning  o_c_id into :o_c_id"
#else
#define SQLTXT3 "UPDATE orders SET o_carrier_id = :carrier_id \
```

```
  WHERE rowid = :o_rowid"
#endif

#ifdef DMLRETDEL
#define SQLTXT4 "UPDATE /*+ buffer */ 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 \
  RETURNING ol_amount into :ol_amount "
#else
#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 SQLTXT5A "\
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 \
"

#define SQLTXT5B "\
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 \
"

#define SQLTXT5C "\
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 \
"

#define SQLTXT5D "\
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 \
"

#define SQLTXT5E "\
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"

#endif
#endif /* PLSQLDEL */

#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"

#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

#ifdef DMLRETDEL
  ub4 del_o_id_len[NDISTS];
  ub4 c_id_len[NDISTS];
  int oid_ctx;
  int cid_ctx;
  OCIBind *olamt_bp;
#else
  ub2 del_o_id_len[NDISTS];
  ub2 c_id_len[NDISTS];
#endif

  ub2 cons_len[NDISTS];
  ub2 w_id_len[NDISTS];
  ub2 d_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];
  int amt[NDISTS];
  ub4 del_o_id_rcnt;
  int retry;
  OCIRowid *no_rowid_ptr[NDISTS];
  OCIRowid *o_rowid_ptr[NDISTS];
  OCIDate del_date[NDISTS];
#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 *w_id_bp3;
  OCIBind *w_id_bp4;
  OCIBind *w_id_bp5;
  OCIBind *w_id_bp6;
  OCIBind *d_id_bp;
  OCIBind *d_id_bp3;
  OCIBind *d_id_bp4;
  OCIBind *d_id_bp6;
  OCIBind *o_id_bp;
  OCIBind *cr_date_bp;
  OCIBind *c_id_bp;
  OCIBind *c_id_bp3;
  OCIBind *no_rowid_bp;
  OCIBind *carrier_id_bp;
  OCIBind *o_rowid_bp;
  OCIBind *del_o_id_bp;
  OCIBind *del_o_id_bp3;
  OCIBind *amt_bp;
  OCIBind *bstr1_bp[10];
  OCIBind *bstr2_bp[10];
  OCIBind *retry_bp;
  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;

/* delctx *dctx; */

#ifdef DMLRETDEL
struct amtctx {
  int ol_amt[NDISTS][NITEMS];
  sb2 ol_amt_ind[NDISTS][NITEMS];
  ub4 ol_amt_len[NDISTS][NITEMS];
  ub2 ol_amt_rcode[NDISTS][NITEMS];
  int ol_cnt[NDISTS];
};
typedef struct amtctx amtctx;
/* amtctx *actx; */

#endif


#ifdef DMLRETDEL
extern sb4 no_data();

sb4 TPC_oid_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
                 dvoid **bufpp, ub4 **alenp, ub1 *piecep,
                 dvoid **indpp, ub2 **rcodepp)
{
  delctx *dctx = (delctx *)ctxp;
  *bufpp = &dctx->del_o_id[iter];
  *indpp= &dctx->del_o_id_ind[iter];
  dctx->del_o_id_len[iter]=sizeof(dctx->del_o_id[0]);
  *alenp= &dctx->del_o_id_len[iter];
```

```
 *rcodepp = &dctx->del_o_id_rcode[iter];
 *piecep =OCI_ONE_PIECE;
 return (OCI_CONTINUE);
}
sb4 cid_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
             dvoid **bufpp, ub4 **alenp, ub1 *piecep,
             dvoid **indpp, ub2 **rcodepp)
{
 delctx *dctx = (delctx *)ctxp;
 *bufpp = &dctx->c_id[iter];
 *indpp= &dctx->c_id_ind[iter];
 dctx->c_id_len[iter]=sizeof(dctx->c_id[0]);
 *alenp= &dctx->c_id_len[iter];
 *rcodepp = &dctx->c_id_rcode[iter];
 *piecep =OCI_ONE_PIECE;
 return (OCI_CONTINUE);
}


sb4 amt_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
             dvoid **bufpp, ub4 **alenp, ub1 *piecep,
             dvoid **indpp, ub2 **rcodepp)
{
 amtctx *actx;
 actx =(amtctx*)ctxp;
 actx->ol_cnt[iter]=actx->ol_cnt[iter]+1;
 *bufpp = &actx->ol_amt[iter][index];
 *indpp= &actx->ol_amt_ind[iter][index];
 actx->ol_amt_len[iter][index]=sizeof(actx->ol_amt[0][0]);
 *alenp= &actx->ol_amt_len[iter][index];
 *rcodepp = &actx->ol_amt_rcode[iter][index];
 *piecep =OCI_ONE_PIECE;
 return (OCI_CONTINUE);
}

#endif


tkvcdinit (ora_cn_data_t *ora_SlotDataP)

{

  int i,j;
  char bstr1[10];
  char bstr2[10];
  text stmbuf[SQL_BUF_SIZE];

  delctx *dctx;
  amtctx *actx;
  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;

  dctx = (delctx *) malloc (sizeof(delctx));
  memset(dctx,(char)0,sizeof(delctx));
  dctx->norow = 0;

  ora_SlotDataP->dctx = (void *)dctx;
  delP = (global_delivery_t *)malloc(sizeof(global_delivery_t));
  memset(delP, (char)0, sizeof(global_delivery_t));
  ora_SlotDataP->delP = delP;

#ifdef DMLRETDEL
  actx = (amtctx *) malloc (sizeof(amtctx));
  memset(actx,(char)0,sizeof(amtctx));
  ora_SlotDataP->actx = (void *)actx;
#else
  for(i=0;i<NDISTS;i++) {
   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));
  }
#endif

#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

/* If PLSQLDEL and ISO? are both defined, then they both try to use
  curd0! This could cause a problem. Will try to fix later - VMM 12/30/97 */

#ifdef PLSQLDEL
  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd0), OCI_HTYPE_STMT,
            0,  (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT);
  OCIStmtPrepare(dctx->curd0, errhp, stmbuf, strlen((char *)stmbuf),
             OCI_NTV_SYNTAX, OCI_DEFAULT);
  OCIBND(dctx->curd0, dctx->w_id_bp , errhp,":w_id",ADR(delP->w_id),SIZ(int),
         SQLT_INT);
  OCIBND(dctx->curd0, dctx->carrier_id_bp , errhp,":carrier_id",
         ADR(dctx->carrier_id), SIZ(int), SQLT_INT);

  OCIBNDRAA(dctx->curd0, dctx->o_id_bp, errhp, ":order_id",
          dctx->del_o_id,SIZ(int),SQLT_INT, dctx->del_o_id_ind,
          dctx->del_o_id_len,dctx->del_o_id_rcode,NDISTS,
          &dctx->del_o_id_rcnt);
  OCIBND(dctx->curd0, dctx->retry_bp , errhp,":retry",ADR(dctx->retry),
         SIZ(int),SQLT_INT);
#else
# ifdef DMLRETDEL
  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd1), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, "%s",    SQLTXT1);
  OCIStmtPrepare(dctx->curd1, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);

  OCIBND(dctx->curd1, dctx->w_id_bp,errhp,":w_id",dctx->w_id,SIZ(int),
         SQLT_INT);
  OCIBNDRA(dctx->curd1, dctx->d_id_bp,errhp,":d_id",dctx->d_id,SIZ(int),
         SQLT_INT,NULL,NULL,NULL);

  OCIBNDRAD(dctx->curd1, dctx->del_o_id_bp, errhp, ":o_id",
          SIZ(int),SQLT_INT,NULL,
          (dvoid *)dctx,no_data,TPC_oid_data);
# else

  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd1), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, "%s%s%s%s%s%s%s%s%s%s",   SQLTXT1A,

SQLTXT1B,

SQLTXT1C,

SQLTXT1D,

SQLTXT1E,

SQLTXT1F,

SQLTXT1G,

SQLTXT1H,

SQLTXT1I,

SQLTXT1J
                );
  OCIStmtPrepare(dctx->curd1, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);

  OCIERROR(errhp,
    OCIAttrSet(dctx->curd1,OCI_HTYPE_STMT,(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(int),SQLT_INT);

  OCIDFNRA(dctx->curd1, dctx->d_id_dp,errhp,1,dctx->d_id,SIZ(int),
        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(int), 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,
        SIZ(OCIRowid *), 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,
        SIZ(OCIRowid *), 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);

# endif /*DMLRETDEL*/

  /* 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);

# ifdef DMLRETDEL
  OCIBNDRA(dctx->curd3, dctx->w_id_bp3, errhp, ":w_id", dctx->w_id,SIZ(int),
        SQLT_INT, NULL, NULL, NULL);
  OCIBNDRA(dctx->curd3, dctx->d_id_bp3, errhp, ":d_id", dctx->d_id,SIZ(int),
        SQLT_INT,NULL, NULL, NULL);
  OCIBNDRA(dctx->curd3, dctx->del_o_id_bp3, errhp, ":o_id", dctx->del_o_id,
        SIZ(int), SQLT_INT,NULL,NULL,NULL);
  OCIBNDRAD(dctx->curd3, dctx->c_id_bp3, errhp, ":o_c_id", SIZ(int),
        SQLT_INT,NULL,(dvoid *)dctx,no_data, cid_data);
# else

  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);
#endif

  /* 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 */

  OCIBND(dctx->curd4, dctx->w_id_bp4,errhp,":w_id",dctx->w_id,
       SIZ(int), SQLT_INT);
  OCIBND(dctx->curd4, dctx->d_id_bp4,errhp,":d_id",dctx->d_id,
       SIZ(int), SQLT_INT);
  OCIBND(dctx->curd4, dctx->o_id_bp,errhp,":o_id",dctx->del_o_id,
        SIZ(int),SQLT_INT);
  OCIBND(dctx->curd4, dctx->cr_date_bp,errhp,":cr_date", dctx->del_date,
        SIZ(OCIDate), SQLT_ODT);
# ifdef DMLRETDEL
  OCIBNDRAD(dctx->curd4, dctx->olamt_bp, errhp, ":ol_amount",
         SIZ(int), SQLT_INT,NULL, actx,no_data,amt_data);
# else



  /* open fifth cursor */

  OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd5), OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, "%s%s%s%s%s",          SQLTXT5A,
                                                                SQLTXT5B,
                                                                SQLTXT5C,
                                                                SQLTXT5D,
                                                                SQLTXT5E
  );
  OCIStmtPrepare(dctx->curd5, errhp, stmbuf, strlen((char *)stmbuf),
                            OCI_NTV_SYNTAX, OCI_DEFAULT);

  OCIERROR(errhp,
    OCIAttrSet(dctx->curd5,OCI_HTYPE_STMT,(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);
#endif
  /* 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 */


  OCIBND(dctx->curd6,dctx->amt_bp,errhp,":amt",dctx->amt,SIZ(int),
       SQLT_INT);
  OCIBND(dctx->curd6,dctx->w_id_bp6,errhp,":w_id",dctx->w_id,SIZ(int),
        SQLT_INT);
  OCIBND(dctx->curd6,dctx->d_id_bp6,errhp,":d_id",dctx->d_id,SIZ(int),
        SQLT_INT);
  OCIBND(dctx->curd6,dctx->c_id_bp,errhp,":c_id",dctx->c_id,SIZ(int),
        SQLT_INT);
#endif
  return (0);

}


void shiftdata(delctx *dctx,int from)
{
  int i;
  for (i=from;i<NDISTS-1; i++)
  {
    dctx->del_o_id_ind[i] = dctx->del_o_id_ind[i+1];
    dctx->del_o_id[i] = dctx->del_o_id[i+1];
    dctx->w_id[i] = dctx->w_id[i+1];
    dctx->d_id[i] = dctx->d_id[i+1];
    dctx->carrier_id[i] = dctx->carrier_id[i+1];
  }
}


tkvcd (ora_cn_data_t *ora_SlotDataP)

{

  int i, j,v;
  int rpc,rcount,count;
  int invalid;
  int tmp_id;
  int tmp_amt;

  delctx *dctx = (delctx *)ora_SlotDataP->dctx;
#ifdef DMLRETDEL /* VMM 1/13/98 */
  amtctx *actx = (amtctx *)ora_SlotDataP->actx;
#endif /* DMLRETDEL */
  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;

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  int hasno;
  int reread;
  char sdate[30];

  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
#ifdef PLSQLDEL
  for (i = 0; i < NDISTS; i++)
  {
    dctx->del_o_id_ind[i] = TRUE;
    dctx->del_o_id_len[i] = sizeof(int);
  }

  OCIERROR(errhp,
    OCIStmtExecute(tpcsvc,dctx->curd0,errhp,1,0,0,0,OCI_DEFAULT));

  for (i = 0; i < NDISTS; i++)
  {
    delP->del_o_id[i] = 0;
    if (dctx->del_o_id_ind[i] == 0)
    {
      delP->del_o_id[i] = dctx->del_o_id[i];
    }
  }
#else

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->d_id[i] = i+1;
    dctx->carrier_id[i] = delP->o_carrier_id;
    memcpy(&dctx->del_date[i],&delP->cr_date,sizeof(OCIDate));
  }

#ifdef DMLRETDEL /* VMM 1/13/98 */
  memset(actx,(char)0,sizeof(amtctx));
#endif /* DMLRETDEL */
  /* 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_ROW_COUNT,errhp);
  rpc = rcount;
#ifdef DMLRETDEL   /* we have to compress the array here */
  if (rcount != NDISTS )
  {
    int j = 0;
    for (i=0;i < NDISTS; i++)
    {
     if (dctx->del_o_id_ind[j] == 0) /* there is data here */
       j++;
     else
       shiftdata(dctx, j);
    }
  }
#else
  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;
  }
#endif

#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

#ifndef DMLRETDEL
  /* 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_ROW_COUNT,errhp);

  if (rcount != rpc) {
#ifdef TUX
    userlog ("Error in TPC-C server %d: %d rows selected, %d rows deleted\n",
          proc_no, rpc, dctx->curd2.rpc);
#else
    fprintf (stderr,
          "Error in TPC-C server %d: %d rows selected, %d rows deleted\n",
          proc_no, rpc, rcount);
#endif /* TUX */
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    return (DEL_ERROR);
  }
#endif /* DMLRETDEL */

  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_ROW_COUNT,errhp);

  if (rcount != rpc) {
#ifdef TUX
    userlog ("Error in TPC-C server %d: %d rows selected, %d ords updated\n",
          proc_no, rpc, rcount);
#else
    fprintf (stderr,
          "Error in TPC-C server %d: %d rows selected, %d ords updated\n",
          proc_no, rpc, rcount);
#endif
    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;
    }
  }
#ifdef DMLRETDEL

OCIAttrGet(dctx->curd4,OCI_HTYPE_STMT,&rcount,NULL,OCI_ATTR_ROW_COUNT,errhp);
/* add up amounts */
  count=0;
  for (i=0;i<rpc;i++)
  {
    dctx->amt[i]=0;
    for (j=0;j<actx->ol_cnt[i];j++)
      if ( actx->ol_amt_rcode[i][j] == 0)
      {
        dctx->amt[i] = dctx->amt[i] + actx->ol_amt[i][j];
        count = count+1;
      }
  }
  if (rcount > rpc*NITEMS) {
    userlog ("Error in TPC-C server %d: %d ordnrs updated, %d ordl updated\n",
          proc_no, rpc, rcount);
  }
#else
  /* 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_ROW_COUNT,errhp);
  if (rcount != rpc) {
#ifdef TUX
     userlog ("Error in TPC-C server %d: %d rows selected, %d ordl selected\n",
           proc_no, rpc, rcount);
#else
     fprintf (stderr,
           "Error in TPC-C server %d: %d rows selected, %d ordl selected\n",
           proc_no, rpc, rcount);
#endif
     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]) {
#ifdef TUX
        userlog ("TPC-C server %d: reordering amount\n", proc_no);
#else
        fprintf (stderr, "TPC-C server %d: reordering amount\n", proc_no);
#endif
        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) {
#ifdef TUX
           userlog ("Error in TPC-C server %d: missing ordl?\n", proc_no);
#else
           fprintf (stderr,
                 "Error in TPC-C server %d: missing ordl?\n", proc_no);
#endif
                    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
           return (-1);
        }
     }
  }
#endif
#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)
  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_ROW_COUNT,errhp);

  if (rcount != rpc) {
#ifdef TUX
     userlog ("Error in TPC-C server %d: %d rows selected, %d cust updated\n",
           proc_no, rpc, rcount);
#else
     fprintf (stderr,
           "Error in TPC-C server %d: %d rows selected, %d cust updated\n",
           proc_no, rpc, rcount);
#endif
     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];
#endif

  return (0);

}



void tkvcddone (ora_cn_data_t *ora_SlotDataP)

{
  delctx *dctx = (delctx *)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
#ifdef PLSQLDEL
    OCIHandleFree((dvoid *)dctx->curd0,OCI_HTYPE_STMT);
#else
    /* Again the above will cause a problem if both PSQLDEL and ISO are
                defined - VMM 12/30/97 */
    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);
#endif
    free (dctx);
    ora_SlotDataP->dctx = NULL;
  }

  if (delP) {
    free(delP);
    ora_SlotDataP->delP = NULL;
  }
}
```

plnew.c

```
#ifdef RCSID
static char *RCSid =
  "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/plnew.c,v 1.1
1999/04/14 19:03:05 wenjian Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*================================================================+
|      Copyright (c) 1996 , 1997, 1998 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 "plora.h"
```

```
#ifdef TUX
#include <userlog.h>
#endif
#include "tpccflags.h"

extern void err_printf(char *format, ...);

#define PLSQLNO

#ifdef PLSQLNO
#define SQLTXT2 "BEGIN initnew.new_init(:idx1arr); END;"
#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"

#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)"
#endif  /* PLSQLNO */

#define NITEMS 15
#define ROWIDLEN 20
#define OCIROWLEN 20

sb4 no_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
            dvoid **bufpp, ub4 *alenp, ub1 *piecep,
            dvoid **indpp)
{
 *bufpp = (dvoid*)0;
 *alenp =0;
 *indpp = (dvoid*)0;
 *piecep =OCI_ONE_PIECE;
 return (OCI_CONTINUE);
}
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];
#ifdef PLSQLNO
 sb2 s_bg_ind[NITEMS];
#endif

 ub2 nol_i_id_len[NITEMS];
 ub2 nol_supply_w_id_len[NITEMS];
 ub2 nol_quantity_len[NITEMS];
 ub2 nol_amount_len[NITEMS];
 ub2 s_quantity_len[NITEMS];
 ub2 i_name_len[NITEMS];
 ub2 i_price_len[NITEMS];
 ub2 i_data_len[NITEMS];
 ub2 s_dist_info_len[NITEMS];
 ub2 s_data_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 ol_dist_info_len[NITEMS];
 ub2 null_date_len[NITEMS];
#ifdef PLSQLNO
 ub2 s_bg_len[NITEMS];
#endif

 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_rcode[NITEMS];
#ifdef PLSQLNO
 ub2 s_bg_rcode[NITEMS];
#endif

 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];
 OCIDate null_date[NITEMS]; /* base date for null date entry */
 OCIStmt *curn1;
#ifdef PLSQLNO
 OCIBind *ol_i_id_bp;
 OCIBind *ol_supply_w_id_bp;
 OCIBind *i_price_bp;
 OCIBind *i_name_bp;
 OCIBind *s_bg_bp;
 OCIBind *s_data_bp;
 OCIBind *i_data_bp;
 ub4 nol_i_count;
 ub4 nol_s_count;
 ub4 nol_q_count;
 ub4 nol_item_count;
 ub4 nol_name_count;
 ub4 nol_qty_count;
 ub4 nol_bg_count;
 ub4 nol_am_count;
 ub4 s_remote_count;
 ub4 s_data_count;
 ub4 i_data_count;
#endif
 OCIStmt *curn2;
 OCIStmt *curn3[10];
 OCIBind *ol_i_id_bp4;
 OCIBind *ol_supply_w_id_bp4;
 OCIBind *ol_quantity_bp;
 OCIBind *ol_quantity_bp4;
 OCIBind *s_remote_bp;
 OCIBind *s_quantity_bp;
 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 *s_rowid_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;

/* context holders */
  int i_name_ctx;
  int i_data_ctx;
  int i_price_ctx;
  int s_data_ctx;
  int s_dist_info_ctx;
  int s_quantity_ctx;
};

typedef struct newctx newctx;

/* newctx *nctx; */

tkvcninit (ora_cn_data_t *ora_SlotDataP)
{

  int i, j;
  text stmbuf[16*1024];
  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;

  nctx = (newctx *) malloc (sizeof(newctx));
  memset(nctx,(char)0,sizeof(newctx));
  ora_SlotDataP->nctx = (void *)nctx;

  ora_SlotDataP->globals = (global_newOrder_t *)malloc(sizeof(global_newOrder_t));
  memset(ora_SlotDataP->globals,(char)0,sizeof(global_newOrder_t));
  newP = ora_SlotDataP->globals;

  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_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));
#ifdef PLSQLNO
/*  sqlfile("../blocks/tkvcpnew.sql",stmbuf); */
  sqlfile("tkvcpnew.sql",stmbuf);
#else
/*  sqlfile("../blocks/tkvcbnew.sql",stmbuf);  */
  sqlfile("tkvcbnew.sql",stmbuf);
#endif
  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_FLT, &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_FLT, &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_FLT,
      &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(OCIDate),
  SQLT_ODT, &nctx->cr_date_ind, &nctx->cr_date_len, &nctx->cr_date_rc);

  #ifdef PLSQLNO
  OCIBNDRAA(nctx->curn1, 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,NITEMS,&nctx->nol_i_count);
OCIBNDRAA(nctx->curn1, 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,
        NITEMS, &nctx->nol_s_count);
OCIBNDRAA(nctx->curn1, 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,NITEMS,&nctx->nol_q_count);
OCIBNDRAA(nctx->curn1, nctx->i_price_bp,errhp,":i_price",newP->i_price,SIZ(float),
         SQLT_FLT, nctx->i_price_ind,nctx->i_price_len,nctx->i_price_rcode,
         NITEMS, &nctx->nol_item_count);
OCIBNDRAA(nctx->curn1, nctx->i_name_bp,errhp,":i_name",newP->i_name,
         SIZ(newP->i_name[0]),SQLT_STR, nctx->i_name_ind,nctx->i_name_len,
         nctx->i_name_rcode,NITEMS,&nctx->nol_name_count);
OCIBNDRAA(nctx->curn1, 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,NITEMS,&nctx->nol_qty_count);
OCIBNDRAA(nctx->curn1, nctx->s_bg_bp,errhp,":brand_generic",newP->brand_generic,
        SIZ(char), SQLT_CHR,nctx->s_bg_ind,nctx->s_bg_len,
        nctx->s_bg_rcode,NITEMS,&nctx->nol_bg_count);
OCIBNDRAA(nctx->curn1, 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,NITEMS,&nctx->nol_am_count);
OCIBNDRAA(nctx->curn1, 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,NITEMS,&nctx->s_remote_count);

/* 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));


/* execute second cursor to init newinit package */
{
  int idx1arr[NITEMS];
  OCIBind *idx1arr_bp;
  ub2 idx1arr_len[NITEMS];
  ub2 idx1arr_rcode[NITEMS];
  sb2 idx1arr_ind[NITEMS];
  ub4 idx1arr_count;
  ub2 idx;

  for (idx = 0; idx < NITEMS; idx++) {
    idx1arr[idx] = idx + 1;
    idx1arr_ind[idx] = TRUE;
    idx1arr_len[idx] = sizeof(int);
  }
  idx1arr_count = NITEMS;
                newP->o_ol_cnt = NITEMS;


  /* Bind array */
  OCIBNDRAA(nctx->curn2, idx1arr_bp,errhp,":idx1arr",idx1arr,
         SIZ(int), SQLT_INT, idx1arr_ind,idx1arr_len,
         idx1arr_rcode,NITEMS,&idx1arr_count);

   newP->execstatus = OCIStmtExecute(tpcsvc,nctx->curn2,errhp,1,0,0,0,OCI_DEFAULT);
   if(newP->execstatus != OCI_SUCCESS) {
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     newP->errcode = OCIERROR(errhp,newP->execstatus);
     return -1;
   }
  }
#else

/* 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 */

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);
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, 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_STMT,(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(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);
  }

  OCIDEF((nctx->curn3)[i],(nctx->Dcons)[i],errhp,1,&(nctx->cons[0]),
                SIZ(nctx->cons[0]),SQLT_INT);
  OCIDEF((nctx->curn3)[i], (nctx->Ds_rowid)[i],errhp,2,
        nctx->s_rowid_ptr, sizeof(nctx->s_rowid_ptr[0]), SQLT_RDD);
  OCIDEF((nctx->curn3)[i], (nctx->Di_price)[i],errhp,3,newP->i_price,SIZ(int),
        SQLT_INT);

  OCIDFNRA((nctx->curn3)[i], (nctx->Di_name)[i],errhp,4,newP->i_name,
        SIZ(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, NULL,nctx->i_data_len,NULL);
  OCIDFNRA((nctx->curn3)[i], (nctx->Ds_dist_info)[i],errhp,6,
        nctx->s_dist_info, SIZ(nctx->s_dist_info[0]),SQLT_STR,
        NULL,nctx->s_dist_info_len, NULL);
  OCIDFNRA((nctx->curn3)[i],(nctx->Ds_data)[i],errhp,7,nctx->s_data,
                SIZ(nctx->s_data[0]),SQLT_STR,NULL,nctx->s_data_len,NULL);
  OCIDEF((nctx->curn3)[i],(nctx->Ds_quantity)[i],errhp,8,newP->s_quantity,
                SIZ(int),SQLT_INT);

  }

  /* 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, NULL,nctx->ol_o_id_len,
        NULL);
  OCIBNDRA(nctx->curn4, nctx->ol_d_id_bp,errhp,":ol_d_id",nctx->ol_d_id,
        SIZ(int),SQLT_INT, NULL,nctx->ol_d_id_len,
        NULL);
  OCIBNDRA(nctx->curn4, nctx->ol_w_id_bp,errhp,":ol_w_id",nctx->ol_w_id,
        SIZ(int),SQLT_INT, NULL,nctx->ol_w_id_len,
        NULL);
  OCIBNDRA(nctx->curn4, nctx->ol_number_bp,errhp,":ol_number",nctx->ol_number,
        SIZ(int),SQLT_INT, NULL,nctx->ol_number_len,
        NULL);
  OCIBNDRA(nctx->curn4, nctx->ol_i_id_bp4,errhp,":ol_i_id",newP->nol_i_id,SIZ(int),
        SQLT_INT, NULL,nctx->nol_i_id_len, NULL);
  OCIBNDRA(nctx->curn4, nctx->ol_supply_w_id_bp4,errhp,":ol_supply_w_id",
        newP->nol_supply_w_id,SIZ(int),SQLT_INT, NULL,
        nctx->nol_supply_w_id_len, NULL);
  OCIBNDRA(nctx->curn4, nctx->ol_quantity_bp4,errhp,":ol_quantity",newP->nol_quantity,
        SIZ(int),SQLT_INT, NULL,nctx->nol_quantity_len,
                NULL);
  OCIBNDRA(nctx->curn4, nctx->ol_amount_bp,errhp,":ol_amount",newP->nol_amount,
        SIZ(int),SQLT_INT, NULL,nctx->nol_amount_len,
        NULL);
  OCIBNDRA(nctx->curn4, nctx->ol_dist_info_bp,errhp,":ol_dist_info",
        nctx->s_dist_info, SIZ(nctx->s_dist_info[0]),SQLT_AFC,
        NULL, nctx->ol_dist_info_len,
        NULL);
  OCIBNDRA(nctx->curn4, nctx->null_date_bp,errhp,":null_date",nctx->null_date,
        SIZ(OCIDate), SQLT_ODT,NULL,
        nctx->null_date_len, NULL);

  /* set up the null date Null date is 15-sep-11 */
  for (i=0;i<NITEMS;i++)
  {
    OCIDateSetDate(&nctx->null_date[i],(sb2)1811,(ub1)9,(ub1)15);
  }
 #endif

  return (0);

}
```

```
tkvcn (ora_cn_data_t *ora_SlotDataP)
{

  int i, j, k;
  int rpc, rpc3, rowoff, iters,rcount;
  ub4 flags;
  int failed = 0;

  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 = (newctx *)ora_SlotDataP->nctx;

retry:

  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);
#ifdef PLSQLNO
  /* this is the row count */
  rcount = newP->o_ol_cnt;
  nctx->nol_i_count = newP->o_ol_cnt;
  nctx->nol_q_count = newP->o_ol_cnt;
  nctx->nol_s_count = newP->o_ol_cnt;
  nctx->s_remote_count = newP->o_ol_cnt;

  nctx->nol_qty_count  = 0;
  nctx->nol_bg_count = 0;
  nctx->nol_item_count = 0;
  nctx->nol_name_count = 0;
  nctx->nol_am_count   = 0;
  /* following not relevant */
  nctx->s_data_count = newP->o_ol_cnt;
  nctx->i_data_count = newP->o_ol_cnt;

  /* 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] = 0;
    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_data_ind[i] = TRUE;
    nctx->i_data_ind[i] = TRUE;
    nctx->s_quant_ind[i] = TRUE;
    nctx->s_bg_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] = nctx->s_dist_info_len[i];
    nctx->null_date_len[i]=sizeof(OCIDate);
    nctx->s_remote_len[i] = sizeof(int);
    nctx->s_data_len[i] = sizeof(int);
    nctx->i_data_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);
    nctx->i_name_len[i]=0;
    nctx->s_bg_len[i] = 0;
  }
  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_data_ind[i] = NA;
    nctx->i_data_ind[i] = NA;
    nctx->s_quant_ind[i] = NA;
    nctx->s_bg_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->i_data_len[i] = 0;
    nctx->s_data_len[i] = 0;
    nctx->s_quant_len[i] = 0;
    nctx->s_rowid_len[i] = 0;
    nctx->cons_len[i] = 0;
    nctx->i_name_len[i]=0;
    nctx->s_bg_len[i] = 0;
  }

  newP->execstatus = OCIStmtExecute(tpcsvc,nctx->curn1,errhp,1,0,0,0,
                                    OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);

#else
  newP->execstatus = OCIStmtExecute(tpcsvc,nctx->curn1,errhp,1,0,0,0,OCI_DEFAULT);
#endif


  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;
    }
  }

#ifdef PLSQLNO
  /* did the txn succeed ? */
  if (rcount != newP->o_ol_cnt)
  {
    newP->status = rcount - newP->o_ol_cnt;
    newP->o_ol_cnt = rcount;
  }
#endif

#ifdef DEBUG
  err_printf("tkvcn (NO): w_id = %d, d_id = %d, c_id = %d\n",w_id, d_id, c_id);
```

```
#endif


#ifndef PLSQLNO
  /* initialization for array operations */

  for (i = 0; i < o_ol_cnt; i++) {
    nctx->ol_w_id[i] = w_id;
    nctx->ol_d_id[i] = 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] = nctx->s_dist_info_len[i];
    nctx->null_date_len[i]=sizeof(OCIDate);
    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 = 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;
  }

  rpc3 = SelItemStk (nctx, newP, tpcsvc, errhp);
  if (rpc3 == -2)
    goto retry;
  else if (rpc3 == -1)
    return (-1);

  /* compute order line amounts, total amount and stock quantities */

  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) {
      newP->s_quantity[i] -= newP->nol_quantity[i];
      if (newP->s_quantity[i] < 10)
        newP->s_quantity[i] += 91;
      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';
    }

   }
    total_amount *= ((float)(10000 - c_discount)/10000) * (1.0 + ((float)(d_tax)/10000) +
((float)(w_tax)/10000));
    newP->total_amount = newP->total_amount/100;

    rpc = UpdStk2 (nctx, newP, tpcsvc, errhp);
    if (rpc == -2)
      goto retry;
    else if (rpc == -1)
      return (-1);

    /* error processing - will keep it separated for readablity */
    /* number of items selected != number of stock updated */

    if (rpc3 != rpc) {
      userlog ("Error in TPC-C server %d: %d rows of item read, ",
            newP->proc_no, rpc3);
      userlog ("                    but %d rows of stock  updated\n", rpc);
      /* rollback */
      OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      return (-1);
    }


/* common code for insert into order_line */
    for (i=0; i< newP->o_ol_cnt; i++)  /* move district info in place */
    {
      nctx->ol_dist_info_len[i]=nctx->s_dist_info_len[i];
    }

    /* array insert into order line table */
    flags= (newP->status ? OCI_DEFAULT : (OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS));
    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,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_ROW_COUNT, errhp);
      if (rcount != (newP->o_ol_cnt - newP->status))
      {
       userlog ("Error in TPC-C server %d: array insert failed\n",
                                  newP->proc_no);
       /* rollback */
       OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
       return (-1);
      }
    }

    /* commit if no invalid item */

    if (newP->status) {
      OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      fflush(stdout);
    }

#else
    newP->total_amount = 0.0;
    for (i = 0; i < newP->o_ol_cnt; i++)
    {
      if (nctx->nol_i_id_ind[i] != NA) {
        newP->total_amount += newP->nol_amount[i];
      }
    }
    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;
#endif
    return (0);

}



void tkvcndone (ora_cn_data_t *ora_SlotDataP)

{

    int i;
    newctx *nctx = (newctx *)ora_SlotDataP->nctx;
    global_newOrder_t *newP = ora_SlotDataP->globals;

    if (nctx)
    {
      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);
    free (nctx);
  }
  if (newP) {
    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


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_ROW_COUNT, 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)
  {
    userlog ("TPC-C server %d: more than 1 invalid item?\n", 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 */
  }


        userlog ("TPC-C server %d: reordering items and stocks\n",
                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);

}



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_ROW_COUNT,
errhp);
   rpc = rcount;

   if (rpc != (newP->o_ol_cnt - newP->status)) {
     userlog ("Error in TPC-C server %d: array update failed\n",
            newP->proc_no);
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     return (-1);
   }

   return (rpc);

}
```

plora.h

```
#ifndef TPCC_PLORA_H
#define TPCC_PLORA_H

#include "tpcc.h"
#include "tpcc_info.h"

#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)

/* 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;
  ub4  datelen;
  text o_entry_d[20];
  int o_id;
  int o_ol_cnt;
  char c_last[17];
  char c_credit[3];
  float c_discount;
  float w_tax;
  float d_tax;
  float total_amount;
  char i_name[15][25];
  int s_quantity[15];
  char brand_gen[15];
  float i_price[15];
  int nol_amount[15];
  int status;
  int o_all_local;
  int errcode;
  int execstatus;
  int proc_no;
  char brand_generic[15][1];
  int tracelevel;
  OCIDate cr_date;
  OCIDate c_since;
  OCIDate o_entry_d_base;
  OCIDate ol_d_base[15];
};

typedef struct global_newOrder_t global_newOrder_t;

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;
  OCIDate c_since;
  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];
  ub4  sincelen;
  text c_since_d[11];
  float c_discount;
  char c_credit[3];
  int c_credit_lim;
  char c_data[201];
  ub4 hlen;
  text h_date[20];
```

```
  OCIDate cr_date;
};

typedef struct global_payment_t global_payment_t;

struct global_order_t {
  OCIDate ol_d_base[15];
  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];
  ub4  ol_del_len[15];
  text 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;

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;
  OCIDate cr_date;
};
typedef struct global_delivery_t global_delivery_t;

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;


/* 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;
  void *nctx;
  void *pctx;
  void *octx;
  void *sctx;
  void *dctx;
  void *actx;  /* for #ifdef DMLRETDEL */
  void *cbctx; /* for orderstatus */
  void *ctxp_octx; /* for orderstatus */
};

typedef struct ora_cn_data_t 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/server/ora8.1_mt/RCS/plord.c,v 1.1
1999/04/14 19:03:05 wenjian 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 "plora.h"  /*  */
#include "tpccflags.h"

#ifdef PLSQLORD
#define SQLTXT "BEGIN orderstatus.getstatus (: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;"
#else

#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_last, c_d_id, c_w_id, c_first"

#define SQLCUR1 "SELECT c_id, c_balance, c_first, c_middle, c_last, \
              o_id, o_entry_d, o_carrier_id, o_ol_cnt \
             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_c_id, o_d_id, o_w_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_c_id, o_d_id, o_w_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"

#define SQLCUR4 "SELECT count(c_last) FROM customer \
             WHERE  c_d_id = :d_id AND c_w_id = :w_id AND c_last = :c_last "
#endif



struct ordctx {
  sb2 c_rowid_ind[100];
  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;

  ub4 c_rowid_len[100];
  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[100];
  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;
  OCIBind *w_id_bp0;
  OCIBind *d_id_bp0;
  OCIBind *c_id_bp;
  OCIBind *c_last_bp;
#ifdef PLSQLORD
  OCIBind *byln_bp;
  OCIBind *c_first_bp;
  OCIBind *c_middle_bp;
  OCIBind *c_balance_bp;
  OCIBind *o_entry_d_bp;
  OCIBind *o_cr_id_bp;
  OCIBind *o_ol_cnt_bp;
  OCIBind *ol_i_id_bp;
  OCIBind *ol_supply_w_id_bp;
  OCIBind *ol_quantity_bp;
  OCIBind *ol_amount_bp;
  OCIBind *ol_d_base_bp;
  ub4 ol_i_id_cnt;
  ub4 ol_sup_cnt;
  ub4 ol_qty_cnt;
  ub4 ol_amt_cnt;
  ub4 ol_del_d_cnt;
#else
  OCIStmt *curo1;
  OCIStmt *curo2;
  OCIStmt *curo3;
  OCIStmt *curo4;
  OCIBind *w_id_bp2;
  OCIBind *w_id_bp3;
  OCIBind *w_id_bp4;
  OCIBind *d_id_bp2;
  OCIBind *d_id_bp3;
  OCIBind *d_id_bp4;
  OCIBind *c_last_bp4;
  OCIBind *o_id_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;
  OCIDefine *c_count_dp;
  OCIRowid *c_rowid_ptr[100];
  int cs;
  int cust_idx;
  int norow;
  int rcount;
  int somerows;
#endif
};

typedef struct ordctx ordctx;

struct defctx
{
 boolean reexec;
 ub4 count;
};
typedef struct defctx defctx;

struct defctx_ordctx {
    defctx *ctxp;
    ordctx *octx;
};
typedef struct defctx_ordctx defctx_ordctx;

/* ordctx *octx; */

/* defctx cbctx; */

#ifndef PLSQLORD
sb4 rid_data(dvoid *ctxp_octx, OCIDefine *dp, ub4 iter,
             dvoid **bufpp, ub4 **alenp, ub1 *piecep,
             dvoid **indpp, ub2 **rcodepp)
{
  ub4 i;
  defctx *ctxp = ((defctx_ordctx *)ctxp_octx)->ctxp;
  ordctx *octx = ((defctx_ordctx *)ctxp_octx)->octx;
  if (((defctx*)ctxp)->reexec)/* if this is the second execute - use entry 0 */
  {
    i = 0;
    ((defctx*)ctxp)->count--;   /* count down */
  }
  else
    i = iter;
  *bufpp = octx->c_rowid_ptr[i];
  *indpp= &octx->c_rowid_ind[i];
  *alenp= &octx->c_rowid_len[i];
  *rcodepp = &octx->c_rowid_rcode[i];
  *piecep =OCI_ONE_PIECE;
  return (OCI_CONTINUE);
}
#endif

tkvcoinit (ora_cn_data_t *ora_SlotDataP)

{

  int i;
  text stmbuf[SQL_BUF_SIZE];

  ordctx *octx;
  defctx *cbctx;
  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;
  defctx_ordctx *ctxp_octx;

  octx = (ordctx *) malloc (sizeof(ordctx));
  memset(octx,(char)0,sizeof(ordctx));
  ora_SlotDataP->octx = (void *)octx;  /*  */

  cbctx = (defctx *) malloc (sizeof(defctx));
  memset(cbctx,(char)0,sizeof(defctx));
  ora_SlotDataP->cbctx = (void *)cbctx;  /*  */

  /* allocate the space */
  ctxp_octx = (defctx_ordctx *)malloc(sizeof(defctx_ordctx));
  ora_SlotDataP->ctxp_octx = (void *)ctxp_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;

#ifndef PLSQLORD
  octx->cs = 1;
  octx->norow = 0;
  octx->somerows = 10;
/* get the rowid handles */
  for(i=0;i<100;i++) {
   OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&octx->c_rowid_ptr[i],
     OCI_DTYPE_ROWID,0,(dvoid**)0));
  }
#endif


  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
#ifndef PLSQLORD
  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));
  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo4,OCI_HTYPE_STMT,0,(dvoid**)0));
#endif

#ifdef PLSQLORD
  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_STMT,(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_STMT,(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_STMT,(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_STMT,(dvoid*)&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));

  sprintf((char *) stmbuf, SQLCUR4);
  OCIERROR(errhp,
    OCIStmtPrepare(octx->curo4,errhp,stmbuf,strlen((char*)stmbuf),
                                OCI_NTV_SYNTAX,OCI_DEFAULT));
  OCIERROR(errhp,
    OCIAttrSet(octx->curo4,OCI_HTYPE_STMT,(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 PLSQLORD

  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_id_bp , errhp,":c_id",ADR(ordP->c_id),
       SIZ(c_id),SQLT_INT);
  OCIBND(octx->curo0, octx->byln_bp , errhp,":byln",ADR(ordP->bylastname),
       SIZ(int),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 (float),SQLT_FLT);
  OCIBND(octx->curo0, octx->c_id_bp , errhp,":o_id",ADR(ordP->o_id),
       SIZ(int),SQLT_INT);
  OCIBND(octx->curo0, octx->o_entry_d_bp , errhp,":o_entry_d",ordP->o_entry_d,
       SIZ(ordP->o_entry_d),SQLT_STR);
  OCIBND(octx->curo0, octx->o_cr_id_bp , errhp,":o_cr_id",ADR(ordP->o_carrier_id),
       SIZ(int), SQLT_INT);
  OCIBND(octx->curo0, octx->o_ol_cnt_bp , errhp,":o_ol_cnt",ADR(ordP->o_ol_cnt),
       SIZ(int),SQLT_INT);

  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,&octx->ol_i_id_cnt);
  OCIBNDRAA(octx->curo0,octx->ol_supply_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,&octx->ol_sup_cnt);
  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,&octx->ol_qty_cnt);
  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,
        &octx->ol_amt_cnt);
  OCIBNDRAA(octx->curo0,octx->ol_d_base_bp,errhp,":ol_d_d",ordP->ol_d_base,
        SIZ(OCIDate),SQLT_ODT,octx->ol_delivery_d_ind,
        octx->ol_delivery_d_len, octx->ol_delivery_d_rcode,NITEMS,
        &octx->ol_del_d_cnt);

#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);
  ctxp_octx->ctxp = cbctx;
  ctxp_octx->octx = octx;
  OCIDFNDYN(octx->curo0,octx->c_rowid_dp,errhp,1,octx->c_rowid_ptr,
       SIZ(OCIRowid*), SQLT_RDD, octx->c_rowid_ind, (dvoid *)ctxp_octx, rid_data);

  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)-1,
       SQLT_CHR);
  OCIDEF(octx->curo1,octx->c_middle_dp1,errhp,4,ordP->c_middle,
       SIZ(ordP->c_middle)-1,SQLT_AFC);
  OCIDEF(octx->curo1,octx->c_last_dp1,errhp,5,ordP->c_last,SIZ(ordP->c_last)-1,
       SQLT_CHR);
  OCIDEF(octx->curo1,octx->o_id_dp1,errhp,6,ADR(ordP->o_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_entry_d_dp1,errhp,7,
       &ordP->o_entry_d_base,SIZ(OCIDate),SQLT_ODT);
  OCIDEF(octx->curo1,octx->o_cr_id_dp1,errhp,8,ADR(ordP->o_carrier_id),
       SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_ol_cnt_dp1,errhp,9,ADR(ordP->o_ol_cnt),
       SIZ(int),SQLT_INT);

/* 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)-1,
       SQLT_CHR);
  OCIDEF(octx->curo2,octx->c_middle_dp2,errhp,3,ordP->c_middle,
       SIZ(ordP->c_middle)-1,SQLT_AFC);
  OCIDEF(octx->curo2,octx->c_last_dp,errhp,4,ordP->c_last,SIZ(ordP->c_last)-1,SQLT_CHR);
  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(OCIDate),SQLT_ODT);
  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,SIZ(OCIDate),
       SQLT_ODT, octx->ol_delivery_d_ind,octx->ol_delivery_d_len,
          octx->ol_delivery_d_rcode);

  OCIBND(octx->curo4,octx->w_id_bp4,errhp,":w_id",ADR(ordP->w_id),SIZ(int),SQLT_INT);
  OCIBND(octx->curo4,octx->d_id_bp4,errhp,":d_id",ADR(ordP->d_id),SIZ(int),SQLT_INT);
  OCIBND(octx->curo4,octx->c_last_bp4,errhp,":c_last",ordP->c_last,SIZ(ordP->c_last),
       SQLT_STR);
  OCIDEF(octx->curo4,octx->c_count_dp,errhp,1,ADR(octx->rcount),SIZ(int),
       SQLT_INT);


#endif
  return (0);

}
```

```
tkvco (ora_cn_data_t *ora_SlotDataP)

{

  int i;
  int rcount;

  /*  */
  ordctx *octx = (ordctx *)ora_SlotDataP->octx;
  defctx *cbctx = (defctx *)ora_SlotDataP->cbctx;
  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;

  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(OCIDate);
  }
  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 PLSQLORD
  octx->ol_i_id_cnt = 0;
  octx->ol_sup_cnt = 0;
  octx->ol_qty_cnt = 0;
  octx->ol_amt_cnt = 0;
  octx->ol_del_d_cnt = 0;
  OCIERROR(errhp,
     OCIStmtExecute(tpcsvc,octx->curo0,errhp,1,0,0,0,OCI_DEFAULT));
#else
retry:
  if(ordP->bylastname)
  {
    cbctx->reexec = FALSE;
    ordP->execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,100,0,0,0,OCI_DEFAULT);
    /* will get OCI_NO_DATA if <100 found */
    if ((ordP->execstatus != OCI_NO_DATA) && (ordP->execstatus != OCI_SUCCESS))
    {
      ordP->errcode=OCIERROR(errhp,ordP->execstatus);
      if((ordP->errcode == NOT_SERIALIZABLE) || (ordP->errcode == RECOVERR))
      {
        OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
        ordP->retries++;
        goto retry;
      } else {
        return -1;
      }
    }
    if (ordP->execstatus == OCI_NO_DATA) /* there are no more rows */
    {
     /* get rowcount, find middle one */

OCIAttrGet(octx->curo0,OCI_HTYPE_STMT,&rcount,NULL,OCI_ATTR_ROW_COUNT,errhp);
      if (rcount <1)
      {
        userlog("ORDERSTATUS  rcount=%d\n",rcount);
        return (-1);
      }
      octx->cust_idx=(rcount+1)/2 ;
    }
    else
    {
      /* count the number of rows */
      ordP->execstatus=OCIStmtExecute(tpcsvc,octx->curo4,errhp,1,0,0,0,OCI_DEFAULT);
      if ((ordP->execstatus != OCI_NO_DATA) && (ordP->execstatus != OCI_SUCCESS))
      {
        if ((ordP->errcode == NOT_SERIALIZABLE) || (ordP->errcode == RECOVERR))
        {
          OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
          ordP->retries++;
          goto retry;
        } else {
          return -1;
        }
      }
      if (octx->rcount+1 < 2*10 )
        octx->cust_idx=(octx->rcount+1)/2 ;
      else            /* */
      {
        cbctx->reexec = TRUE;
        cbctx->count = (octx->rcount+1)/2 ;
        ordP->execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,cbctx->count,
                        0,0,0,OCI_DEFAULT);
        /* will get OCI_NO_DATA if <100 found */
        if (cbctx->count > 0)
        {
          userlog ("did not get all rows ");
          return (-1);
        }

        if ((ordP->execstatus != OCI_NO_DATA) && (ordP->execstatus != OCI_SUCCESS))
        {
          ordP->errcode=OCIERROR(errhp, ordP->execstatus);
          if((ordP->errcode == NOT_SERIALIZABLE) || (ordP->errcode == RECOVERR))
          {
            OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
            ordP->retries++;
            goto retry;
          } else {
            return -1;
          }
        }
        octx->cust_idx=0 ;
      }
    }

    ordP->execstatus=OCIStmtExecute(tpcsvc,octx->curo1,errhp,1,0,0,0,OCI_DEFAULT);
    if (ordP->execstatus |= OCI_SUCCESS)
    {
      ordP->errcode=OCIERROR(errhp,ordP->execstatus);
      OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
      if((ordP->errcode == NOT_SERIALIZABLE) || (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)
      {
        ordP->errcode=OCIERROR(errhp,ordP->execstatus);
        OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
        if((ordP->errcode == NOT_SERIALIZABLE) || (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 )
    {
      ordP->errcode=OCIERROR(errhp,ordP->execstatus);
      OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
      if((ordP->errcode == NOT_SERIALIZABLE) || (ordP->errcode == RECOVERR))
      {
        ordP->retries++;
        goto retry;
      }
      else
      {
        return -1;
      }

    }
#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((char*)ordP->ol_delivery_d[i],"15-09-1911",10);
      else
      {
        ordP->ol_del_len[i]=sizeof(ordP->ol_delivery_d[i]);
        OCIERROR(errhp,OCIDateToText(errhp,&ordP->ol_d_base[i],

(text*)SHORTDATE,strlen(SHORTDATE),(text*)0,0,&ordP->ol_del_len[i],ordP->ol_delivery_d[i]));
      }
/*
                cvtdmy(ol_d_base[i],ol_delivery_d[i]);
*/
    }

   return (0);

}
```

```
void tkvcodone (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;
  }
}
}
```

plpay.c

```
#ifdef RCSID
static char *RCSid =
   "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/plpay.c,v 1.1
1999/04/14 19:03:05 wenjian 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 <oci.h>
#include "tpcc.h"
#include "plora.h"   /*  */
#include "tpccflags.h"

#define SQLTXT_INIT "BEGIN initpay.pay_init; END;"
#define SQLTXT_STP  "begin payment.dopayment(:w_id,:d_id,:c_w_id,:c_d_id,     \
                :c_id,:by_lname,: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,   \
                :cr_date,:retry); end;"


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 *by_lname_bp;

  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;



/* payctx *pctx; */

tkvcpinit (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 = (void *)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));
#ifdef PLSQLPAY
  /*   prepare the stub for calling plsql stored procedure */
  sprintf ((char*)stmbuf, SQLTXT_STP);
  OCIERROR(errhp,OCIStmtPrepare(pctx->curp0, errhp, stmbuf,
        strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
#else

  /* customer id !=  0, go by last name */

  sqlfile("paynz.sql",stmbuf);
  OCIERROR(errhp,OCIStmtPrepare(pctx->curp0, errhp, stmbuf,
        strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

  /* customer id ==  0, go by last name */

  sqlfile("payz.sql",stmbuf); /* sqlfile opens $O/bench/.../blocks/... */
  OCIERROR(errhp,OCIStmtPrepare(pctx->curp1, errhp, stmbuf,
        strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

#endif
  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);
#ifdef PLSQLPAY
   OCIBND(pctx->curp0, pctx->by_lname_bp, errhp,":by_lname",ADR(payP->bylastname),
        SIZ(int), SQLT_INT);
#endif
   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,
         SIZ(OCIDate), SQLT_ODT, &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(float), SQLT_FLT, &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(OCIDate),SQLT_ODT, &pctx->cr_date_ind, &pctx->cr_date_len,
         &pctx->cr_date_rc);
  #ifndef PLSQLPAY

  /* ---- 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,
         SIZ(OCIDate), SQLT_ODT, &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_FLT, &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(OCIDate),SQLT_ODT, &pctx->cr_date_ind, &pctx->cr_date_len,
        &pctx->cr_date_rc);
#endif

  return (0);

}



tkvcp (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;

#ifdef PLSQLPAY

payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp0,errhp,1,0,0,0,OCI_DEFAULT|OCI_COMMI
T_ON_SUCCESS);
#else
 if(payP->bylastname) {

payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp1,errhp,1,0,0,0,OCI_DEFAULT|OCI_COMMI
T_ON_SUCCESS);
 } else {

payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp0,errhp,1,0,0,0,OCI_DEFAULT|OCI_COMMI
T_ON_SUCCESS);
 }
#endif

 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 tkvcpdone (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;
 }
}
```

plsto.c

```
#ifdef RCSID
static char *RCSid =
  "$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/plsto.c,v 1.1
1999/04/14 19:03:05 wenjian 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 "plora.h"  /*  */
#include "tpccflags.h"

#ifdef PLSQLSTO
#define SQLTXT "BEGIN stocklevel.getstocklevel (:w_id, :d_id, :threshold, \
  :low_stock); END;"
#else
#define SQLTXT "SELECT /*+ nocache(stock) */ \
                                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)"
/* query using functional index */
/*
#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_o_id BETWEEN (d_next_o_id - 20) AND (d_next_o_id - 1) AND \
               decode(SIGN(s_quantity -21) , -1, s_w_id*100000 + s_i_id, NULL) \
                                =   ol_w_id*100000 + ol_i_id  AND \
               s_quantity < :threshold;"
```

```
*/

#endif

struct stoctx {
  OCIStmt *curs;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *threshold_bp;
#ifdef PLSQLSTO
  OCIBind *low_stock_bp;
#else
  OCIDefine *low_stock_bp;
#endif
  int norow;
};

typedef struct stoctx stoctx;

/* stoctx *sctx; */


tkvcsinit (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 = (void *)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));
#ifndef PLSQLSTO
  OCIERROR(errhp,
    OCIAttrSet(sctx->curs,OCI_HTYPE_STMT,(dvoid*)&sctx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));
#endif

  /* 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);
#ifdef PLSQLSTO
  OCIBND(sctx->curs,sctx->low_stock_bp,errhp,":low_stock" , ADR(stoP->low_stock),
    sizeof(int), SQLT_INT);
#else
  OCIDEFINE(sctx->curs,sctx->low_stock_bp,errhp,1, ADR(stoP->low_stock),
    sizeof(int), SQLT_INT);
#endif

  return (0);

}



tkvcs (ora_cn_data_t *ora_SlotDataP)

{
 stoctx *sctx = (stoctx *)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)
  {
    stoP->errcode=OCIERROR(errhp,stoP->execstatus);
    OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
    if((stoP->errcode == NOT_SERIALIZABLE) || (stoP->errcode == RECOVERR))
    {
      stoP->retries++;
      goto retry;
    } else {
      return -1;
    }
   }

  return (0);
}



void tkvcsdone (ora_cn_data_t *ora_SlotDataP)

{
  /*  */
  stoctx *sctx = (stoctx *)ora_SlotDataP->sctx;
  if (sctx) {
    free(sctx);
    ora_SlotDataP->sctx = NULL;
  }
  if (ora_SlotDataP->stoP) {
    free(ora_SlotDataP->stoP);
    ora_SlotDataP->stoP = NULL;
  }
```

tpcc.h

```
/*
 * $Header: tpcc.h 7030100.1 95/07/19 15:10:55 plai Generic<base> $ 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

#ifndef FALSE
# define FALSE 0
#endif

#ifndef TRUE
# define TRUE 1
#endif

#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

#define FULLDATE "dd-mon-yy.hh:mi:ss"
#define SHORTDATE "dd-mm-yyyy"



#define DELRT 80.0

extern int tkvcninit ();
extern int tkvcpnit ();
extern int tkvcoinit ();
extern int tkvcdinit ();
extern int tkvcsinit ();

extern int tkvcn ();
extern int tkvcp ();
extern int tkvco ();
extern int tkvcd ();
extern int tkvcs ();

extern void tkvcndone ();
extern void tkvcpdone ();
extern void tkvcodone ();
extern void tkvcddone ();
extern void tkvcsdone ();

extern int tkvcss (); /* for alter session to get memory size and trace */
extern boolean multitranx;
extern int ord_init;


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;
extern int doid[];

#if 0
extern int execstatus;
extern int errcode;

extern OCIEnv *tpcenv;
extern OCIServer *tpcsrv;
extern OCIError *errhp;
extern OCISvcCtx *tpcsvc;
extern OCISession *tpcusr;
extern OCIStmt *curntest;
/* The bind and define handles for each transaction are
  included in their respective header files. */



/* for stock-level transaction */

extern int w_id;
extern int d_id;
extern int c_id;
extern int threshold;
extern int low_stock;

/* for delivery transaction */

extern int del_o_id[10];
extern int carrier_id;
extern int retries;

/* for order-status transaction */

extern int bylastname;
extern char c_last[17];
extern char c_first[17];
extern char c_middle[3];
extern double c_balance;
extern int o_id;
extern text o_entry_d[20];
extern int o_carrier_id;
extern int o_ol_cnt;
extern int ol_supply_w_id[15];
extern int ol_i_id[15];
extern int ol_quantity[15];
extern int ol_amount[15];
ub4    ol_del_len[15];
extern text ol_delivery_d[15][11];

/* for payment transaction */

extern int c_w_id;
extern int c_d_id;
extern int h_amount;
extern char w_street_1[21];
extern char w_street_2[21];
extern char w_city[21];
extern char w_state[3];
extern char w_zip[10];
extern char d_street_1[21];
extern char d_street_2[21];
extern char d_city[21];
extern char d_state[3];
extern char d_zip[10];
extern char c_street_1[21];
extern char c_street_2[21];
extern char c_city[21];
extern char c_state[3];
extern char c_zip[10];
extern char c_phone[17];
extern text c_since_d[11];
extern char c_credit[3];
extern int c_credit_lim;
extern float c_discount;
extern char c_data[201];
extern text h_date[20];

/* for new order transaction */

extern int nol_i_id[15];
extern int nol_supply_w_id[15];
extern int nol_quantity[15];
extern int nol_quanti10[15];
extern int nol_quanti91[15];
extern int nol_ytdqty[15];
extern int nol_amount[15];
extern int o_all_local;
extern float w_tax;
extern float d_tax;
extern float total_amount;
extern char i_name[15][25];
extern int i_name_strlen[15];
extern ub2 i_name_strlen_len[15];
extern ub2 i_name_strlen_rcode[15];
extern ub4 i_name_strlen_csize;
extern int s_quantity[15];
extern char brand_gen[15];
extern ub2 brand_gen_len[15];
extern ub2 brand_gen_rcode[15];
extern ub4 brand_gen_csize;
extern int i_price[15];
extern char brand_generic[15][1];
extern int status;
extern int tracelevel;

/* Miscellaneous */
extern OCIDate cr_date;
extern OCIDate c_since;
extern OCIDate o_entry_d_base;
extern OCIDate ol_d_base[15];
#endif

#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 min(x,y)  (((x) < (y)) ? (x) : (y))

#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,cu) \
      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),(cu),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);

#define OCIDFNDYN(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,ctxp,cbf_data) \
      ocierror(__FILE__,__LINE__,(errp),\
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                                                      (dvoid**)0));\
      ocierror(__FILE__,__LINE__,(errp),\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv), (progvl),(ftype),\
                           (indp),NULL,NULL, OCI_DYNAMIC_FETCH));\
      ocierror(__FILE__,__LINE__,(errp),\
      OCIDefineDynamic((dfnp),(errp),(ctxp), (cbf_data)));



/* 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
```

tpcc_info.h

```
/*
 * $Header: tpcc.h 7030100.1 95/07/19 15:10:55 plai Generic<base> $ 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_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/server/ora8.1_mt/RCS/tpccpl.c,v 1.2
1999/04/14 22:51:42 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*+====================================================================+
|      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.
 +====================================================================*/


/**  Multi Threading Support
 *
 *  The code in this file supports either single threaded mode or
 *  multi-threaded mode.
 *  In the single threaded mode there is one global connection for
 *  the process.  That connection is stored in the global variable connectionP
 *
 *  In multi-threaded mode there is a connection for each thread.  A
 *  pointer to that connection is stored in the thread slot (or thread
 *  specific data) that is supplied by pthread.
 *  There are two options for the multi-threaded mode explained later:
 *    - preallocated connections
 *    - connections on demand
 *  Before a thread talks to the database it obtains a connection by calling
 *  get_cn(). This function retrieves the thread slot and checks to see
 *  if there is connection information there.  If there is, it returns
 *  that connection.  Otherwise, it initializes the data.
 *  If all connections are preallocated then during initialization the server
 *  creates an array of connections: cn_array, with the target number of connections
 *  (supplied to the function TPCinit).  The get_cn() function than takes the next
 *  available connection from that array.  If connections are not preallocated than
 *  they are allocated on demand.  get_cn will then create a new connection (dynamically
 *  allocated the structure) and initialize it.
 *  In either case the get_cn function will store the reference to the initialized
 *  connection in the thread slot for future requests.
 *  Every thread will initialize the thread slot once before the first DB call
 *  and after that will reuse the connection.
 *  In either mode there is one connection per active thread and each thread
 *  has a different connection; two threads will never shared the same connection.
 *
 */

#include <stdio.h>
#include <time.h>
#include "tpcc.h"
#ifdef TUX
#include <userlog.h>
#else
#include <stdarg.h>
#endif
#ifdef MULTI_THREADED
#include <dce/pthread.h>
#endif
#include "plora.h"


#define SQLTXT "alter session set isolation_level = serializable"
#define SQLTXTTRC "alter session set sql_trace = true"
#define SQLTXTTIM "alter session set timed_statistics = true"

int proc_no;
static char *db_uid;
static char *db_pwd;

#ifdef MULTI_THREADED

static int preallocate_cn = 1;

/* 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;
ora_cn_data_t *cn_array = NULL;
int num_connections = 0;
int cn_id = 0;
#else
ora_cn_data_t *connectionP = NULL;
#endif

/** Delivery file infomation: Global.
  * One output file for deliveries for the server
  */
static char delivery_file_name[80];

FILE *lfp;
FILE *fopen ();

#ifdef ORA_NT
extern double dpbtimef();
#define gettime dpbtimef
#else
double gettime ();
#endif

/** Initialization of one connection  */
static void initOCIhandles(ora_cn_data_t *cn_dataP, char* uid, char *pwd);
/** Server initialization: all globals and all connections */
static int init_global_data(int);

#ifdef MULTI_THREADED
static ora_cn_data_t *get_cn();
#else  /*  Single Threaded: one global connection  */
#define get_cn() connectionP
#endif

static int init_cn_data(ora_cn_data_t *dataP);
static void clean_cn(void *ptr);

extern char oracle_home[256];

/* NewOrder Binding stuff */

#ifndef TUX
void userlog (char* fmtp, ...)
{
  va_list va;
  va_start(va,fmtp);
  vfprintf(stderr,fmtp,va);
  va_end(va);
}
#endif

/* 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;
  sb4 lstat;
  ub4 recno=2;

  switch (status) {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_SUCCESS_WITH_INFO\n");
    lstat = OCIErrorGet (errhp, recno++, (text *) NULL, &errcode, errbuf,
                         (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    fprintf(stderr,"Error - %s\n", errbuf);
    break;
  case OCI_NEED_DATA:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_NEED_DATA\n");
    return (IRRECERR);
  case OCI_NO_DATA:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_NO_DATA\n");
    return (IRRECERR);
  case OCI_ERROR:
    lstat = OCIErrorGet (errhp, (ub4) 1,
                                  (text *) NULL, &errcode, errbuf,
                                        (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
```

```
   if (errcode == NOT_SERIALIZABLE) return (errcode);
   while (lstat != OCI_NO_DATA)
   {
     fprintf(stderr,"Module %s Line %d\n", fname, lineno);
     fprintf(stderr,"Error - %s\n", errbuf);
     lstat = OCIErrorGet (errhp, recno++, (text *) NULL, &errcode, errbuf,
                  (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
   }
   return (errcode);
/* vmm313    TPCexit(1); */
/* vmm313    exit(1); */
  case OCI_INVALID_HANDLE:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_INVALID_HANDLE\n");
    TPCexit(1);
    exit(-1);
  case OCI_STILL_EXECUTING:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_STILL_EXECUTE\n");
    return (IRRECERR);
  case OCI_CONTINUE:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_CONTINUE\n");
    return (IRRECERR);
  default:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Status - %s\n", status);
    return (IRRECERR);
  }
  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;
char realfile[512];

#ifdef  DEBUG
  fprintf(stderr, "sqlfile() fnam: %s,  linebuf: %#x\n", fnam, linebuf);
#endif

   sprintf(realfile,"%s/bench/tpc/tpcc/blocks/%s",oracle_home,fnam);
   fd = vopen(realfile,"r");
   while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
   {
      nulpt = strlen((char *)linebuf);
   }
   return(nulpt);
}

#ifdef NOT
void vgetdate (unsigned char *oradt)
{
 struct tm *loctime;
 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 */
loctime = localtime( &int_time);

 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);

       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",
                  day,month,year,hour,min,sec);

       return;
}
#endif


TPCexit ()

{
  if (lfp) {
     fclose (lfp);
     lfp = NULL;
  }
#ifndef MULTI_THREADED
  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) {
      OCIServer *tpcsrv;
                OCISession *tpcusr;
                OCIEnv *tpcenv;
                OCIError *errhp;
                OCISvcCtx *tpcsvc;

      err_printf("clean_cn, Freeing OCI handles\n");
      tkvcndone(cn_dataP);
      tkvcpdone(cn_dataP);
      tkvcodone(cn_dataP);
      tkvcddone(cn_dataP);
      tkvcsdone(cn_dataP);

      /* free OCI handles */
      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);
                }

      err_printf("free_handles> free cn_dataP\n");
  }
}


TPCinit (id, uid, pwd, numCn, dvryFileName)
int id;
char *uid;
char *pwd;
int  numCn;
char *dvryFileName;
{

  int i;
  text stmbuf[100];

  fprintf(stderr, "TPCinit id %d, uid %s pwd %s\n", id, uid, pwd);

  proc_no = id;
  db_uid = (char *)calloc(strlen(uid) + 1, sizeof(char));
  strcpy(db_uid, uid);
  db_pwd = (char *)calloc(strlen(pwd) + 1, sizeof(char));
  strcpy(db_pwd, pwd);

  sprintf (delivery_file_name, "%s.%d", dvryFileName, proc_no);
  lfp = NULL;  /*  The file will be opened on demand  */

#ifdef USE_ORACLE_WAY /* the original oracle code open lfp here */
  if ((lfp = fopen (delivery_file_name, "w")) == NULL) {
#ifdef TUX
      userlog ("Error in TPC-C server %d: Failed to open %s\n",
            proc_no, delivery_file_name);
#else
      fprintf (stderr, "Error in TPC-C server %d: Failed to open %s\n",
            proc_no, delivery_file_name);
#endif
      return (-1);
  }
#endif

  OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0,0); /* check tpccpl.c */

  if (init_global_data(numCn))
    return -1;

  err_printf("Initialized %d connection(s) to DB.\n", numCn);
  return (0);

}
```

```
static void initOCIhandles(ora_cn_data_t *cn_dataP, char *uid, char *pwd)
{
  int tracelevel = 0;  /* new define */
  OCIDate cr_date;
  text stmbuf[100];
  OCIEnv *tpcenv;
  OCIServer *tpcsrv;
  OCIError *errhp;
  OCISvcCtx *tpcsvc;
  OCISession *tpcusr;
  OCIStmt *curi;

  OCIEnvInit(&tpcenv, OCI_DEFAULT, 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_SERVER, errhp);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid **)0);
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(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_SESSION, 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);


/*
This is done in cvdrv.c
  if (tracelevel == 2) {
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
    memset(stmbuf,0,100);
    sprintf ((char *) stmbuf, SQLTXTTRC);
    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);
  }
  */
  if (tracelevel == 3) {
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
    memset(stmbuf,0,100);
    sprintf ((char *) stmbuf, SQLTXTTIM);
    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);
  }


  OCIERROR(errhp,OCIDateSysDate(errhp,&cr_date));

  /*  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;
}

/*
 *  init_global_data
 *
 *  Called once during initialization to initialize DB connections
 */
static int init_global_data(int numCn)
{
  int status = 0;
#ifdef MULTI_THREADED
  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);
  if (preallocate_cn) {
     int i, env_val;
     char *env_str;
```

```
    ora_cn_data_t *curP;

    pthread_mutex_lock(&init_lock);
    num_connections = numCn;
    if (num_connections < 1) num_connections = 1;
    err_printf("Preallocating %d connections to oracle\n", num_connections);
    cn_array = (ora_cn_data_t *)calloc(num_connections, sizeof(ora_cn_data_t));
    if (cn_array == NULL) {
                err_printf("Failed to allocated %d entried for CN array\n",
                                num_connections);
                exit(3);
    }
    for (i=0, curP = cn_array; i<num_connections; i++, curP++) {
                memset(curP, (char)0, sizeof(*curP));
                if (status = init_cn_data(curP))
                  return status;
    }
    pthread_mutex_unlock(&init_lock);
  }
#else
  connectionP = malloc(sizeof(*connectionP));
  memset(connectionP, (char)0, sizeof(*connectionP));
  if (status = init_cn_data(connectionP))
    return status;
#endif
}

/*
 *  init_cn_data
 *      Initializes all the transactions for a single connection
 */
static int init_cn_data(ora_cn_data_t *cnP)
{
    int status;

    initOCIhandles(cnP, db_uid, db_pwd);

    if (status = tkvcninit (cnP)) {  /* new order */
                err_printf("tkvcninit failed: %d\n", status);
                TPCexit ();
                return (status);
    }

    if (status = tkvcpinit (cnP)) {   /* payment */
                err_printf("tkvcpinit failed: %d\n", status);
                TPCexit ();
                return (status);
    }

    if (status = tkvcoinit (cnP)) {   /* order status */
                err_printf("tkvcoinit failed: %d\n", status);
                TPCexit ();
                return (status);
    }

    if (status = tkvcdinit (cnP)) {   /* delivery */
                err_printf("tkvcdinit failed: %d\n", status);
                TPCexit ();
                return (status);
    }

    if (status = tkvcsinit (cnP)) {   /* stock level */
                err_printf("tkvcsinit failed: %d\n", status);
                TPCexit ();
                return (status);
    }
    return 0;
}

#ifdef MULTI_THREADED
/*
 * 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()
{
    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.
     *  There is a special case for the single connection case:  If there
     *  is exactly one connection then it is global and not per thread.
     *  There may be maany threads but it is assumed that the application is
     *  responsible for synchronizing the threads so that no two threads
     *  ever use the connection at the same time.
     */
    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");

                if (preallocate_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);
                }
        pthread_setspecific(thread_key, dataP);  /*  Store it  */

        err_printf("get_cn> initialized connection 0x%x\n", dataP);
                pthread_mutex_unlock(&init_lock);

    }
    return dataP;
}
#endif

/*****   The Transaction Code  *****/

TPCnew (str)

struct newstruct *str;

{
    int i;
    ora_cn_data_t *cn_dataP = get_cn();
    global_newOrder_t *newP = cn_dataP->globals;
    OCIError *errhp = cn_dataP->errhp;

    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); */

    OCIERROR(errhp,OCIDateSysDate(errhp,&newP->cr_date));

    if (str->newout.terror = tkvcn (cn_dataP)) {
      if (str->newout.terror != RECOVERR)
        str->newout.terror = IRRECERR;
      str->newout.retry = newP->retries;
      return (-1);
    }

    /* fill in date for o_entry_d from time in beginning of txn*/
/*
    cvtdmyhms(newP->cr_date,newP->o_entry_d);
*/
     newP->datelen = sizeof(newP->o_entry_d);
     OCIERROR(errhp,
        OCIDateToText(errhp,&newP->cr_date,(text*)FULLDATE,SIZ(FULLDATE),(text*)0,0,
                &newP->datelen,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 = newP->c_discount;
    str->newout.w_tax = newP->w_tax;
    str->newout.d_tax = newP->d_tax;
    strncpy (str->newout.o_entry_d, (char*)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;

    return (0);

}
```

```
TPCpay (str)

struct paystruct *str;

{
  ora_cn_data_t *cn_dataP = get_cn();
  global_payment_t *payP = cn_dataP->payP;
  OCIError *errhp = cn_dataP->errhp;

  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);  */
  OCIERROR(errhp,OCIDateSysDate(errhp,&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 = tkvcp (cn_dataP)) {
    if (str->payout.terror != RECOVERR)
      str->payout.terror = IRRECERR;
    return (-1);
  }

/*
  cvtdmyhms(cr_date,h_date);
*/
      payP->hlen=SIZ(payP->h_date);
      OCIERROR(errhp,OCIDateToText(errhp,&payP->cr_date,
           (text*)FULLDATE,strlen(FULLDATE),(text*)0,0,&payP->hlen,payP->h_date));

/*
  cvtdmy(c_since,c_since_d);
*/
      payP->sincelen=SIZ(payP->c_since_d);
      OCIERROR(errhp,OCIDateToText(errhp,&payP->c_since,
        (text*)SHORTDATE,strlen(SHORTDATE),(text*)0,0,&payP->sincelen,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, (char*)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 = payP->c_discount;
  str->payout.c_balance = (float)(payP->c_balance)/100;
  strncpy (str->payout.c_data, payP->c_data, 201);
  strncpy (str->payout.h_date, (char*)payP->h_date, 20);
  str->payout.retry = payP->retries;

  return (0);

}



TPCord (str)

struct ordstruct *str;

{
  ora_cn_data_t *cn_dataP = get_cn();
  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 = tkvco (cn_dataP)) {
    if (str->ordout.terror != RECOVERR)
      str->ordout.terror = IRRECERR;
    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, (char*)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((char*)ordP->ol_delivery_d[i],"15-09-1911") )
                strncpy((char*)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], (char*)ordP->ol_delivery_d[i], 11);
  }
  str->ordout.retry = ordP->retries;

  return (0);
}



TPCdel (str)

struct delstruct *str;

{
  ora_cn_data_t *cn_dataP = get_cn();
  global_delivery_t *delP = cn_dataP->delP;
  OCIError *errhp = cn_dataP->errhp;
  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) {
#ifdef MULTI_THREADED
    pthread_mutex_lock(&dvry_log_lock);
#endif
    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);
#ifdef MULTI_THREADED
        pthread_mutex_unlock(&dvry_log_lock);
#endif
        return(-1);
      }
      err_printf("Opened delivery file %s\n", delivery_file_name);
    }
#ifdef MULTI_THREADED
    pthread_mutex_unlock(&dvry_log_lock);
#endif
  }

  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(cr_date);  */
  OCIERROR(errhp,OCIDateSysDate(errhp,&delP->cr_date));

  if (str->delout.terror = tkvcd (cn_dataP)) {
    if(str->delout.terror == DEL_ERROR)
      return DEL_ERROR;
    if (str->delout.terror != RECOVERR)
```

```
      str->delout.terror = IRRECERR;
    return (-1);
  }

#ifndef USE_ORACLE_DVRY_FORMAT
  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;
#else
  tr_end = gettime ();
#endif

#ifdef MULTI_THREADED
  pthread_mutex_lock(&dvry_log_lock);
#endif
#ifndef USE_ORACLE_DVRY_FORMAT
  queue_time = str->delin.qtime;
  pos = 0;
  ++tran_cntr;

  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 new orders.\n", i+1);
        }
      }
    }
  }

  fprintf(lfp, "%send-time: %.3f\n", stdout_buf, tr_end);
  fflush (lfp);

#else /* USE_ORACLE_DVRY_FORMAT */
  fprintf (lfp, "%d %d %f %f %d %d", 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);
  for (i = 0; i < 10; i++) {
    fprintf (lfp, " %d %d", i + 1, delP->del_o_id[i]);
    if (delP->del_o_id[i] <= 0) {
#ifdef TUX
      userlog ("DELIVERY: no new order for w_id: %d, d_id %d\n",
            delP->w_id, i + 1);
#else
      fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
            delP->w_id, i + 1);
#endif
    }
  }
  fprintf (lfp, " %d\n", delP->retries);
#endif

#ifdef MULTI_THREADED
  pthread_mutex_unlock(&dvry_log_lock);
#endif
  str->delout.terror = NOERR;
  str->delout.retry = delP->retries;

  return (0);

}



TPCsto (str)

struct stostruct *str;

{
  ora_cn_data_t *cn_dataP = get_cn();
  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 = tkvcs (cn_dataP)) {
    if (str->stoout.terror != RECOVERR)
      str->stoout.terror = IRRECERR;
    return (-1);
  }

  str->stoout.terror = NOERR;
  str->stoout.low_stock = stoP->low_stock;
  str->stoout.retry = stoP->retries;

  return (0);

}
```

initpay.sql

```
CREATE OR REPLACE PACKAGE initpay
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 initpay;
/
CREATE OR REPLACE PACKAGE BODY initpay AS
 PROCEDURE pay_init IS
 BEGIN
   NULL;
 END pay_init;
END initpay;
/
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 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 initpay.ware_name, :w_street_1, :w_street_2, :w_city,
              :w_state, :w_zip;

        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 initpay.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%NOTFOUND THEN
          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 = initpay.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 initpay.dist_name,:d_street_1,:d_street_2,:d_city,:d_state,
              :d_zip;
       IF SQL%NOTFOUND THEN
        raise NO_DATA_FOUND;
       END IF;


      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, initpay.ware_name || '    ' || initpay.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 */
    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 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 initpay.ware_name,
              :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


--Bulk fetch
      SELECT rowid
      BULK COLLECT INTO initpay.row_id
      FROM customer
      WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last = :c_last
      ORDER BY c_last, c_d_id, c_w_id, c_first;

--Store number of rows processed
     initpay.c_num := sql%rowcount;
     initpay.cust_rowid := initpay.row_id((initpay.c_num) / 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 = initpay.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 = initpay.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 initpay.dist_name, :d_street_1, :d_street_2, :d_city,
             :d_state, :d_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, initpay.ware_name || '    ' || initpay.dist_name);

--Sanjay-No commit needed iff Commit on Success done
--      COMMIT;
      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_cust
(w_id, w_tax, c_id, c_d_id, c_w_id, c_discount, c_last, c_credit)
as select w.w_id, w.w_tax,
        c.c_id, c.c_d_id, c.c_w_id, c.c_discount, c.c_last, c.c_credit
  from customer c, warehouse w
  where w.w_id = c.c_w_id
/

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

```
# $Header: p_run.ora 7030100.1 95/07/14 18:49:15 plai Generic<base> $ Copyr (c) 1993 Oracle
#
#===============================================================+
#       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA     |
#              OPEN SYSTEMS PERFORMANCE GROUP                   |
#                 All Rights Reserved                           |
#===============================================================+
# FILENAME
#      p_run.ora
# DESCRIPTION
#      Oracle parameter file for running TPC-C.
#================================================================
#
control_files = /dev/rlvtpcc_cntl1, /dev/rlvtpcc_cntl2, /dev/rlvtpcc_cntl3

disk_asynch_io          = TRUE

#_huge_sga_base_address  = 0x400000000
db_writer_processes=1
```

```
recovery_parallelism = 20
#compatible            = 8.0.4.0.0S
compatible             = 8.1.0
db_name                =  tpcc
db_files                          = 300
db_block_size                  =4096
# This will trace ORA00064 error
# event                = "64 trace name  errostack, level 2";
dml_locks              = 500
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            = 0
max_rollback_segments      = 400
max_dump_file_size           = 3000
open_cursors           = 400
sessions               = 1000
transactions           = 1000
distributed_transactions = 0
transactions_per_rollback_segment = 1
rollback_segments      = (t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,  \
t16,t17,t18,t19,t20,t21,t22,t23,t24,t25,t26,t27,t28,t29,t30,  \
t31,t32,t33,t34,t35,t36,t37,t38,t39,t40,t41,t42,t43,t44,t45,  \
t46,t47,t48,t49,t50,t51,t52,t53,t54,t55,t56,t57,t58,t59,t60,  \
t61,t62,t63,t64,t65,t66,t67,t68,t69,t70,t71,t72,t73,t74,t75,  \
t76,t77,t78,t79,t80,t81,t82,t83,t84,t85,t86,t87,t88,t89,t90,  \
t91,t92,t93,t94,t95,t96,t97,t98,t99)



#_discrete_transactions_enabled = FALSE
#_disable_incremental_checkpoints = TRUE
cursor_space_for_time        = TRUE
replication_dependency_tracking=FALSE
# log_small_entry_max_size = 80
processes              = 200
# shared_pool_size = 100000000
# shared_pool_reserved_size = 20000000
shared_pool_size = 30000000
db_block_lru_latches = 24
#db_block_lru_latches = 8

# log_checkpoint_timeout = 0
# hash_area_size = 0
# lock_sga               = TRUE
# fast_start_io_target    = 0


#buffer_pool_recycle =  (buffers:1000,lru_latches:2)
buffer_pool_recycle =  (buffers:1000,lru_latches:4)
#buffer_pool_keep    =  (buffers:1277952, lru_latches:2)
_spin_count = 8000

# enqueue_resources      = 100000
# timed_statistics       = TRUE
log_buffer             = 2097152

# _db_block_write_batch   = 4096
# db_block_checkpoint_batch = 2048


#db_block_buffers =      1225292
#db_block_buffers =      1625292
#db_block_buffers =      1753292
db_block_buffers =      1757292
_db_block_hash_buckets =   583430
#_db_block_hash_buckets =  582078
#_db_block_hash_buckets =  539584


db_block_max_dirty_target=0
```

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 silver

## The number PAs each server is configured to have
set MT_PAS 5
set MT_THREADS 2

set MT_DVRY_PAS 1
set DVRY_THREADS 2

set DB_NAME tpcc

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
      set flag 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}/server_stats"
   mas create  ${mas_name} -node ${node} \
                    -executable ${TPCC_BIN_DIR}/server_stats\
                    -commandLineArgs "$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 S70

                OS PARAMETERS

keylock     normal      State of system keylock at boot time          False
maxbuf      20          Maximum number of pages in block I/O BUFFER CACHE True
maxmbuf     0           Maximum Kbytes of real memory allowed for MBUFS   True
```

```
maxuproc    20000        Maximum number of PROCESSES allowed per user      True
autorestart  false       Automatically REBOOT system after a crash         True
iostat      false        Continuously maintain DISK I/O history            True
realmem     8388608      Amount of usable physical memory in Kbytes        False
conslogin    enable      System Console Login                              False
fwversion    IBM,vaSS99071 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,7026-H70  Machine name                                    False
systemid     IBM,011004981 Hardware system identifier                      False
boottype     disk        N/A                                               False
SW_dist_intr false       Enable SW distribution of interrupts              True
```

Appendix C: Database Setup Code

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  Oracle Corp, 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
  set echo on
  alter tablespace $1 add datafile '$2' size $3 reuse;
  exit;
!

if [ -d ./outdir ]
then
  echo `date` >> ./outdir/${FILE}.addf
fi
```

addroll.sh

```
#!/bin/ksh
#=============================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#=============================================================+
# FILENAME
#     addts.sh
# DESCRIPTION
#     Add tablespace to database.
# USAGE
#     addts.sh <tablespace> <data file> <size>
#=============================================================*/
echo 'ORACLE_HOME=' $ORACLE_HOME
echo 'ORACLE_SID=' $ORACLE_SID

FILE=`basename $1`

if [ -d ./outdir ]
then
  echo `date` > ./outdir/${FILE}.addts
fi

svrmgrl <<!
  connect internal
  create tablespace roll datafile '$1' size $2 reuse extent management local uniform size 40Knologging
;

  exit;
!

if [ -d ./outdir ]
then
  echo `date` >> ./outdir/${FILE}.addts
fi
```

addts.sh

```
#!/bin/ksh
#=============================================================+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#=============================================================+
# FILENAME
#     addts.sh
# DESCRIPTION
#     Add tablespace to database.
# USAGE
#     addts.sh <tablespace> <data file> <size>
#=============================================================*/

FILE=`basename $2`

if [ -d ./outdir ]
then
  echo `date` > ./outdir/${FILE}.addts
fi

svrmgrl <<!
  connect internal
  set echo on
  create tablespace $1 datafile '$2' size $3 reuse extent management local uniform size $4nologging ;

  exit;
!

if [ -d ./outdir ]
then
  echo `date` >> ./outdir/${FILE}.addts
fi
```

alter_temp.sh

```
#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#=============================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#=============================================================+
# NAME
#     benchsetup
# DESCRIPTION
#     Usage: benchsetup.sh [options]
#=============================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH
```

```
sqlplus system/manager <<!
  alter user tpcc temporary tablespace temp;
  quit;
!

svrmgrl <<!
  connect internal
  alter tablespace temp
    default storage (initial 300M next 300M pctincrease 0);
  exit;
!
```

benchdb.sh

```
#
# benchdb.sh 8030100 98/7/7 15:45 vmakhija
# Copyr (c) 1998 Oracle
#
#
#=================================================================+
#       Copyright (c) 1997  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                   All Rights Reserved                           |
#=================================================================+
# FILENAME
#      benchdb.sh
# DESCRIPTION
#      Usage: benchdb.sh [options]
#           -n          do not create new  tpcc database
#           -c          do not run catalog scripts
#=================================================================
#


BENCH_HOME=$ORACLE_HOME/bench/tpc
TPCC_ADMIN=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k/admin

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 maxdatafiles 400
          datafile '/dev/rlvsys1' size  383M reuse
          logfile  '/dev/rlvlog1' size 7999M reuse,
                   '/dev/rlvlog2' size 7999M reuse;
      exit
!

#
# Create more rollback segments
#

svrmgrl <<!
  connect internal
  create rollback segment s1  storage (initial 200k minextents 2 next 200k);
  create rollback segment s2  storage (initial 200k minextents 2 next 200k);
  create rollback segment s3  storage (initial 200k minextents 2 next 200k);
  create rollback segment s4  storage (initial 200k minextents 2 next 200k);
  create rollback segment s5  storage (initial 200k minextents 2 next 200k);
  create rollback segment s6  storage (initial 200k minextents 2 next 200k);
  create rollback segment s7  storage (initial 200k minextents 2 next 200k);
  create rollback segment s8  storage (initial 200k minextents 2 next 200k);
  create rollback segment s9  storage (initial 200k minextents 2 next 200k);
  create rollback segment s10 storage (initial 200k minextents 2 next 200k);
  create rollback segment s11 storage (initial 200k minextents 2 next 200k);
  create rollback segment s12 storage (initial 200k minextents 2 next 200k);
  create rollback segment s13 storage (initial 200k minextents 2 next 200k);
  create rollback segment s14 storage (initial 200k minextents 2 next 200k);
  create rollback segment s15 storage (initial 200k minextents 2 next 200k);
  create rollback segment s16 storage (initial 200k minextents 2 next 200k);
  create rollback segment s17 storage (initial 200k minextents 2 next 200k);
  create rollback segment s18 storage (initial 200k minextents 2 next 200k);
  create rollback segment s19 storage (initial 200k minextents 2 next 200k);
  create rollback segment s20 storage (initial 200k minextents 2 next 200k);
  create rollback segment s21 storage (initial 200k minextents 2 next 200k);
  create rollback segment s22 storage (initial 200k minextents 2 next 200k);
  create rollback segment s23 storage (initial 200k minextents 2 next 200k);
  create rollback segment s24 storage (initial 200k minextents 2 next 200k);
  create rollback segment s25 storage (initial 200k minextents 2 next 200k);
  create rollback segment s26 storage (initial 200k minextents 2 next 200k);
  create rollback segment s27 storage (initial 200k minextents 2 next 200k);
  create rollback segment s28 storage (initial 200k minextents 2 next 200k);
  create rollback segment s29 storage (initial 200k minextents 2 next 200k);
  create rollback segment s30 storage (initial 200k minextents 2 next 200k);
  shutdown;
  exit;
!
fi

#
#  Startup database with params file that includes new rollback segments
#

svrmgrl <<!
      connect internal
      startup pfile=$TPCC_ADMIN/p_build.ora;
      connect system/manager
      create tablespace temp datafile
        '/dev/rlvtemp0'  size   1999M reuse;
      exit;
!

#
#      Add tablespaces in parallel
#
addts.sh misc  /dev/rlvmisc 199M 10M &
addts.sh stock /dev/rlvstock1 1812M 1810M &
addts.sh ordr /dev/rlvordr1  700M 50M &
addts.sh iord1 /dev/rlviord11 495M 100M &
addts.sh iord2 /dev/rlviord21 799M 100M &
addts.sh ordl /dev/rlvordl1 3439M 100M &
addts.sh nord /dev/rlvnord1 1183M 100M &
addts.sh cust /dev/rlvcust1 783M 770M &
addts.sh icust2 /dev/rlvicust21 559M 100M &
addts.sh hist /dev/rlvhist1  1375M 100M &
addts.sh unused /dev/rlvunused1 1999M 100M &
addroll.sh /dev/rlvroll1 95M &

wait

#
#                  Add datafiles to tablespaces in parallel
#

addfile.sh unused /dev/rlvunused2 1999M &
addfile.sh unused /dev/rlvunused3 1999M &
addfile.sh ordr /dev/rlvordr2  700M &
addfile.sh ordr /dev/rlvordr3  700M &
addfile.sh ordr /dev/rlvordr4  700M &
addfile.sh ordr /dev/rlvordr5  700M &
addfile.sh ordr /dev/rlvordr6  700M &
wait
addfile.sh iord1 /dev/rlviord12 495M &
addfile.sh iord1 /dev/rlviord13 495M &
addfile.sh iord1 /dev/rlviord14 495M &
wait
addfile.sh iord2 /dev/rlviord22 799M &
addfile.sh iord2 /dev/rlviord23 799M &
addfile.sh iord2 /dev/rlviord24 799M &
wait
addfile.sh ordl /dev/rlvordl2 3439M &
addfile.sh ordl /dev/rlvordl3 3439M &
addfile.sh ordl /dev/rlvordl4 3439M &
addfile.sh ordl /dev/rlvordl5 3439M &
addfile.sh ordl /dev/rlvordl6 3439M &
addfile.sh ordl /dev/rlvordl7 3439M &
addfile.sh ordl /dev/rlvordl8 3439M &
addfile.sh ordl /dev/rlvordl9 3439M &
addfile.sh ordl /dev/rlvordl10 3439M &
addfile.sh ordl /dev/rlvordl11 3439M &
addfile.sh ordl /dev/rlvordl12 3439M &
addfile.sh ordl /dev/rlvordl13 3439M &
addfile.sh ordl /dev/rlvordl14 3439M &
addfile.sh ordl /dev/rlvordl15 3439M &
addfile.sh ordl /dev/rlvordl16 3439M &
addfile.sh ordl /dev/rlvordl17 3439M &
addfile.sh ordl /dev/rlvordl18 3439M &
addfile.sh ordl /dev/rlvordl19 3439M &
addfile.sh ordl /dev/rlvordl20 3439M &
addfile.sh ordl /dev/rlvordl21 3439M &
addfile.sh ordl /dev/rlvordl22 3439M &
addfile.sh ordl /dev/rlvordl23 3439M &
addfile.sh ordl /dev/rlvordl24 3439M &
addfile.sh ordl /dev/rlvordl25 3439M &
```

```
addfile.sh ordl /dev/rlvordl26 3439M &
addfile.sh ordl /dev/rlvordl27 3439M &
addfile.sh ordl /dev/rlvordl28 3439M &
wait
addfile.sh stock /dev/rlvstock2 1812M &
addfile.sh stock /dev/rlvstock3 1812M &
addfile.sh stock /dev/rlvstock4 1812M &
addfile.sh stock /dev/rlvstock5 1812M &
addfile.sh stock /dev/rlvstock6 1812M &
addfile.sh stock /dev/rlvstock7 1812M &
addfile.sh stock /dev/rlvstock8 1812M &
addfile.sh stock /dev/rlvstock9 1812M &
addfile.sh stock /dev/rlvstock10 1812M &
addfile.sh stock /dev/rlvstock11 1812M &
addfile.sh stock /dev/rlvstock12 1812M &
addfile.sh stock /dev/rlvstock13 1812M &
addfile.sh stock /dev/rlvstock14 1812M &
addfile.sh stock /dev/rlvstock15 1812M &
wait
addfile.sh stock /dev/rlvstock16 1812M &
addfile.sh stock /dev/rlvstock17 1812M &
addfile.sh stock /dev/rlvstock18 1812M &
addfile.sh stock /dev/rlvstock19 1812M &
addfile.sh stock /dev/rlvstock20 1812M &
addfile.sh stock /dev/rlvstock21 1812M &
addfile.sh stock /dev/rlvstock22 1812M &
addfile.sh stock /dev/rlvstock23 1812M &
addfile.sh stock /dev/rlvstock24 1812M &
addfile.sh stock /dev/rlvstock25 1812M &
addfile.sh stock /dev/rlvstock26 1812M &
addfile.sh stock /dev/rlvstock27 1812M &
addfile.sh stock /dev/rlvstock28 1812M &
addfile.sh stock /dev/rlvstock29 1812M &
addfile.sh stock /dev/rlvstock30 1812M &
wait
addfile.sh icust2 /dev/rlvicust22 559M &
addfile.sh icust2 /dev/rlvicust23 559M &
addfile.sh icust2 /dev/rlvicust24 559M &
wait
addfile.sh cust /dev/rlvcust2 783M &
addfile.sh cust /dev/rlvcust3 783M &
addfile.sh cust /dev/rlvcust4 783M &
addfile.sh cust /dev/rlvcust5 783M &
addfile.sh cust /dev/rlvcust6 783M &
addfile.sh cust /dev/rlvcust7 783M &
addfile.sh cust /dev/rlvcust8 783M &
addfile.sh cust /dev/rlvcust9 783M &
addfile.sh cust /dev/rlvcust10 783M &
addfile.sh cust /dev/rlvcust11 783M &
addfile.sh cust /dev/rlvcust12 783M &
addfile.sh cust /dev/rlvcust13 783M &
addfile.sh cust /dev/rlvcust14 783M &
addfile.sh cust /dev/rlvcust15 783M &
addfile.sh cust /dev/rlvcust16 783M &
addfile.sh cust /dev/rlvcust17 783M &
addfile.sh cust /dev/rlvcust18 783M &
addfile.sh cust /dev/rlvcust19 783M &
addfile.sh cust /dev/rlvcust20 783M &
wait
addfile.sh cust /dev/rlvcust21 783M &
addfile.sh cust /dev/rlvcust22 783M &
addfile.sh cust /dev/rlvcust23 783M &
addfile.sh cust /dev/rlvcust24 783M &
addfile.sh cust /dev/rlvcust25 783M &
addfile.sh cust /dev/rlvcust26 783M &
addfile.sh cust /dev/rlvcust27 783M &
addfile.sh cust /dev/rlvcust28 783M &
addfile.sh cust /dev/rlvcust29 783M &
addfile.sh cust /dev/rlvcust30 783M &
addfile.sh cust /dev/rlvcust31 783M &
addfile.sh cust /dev/rlvcust32 783M &
addfile.sh cust /dev/rlvcust33 783M &
addfile.sh cust /dev/rlvcust34 783M &
addfile.sh cust /dev/rlvcust35 783M &
addfile.sh cust /dev/rlvcust36 783M &
addfile.sh cust /dev/rlvcust37 783M &
addfile.sh cust /dev/rlvcust38 783M &
addfile.sh cust /dev/rlvcust39 783M &
addfile.sh cust /dev/rlvcust40 783M &
wait
addfile.sh cust /dev/rlvcust41 783M &
addfile.sh cust /dev/rlvcust42 783M &
addfile.sh cust /dev/rlvcust43 783M &
addfile.sh cust /dev/rlvcust44 783M &
addfile.sh cust /dev/rlvcust45 783M &
addfile.sh cust /dev/rlvcust46 783M &
addfile.sh cust /dev/rlvcust47 783M &
addfile.sh cust /dev/rlvcust48 783M &
addfile.sh cust /dev/rlvcust49 783M &
addfile.sh cust /dev/rlvcust50 783M &
addfile.sh cust /dev/rlvcust51 783M &
addfile.sh cust /dev/rlvcust52 783M &
addfile.sh cust /dev/rlvcust53 783M &
addfile.sh cust /dev/rlvcust54 783M &
addfile.sh cust /dev/rlvcust55 783M &
addfile.sh cust /dev/rlvcust56 783M &
addfile.sh cust /dev/rlvcust57 783M &
addfile.sh cust /dev/rlvcust58 783M &
wait
addfile.sh hist /dev/rlvhist2 1375M &
addfile.sh hist /dev/rlvhist3 1375M &
addfile.sh hist /dev/rlvhist4 1375M &
wait
addfile.sh temp /dev/rlvtemp1 1999M &
addfile.sh temp /dev/rlvtemp2 1999M &
addfile.sh temp /dev/rlvtemp3 1999M &
addfile.sh temp /dev/rlvtemp4 1999M &
addfile.sh temp /dev/rlvtemp5 1999M &
addfile.sh temp /dev/rlvtemp6 1999M &
addfile.sh temp /dev/rlvtemp7 1999M &
addfile.sh temp /dev/rlvtemp8 1999M &
addfile.sh temp /dev/rlvtemp9 1999M &
addfile.sh temp /dev/rlvtemp10 1999M &
addfile.sh temp /dev/rlvtemp11 1999M &
addfile.sh temp /dev/rlvtemp12 1999M &
addfile.sh temp /dev/rlvtemp13 1999M &
addfile.sh temp /dev/rlvtemp14 1999M &
addfile.sh temp /dev/rlvtemp15 1999M &
wait
#
# run catalog if NO_CAT unset
#

if [ "$NO_CAT" = "" ]
then
svrmgrl <<!
      set echo off;
      connect sys/change_on_install;
      @?/rdbms/admin/catalog;
      @?/rdbms/admin/catproc;
      @?/rdbms/admin/catparr;
      exit;
!
fi
```

benchsetup.sh

```
#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                    All Rights Reserved                          |
#=================================================================+
# NAME
#     benchsetup
# DESCRIPTION
#     Usage: benchsetup.sh [options]
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k/
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH


function usage {
  echo ""
 echo "Usage:       $PROGNAME [<start> <stop>] [<start>] [-step <stepno>]"
  echo "     [<start> <stop>]       - allows user to run a specified"
  echo "                            range of steps."
  echo "     [<start>]              - runs from step number <start> till"
  echo "                            the end of the script."
  echo "     [-step <stepno>]       - runs only step number <stepno> and"
  echo "                            then stops."
  echo ""
  echo "             STEP    FUNCTION"
  echo "-----------------------------------------------------------------"
```

```
  echo "            0     Create  DB."
  echo "            1     Create user  tpcc."
  echo "            2     Create warehouse table."
  echo "            3     Create district table."
  echo "            4     Create history table."
  echo "            5     Create Orders table."
  echo "            6     Create New-order table."
  echo "            7     Create  Orderline table."
  echo "            8     Create Item table."
  echo "            9     Create Customer table."
  echo "            10    Create Stock table."
  echo "            11    Create rollback segments."
  echo "            12    Load New-order."
  echo "            13    Load History."
  echo "            14    Load  Order/Orderline."
  echo "            15    Load Warehouse."
  echo "            16    Load District."
  echo "            17    Load Item."
  echo "            18    Load Customer."
  echo "            19    Load Stock"
  echo "            20    Alter temp space."
  echo "            21    Create Warehouse index."
  echo "            22    Create District index."
  echo "            23    Create Item index."
  echo "            24    Create Customer index."
  echo "            25    Create  Customer2 index."
  echo "            26    Create Stock index."
  echo "            27    Create Orders index."
  echo "            28    Create  Orders2 index."
  echo "            29    Create New-order index."
  echo "            30    Create  Orderline index."
  echo "            31    Re-alter temp space."
  echo "            32    Analyze."
  echo "            33    Create  TPC-C reports tables."
  echo "            34    Create stored  procs."
  echo "            35    Space  rpts / etc."
  echo "            36    Alter extents and Lock tables."
  echo "            37    Delete temp  tablespace."
  echo "            38    Run catalog scripts."
  echo "            39    Shutdown database."
  echo "------------------------------------------------------------"

  exit 1;
}

function runnable {
 if [ -a "./stop" ]
 then
   exit 1;
 fi
 if [ $STEP -ge $START ]
 then
   if [ $STEP -le $END ]
   then
      STEP=‘expr $STEP + 1‘;
      return 0;
   else
      if [ $CONTINUE = 0 ]
      then
         STEP=‘expr $STEP + 1‘;
         return 0;
      fi
   fi
 fi
 STEP=‘expr $STEP + 1‘;
 return 1;
}

case $# in
      0)    usage
            CONTINUE=0
            ;;
      1)    case $1 in
                  -h)   usage
                        ;;
                  *)    START=$1
                        CONTINUE=0
                        ;;
            esac
            ;;
      2)    case $1 in
                  -step) shift
                        START=$1
                        END=$1
                        CONTINUE=1
                        ;;
                  *)    START=$1
                        shift
                        END=$1
                        CONTINUE=1
                        ;;
            esac
            ;;
      *)    usage
            ;;
esac

if [ ! -d $BUILD_HOME ]
then
   mkdir $BUILD_HOME
fi

if [ ! -d $LOAD_SCRIPTS ]
then
   mkdir $LOAD_SCRIPTS
fi



if [ ! -d $OUTDIR ]
then
   mkdir $OUTDIR
fi

if runnable
then
${LOAD_SCRIPTS}/benchdb.sh > ${OUTDIR}/benchdb.out 2>&1
  echo "Switching Logs ..."
  ${TPCC_UTILS}/switchlog.sh >> ${OUTDIR}/switchlog.out 2>&1
fi


if runnable
then
${LOAD_SCRIPTS}/create_user.sh > ${OUTDIR}/create_user.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_ware.sh > ${OUTDIR}/create_ware.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_dist.sh > ${OUTDIR}/create_dist.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_hist.sh > ${OUTDIR}/create_hist.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_ordr.sh > ${OUTDIR}/create_ordr.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_nord.sh > ${OUTDIR}/create_nord.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_ordl.sh > ${OUTDIR}/create_ordl.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_item.sh > ${OUTDIR}/create_item.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_cust.sh > ${OUTDIR}/create_cust.out 2>&1 &
fi

if runnable
then
${LOAD_SCRIPTS}/create_stok.sh > ${OUTDIR}/create_stok.out 2>&1 &
fi

wait

if runnable
then
${LOAD_SCRIPTS}/tpcc_rol.sh > ${OUTDIR}/tpcc_rol.out 2>&1 &
fi

echo "Switching Logs ..."
${TPCC_UTILS}/switchlog.sh >> ${OUTDIR}/switchlog.out 2>&1

if runnable
then
${LOAD_SCRIPTS}/load_nord.sh > ${OUTDIR}/load_nord.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/load_hist.sh > ${OUTDIR}/load_hist.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/load_ordr.sh > ${OUTDIR}/load_ordr.out 2>&1
fi

echo "Switching Logs ..."
```

```
${TPCC_UTILS}/switchlog.sh >> ${OUTDIR}/switchlog.out 2>&1

if runnable
then
${LOAD_SCRIPTS}/load_ware.sh > ${OUTDIR}/load_ware.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/load_dist.sh > ${OUTDIR}/load_dist.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/load_item.sh > ${OUTDIR}/load_item.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/load_cust.sh > ${OUTDIR}/load_cust.out 2>&1 &
fi

if runnable
then
${LOAD_SCRIPTS}/load_stok.sh > ${OUTDIR}/load_stok.out 2>&1 &
fi

wait

echo "Switching Logs ..."
${TPCC_UTILS}/switchlog.sh >> ${OUTDIR}/switchlog.out 2>&1


if runnable
then
${LOAD_SCRIPTS}/alter_temp.sh > ${OUTDIR}/alter_temp.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_iware.sh > ${OUTDIR}/create_iware.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_idist.sh > ${OUTDIR}/create_idist.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_iitem.sh > ${OUTDIR}/create_iitem.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_icust.sh > ${OUTDIR}/create_icust.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_icust2.sh > ${OUTDIR}/create_icust2.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_istok.sh > ${OUTDIR}/create_istok.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_iordr.sh > ${OUTDIR}/create_iordr.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/create_iordr2.sh > ${OUTDIR}/create_iordr2.out 2>&1
fi

if runnable
then
echo "No need to create inord"
#${LOAD_SCRIPTS}/create_inord.sh > ${OUTDIR}/create_inord.out 2>&1
fi

if runnable
then
echo "No need to create iordl"
#${LOAD_SCRIPTS}/create_iordl.sh > ${OUTDIR}/create_iordl.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/realter_temp.sh > ${OUTDIR}/realter_temp.out 2>&1
fi


if runnable
then
sqlplus tpcc/tpcc @$TPCC_SQL/tpcc_ana > ${OUTDIR}/tpcc_ana.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/tpcc_reports.sh > ${OUTDIR}/tpcc_reports.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/tpcc_stored_proc.sh > ${OUTDIR}/tpcc_stored_prod.out 2>&1
$TPCC_UTILS/create_cache_views.sh > ${OUTDIR}/create_cache_views.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/tpcc_misc.sh > ${OUTDIR}/tpcc_misc.out 2>&1
fi

if runnable
then
${LOAD_SCRIPTS}/altundef.sh > ${OUTDIR}/altundef.out 2>&1
${TPCC_UTILS}/dml.sh > ${OUTDIR}/dml.out 2>&1
fi

if runnable
then
svrmgrl <<!
  connect internal;
  drop tablespace temp including contents;
  exit;
fi

if runnable
then
${LOAD_SCRIPTS}/cat.sh > ${OUTDIR}/cat.out 2>&1
fi

if runnable
then
svrmgrl <<!
  connect internal;
  alter system switch logfile;
  alter system switch logfile;
  shutdown;
  exit;
!
fi
```

cat.sh

```
#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#               OPEN SYSTEMS PERFORMANCE GROUP                   |
#                   All Rights Reserved                          |
#================================================================+
# NAME
#     benchsetup
# DESCRIPTION
#     Usage: benchsetup.sh [options]
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH


svrmgrl <<!
  set echo off;
  connect sys/change_on_install;
  @?/rdbms/admin/catparr;
  exit;
```

```
!
```

create_cust.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @cust
```

create_dist.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @dist
```

create_hist.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @hist
```

create_icust.sh

```
#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450
```

```
PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @icust
```

create_icust2.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=============================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA     |
#                OPEN SYSTEMS PERFORMANCE GROUP                 |
#                    All Rights Reserved                        |
#=============================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#=============================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @icust2
```

create_idist.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=============================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA     |
#                OPEN SYSTEMS PERFORMANCE GROUP                 |
#                    All Rights Reserved                        |
#=============================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#=============================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @idist
```

create_iitem.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=============================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA     |
#                OPEN SYSTEMS PERFORMANCE GROUP                 |
#                    All Rights Reserved                        |
#=============================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#=============================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @iitem
```

create_iordr.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=============================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA     |
#                OPEN SYSTEMS PERFORMANCE GROUP                 |
#                    All Rights Reserved                        |
#=============================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#=============================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
```

```
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @iordr
```

create_iordr2.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#               OPEN SYSTEMS PERFORMANCE GROUP                   |
#                  All Rights Reserved                           |
#================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @iordr2
```

create_istok.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#               OPEN SYSTEMS PERFORMANCE GROUP                   |
#                  All Rights Reserved                           |
#================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @istok
```

create_item.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#               OPEN SYSTEMS PERFORMANCE GROUP                   |
#                  All Rights Reserved                           |
#================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @item


create
```

create_iware.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#               OPEN SYSTEMS PERFORMANCE GROUP                   |
#                  All Rights Reserved                           |
#================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
```

```
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @iware
```

create_nord.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                   All Rights Reserved                           |
#=================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>     (# of warehouses)
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @nord
```

create_ordl.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                   All Rights Reserved                           |
#=================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>     (# of warehouses)
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @ordl
```

create_ordr.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                   All Rights Reserved                           |
#=================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>     (# of warehouses)
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @ordr
```

create_stok.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                   All Rights Reserved                           |
#=================================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>     (# of warehouses)
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
```

```
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @stok
```

create_user.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=============================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA   |
#                OPEN SYSTEMS PERFORMANCE GROUP               |
#                   All Rights Reserved                       |
#=============================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#=============================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH


svrmgrl <<!
rem
rem ============================================================+
rem       Copyright (c) 1997  Oracle Corp, Redwood Shores, CA   |
rem                OPEN SYSTEMS PERFORMANCE GROUP               |
rem                   All Rights Reserved                       |
rem ============================================================+
rem  FILENAME
rem      tpcc_user.sql
rem  DESCRIPTION
rem      Create user for TPC-C database.
rem ============================================================
rem

rem
rem Create TPCC userid and connect to it.
rem
      connect internal;
      grant connect,resource,unlimited tablespace to tpcc identified by tpcc;
      alter user tpcc temporary tablespace temp;
      connect tpcc/tpcc;
      exit;
!
```

create_ware.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=============================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA   |
#                OPEN SYSTEMS PERFORMANCE GROUP               |
#                   All Rights Reserved                       |
#=============================================================+
# FILENAME
#      create_<obj>.sh
# DESCRIPTION
#      Usage: create_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#=============================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

sqlplus tpcc/tpcc @ware
```

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;
!
!
```

load_cust.sh

```
#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#=============================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA   |
#                OPEN SYSTEMS PERFORMANCE GROUP               |
#                   All Rights Reserved                       |
#=============================================================+
# NAME
#      benchsetup
# DESCRIPTION
#      Usage: benchsetup.sh [options]
```

```
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH


#
#  Load customer table (in parallel with loading stock table)
#

I=1
SW=1
EW=25
INC=25
while [ $I -le 58 ]
do
  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
```

load_dist.sh

```
#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#=====================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                 |
#                   All Rights Reserved                         |
#=====================================================================+
# NAME
#      benchsetup
# DESCRIPTION
#      Usage: benchsetup.sh [options]
#=====================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH


tpccload -M $MULT -d
```

load_hist.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=====================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                 |
#                   All Rights Reserved                         |
#=====================================================================+
# FILENAME
#      pload.sh
# DESCRIPTION
#      Usage: load_<obj>.sh [options]
#           -mu <multiplier>    (# of warehouses)
#=====================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
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 "Bad  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 $BUILD_HOME ]
then
  mkdir $BUILD_HOME
fi

if [ ! -d $LOAD_SCRIPTS ]
then
  mkdir $LOAD_SCRIPTS
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=363
INC=363
while [ $I -le 3 ]
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
 tpccload -M $MULT -h -g -b $SW -e $MULT > ${LDIR}/hist${I}.dat 2> \
    ${OUTDIR}/hist${I}.out &


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 \
       file=/dev/rlvhist${I} \
       discard=${OUTDIR}/hist${I}.dsc &
  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_item.sh

```
#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                   All Rights Reserved                           |
#=================================================================+
# NAME
#      benchsetup
# DESCRIPTION
#      Usage: benchsetup.sh [options]
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH


tpccload -M $MULT -i
```

load_nord.sh

```
#!/bin/ksh

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                   All Rights Reserved                           |
#=================================================================+
# FILENAME
#      pload.sh
# DESCRIPTION
#      Usage: load_<obj>.sh [options]
#            -mu <multiplier>    (# of warehouses)
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
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 "Bad  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 $BUILD_HOME ]
then
  mkdir $BUILD_HOME
```

```
fi

if [ ! -d $LOAD_SCRIPTS ]
then
  mkdir $LOAD_SCRIPTS
fi

if [ ! -d $LDIR ]
then
  mkdir $LDIR
fi

if [ ! -d $OUTDIR ]
then
  mkdir $OUTDIR
fi

#
#  Load new-order table
#

#I=1
#mknod ${LDIR}/neword${I}.dat p

I=1
SW=1
EW=1450
INC=1450
while [ $I -le 1 ]
do
 tpccload -M $MULT -n -b $SW -e $EW > ${LDIR}/neword${I}.dat 2> \
    ${OUTDIR}/neword${I}.out &
  I=`expr $I + 1`
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

wait
#sleep 30
#
I=1
while [ $I -le 1 ]
do
  sqlldr tpcc/tpcc control=$TPCC_LOADER/neword.ctl \
      log=${OUTDIR}/neword${I}.log \
      bad=${OUTDIR}/neword${I}.bad data=${LDIR}/neword${I}.dat \
      file=/dev/rlvnord${I} \
      discard=${OUTDIR}/neword${I}.dsc
  I=`expr $I + 1`
done

wait

#I=1
#rm -f ${LDIR}/neword${I}.dat
```

load_ordr.sh

```
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#================================================================+
# FILENAME
#      pload.sh
# DESCRIPTION
#      Usage: load_<obj>.sh [options]
#            -mu <multiplier>    (# of warehouses)
#================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
TPCC_LOADER=$BUILD_HOME/loader
LOAD_SCRIPTS=${BUILD_HOME}/scripts
OUTDIR=${BUILD_HOME}/outdir
LDIR=${BUILD_HOME}/data
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
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 "Bad  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 $BUILD_HOME ]
then
  mkdir $BUILD_HOME
fi

if [ ! -d $LOAD_SCRIPTS ]
then
  mkdir $LOAD_SCRIPTS
fi

if [ ! -d $LDIR ]
then
  mkdir $LDIR
fi

if [ ! -d $OUTDIR ]
then
  mkdir $OUTDIR
fi

#
#  Load order and order-line table
#

#I=1
#while [ $I -le 28 ]
#do
#   mknod ${LDIR}/order${I}.dat p
#   mknod ${LDIR}/ordline${I}.dat p
#   I=`expr $I + 1`
#done

I=1
SW=1
EW=51
INC=51
while [ $I -le 27 ]
do
   tpccload -M $MULT -o ${LDIR}/ordline${I}.dat -b $SW -e $EW > \
     ${LDIR}/order${I}.dat 2> ${OUTDIR}/order${I}.out &
  I=`expr $I + 1`
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

   tpccload -M $MULT -o ${LDIR}/ordline${I}.dat -b $SW -e $MULT > \
     ${LDIR}/order${I}.dat 2> ${OUTDIR}/order${I}.out &

wait
sleep 30

I=1
while [ $I -le 28 ]
```

```
do
   J=‘expr $I - 1‘
   J=‘expr $J / 5‘
   J=‘expr $J + 1‘
  sqlldr tpcc/tpcc control=$TPCC_LOADER/order.ctl\
        log=${OUTDIR}/order${I}.log\
        bad=${OUTDIR}/order${I}.bad data=${LDIR}/order${I}.dat\
        file=/dev/rlvordr${J}\
        discard=${OUTDIR}/order${I}.dsc &
  I=‘expr $I + 1‘
done

wait

I=1
while [ $I -le 28 ]
do
   sqlldr tpcc/tpcc control=$TPCC_LOADER/ordline.ctl\
        log=${OUTDIR}/ordline${I}.log\
        bad=${OUTDIR}/ordline${I}.bad data=${LDIR}/ordline${I}.dat\
        file=/dev/rlvordl${I}\
        discard=${OUTDIR}/ordline${I}.dsc
  I=‘expr $I + 1‘
done

wait

#I=1
#while [ $I -le 18 ]
#do
#   rm -f ${LDIR}/order${I}.dat
#   rm -f ${LDIR}/ordline${I}.dat
#   I=‘expr $I + 1‘
#done
```

load_stok.sh

```
# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                    All Rights Reserved                          |
#=================================================================+
# NAME
#      benchsetup
# DESCRIPTION
#      Usage: benchsetup.sh [options]
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

#
#  Load stock table (in parallel with loading customer table)
#

I=1
SI=1
EI=2500
INC=2500
while [ $I -le 40 ]
do
   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
```

load_ware.sh

```
#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                    All Rights Reserved                          |
#=================================================================+
# NAME
#      benchsetup
# DESCRIPTION
#      Usage: benchsetup.sh [options]
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH


tpccload -M $MULT -w
```

realter_temp.sh

```
#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                OPEN SYSTEMS PERFORMANCE GROUP                   |
#                    All Rights Reserved                          |
#=================================================================+
# NAME
#      benchsetup
# DESCRIPTION
#      Usage: benchsetup.sh [options]
#=================================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH


svrmgrl<<!
  connect internal
  alter tablespace temp
    default storage (initial 20K next 20K pctincrease 50);
  exit;
!
```

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 lmode=y <<!
  connect internal;
  alter system switch logfile;
  alter system switch logfile;
  exit;
!
```

tpcc_rol.sh

```
#!/bin/ksh
#
#
#=================================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA       |
#                  OPEN SYSTEMS PERFORMANCE GROUP                 |
#                     All Rights Reserved                         |
#=================================================================+
# FILENAME
#      tpccrol.sh
# DESCRIPTION
#     Script file for creating the roll;back segments.
#=================================================================
#

svrmgrl <<!
  connect internal;
  host date;
  set timing on;


                        CREATE ROLLBACK SEGMENT t1 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t2 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t3 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t4 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t5 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t6 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t7 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t8 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t9 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t10 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t11 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t12 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t13 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t14 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t15 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t16 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t17 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t18 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t19 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t20 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t21 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t22 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t23 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t24 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t25 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t26 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t27 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t28 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t29 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t30 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t31 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t32 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t33 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t34 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t35 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t36 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t37 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t38 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t39 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t40 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t41 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t42 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t43 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t44 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t45 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t46 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t47 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t48 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t49 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t50 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t51 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t52 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t53 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t54 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t55 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t56 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t57 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t58 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t59 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t60 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t61 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t62 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t63 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t64 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t65 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t66 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t67 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t68 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t69 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t70 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t71 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t72 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t73 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t74 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t75 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t76 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t77 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t78 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t79 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t80 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t81 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t82 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t83 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t84 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t85 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t86 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t87 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t88 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t89 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t90 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t91 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t92 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t93 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t94 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t95 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t96 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t97 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t98 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t99 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t100 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t101 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t102 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t103 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t104 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t105 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t106 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t107 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t108 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t109 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t110 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t111 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t112 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t113 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t114 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t115 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t116 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t117 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t118 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t119 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t120 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t121 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t122 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t123 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t124 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t125 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t126 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t127 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t128 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t129 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t130 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t131 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t132 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t133 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t134 TABLESPACE roll;
                        CREATE ROLLBACK SEGMENT t135 TABLESPACE roll;
```

```
CREATE ROLLBACK SEGMENT t136 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t137 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t138 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t139 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t140 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t141 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t142 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t143 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t144 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t145 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t146 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t147 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t148 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t149 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t150 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t151 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t152 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t153 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t154 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t155 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t156 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t157 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t158 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t159 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t160 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t161 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t162 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t163 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t164 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t165 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t166 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t167 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t168 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t169 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t170 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t171 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t172 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t173 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t174 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t175 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t176 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t177 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t178 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t179 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t180 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t181 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t182 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t183 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t184 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t185 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t186 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t187 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t188 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t189 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t190 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t191 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t192 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t193 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t194 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t195 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t196 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t197 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t198 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t199 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t200 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t201 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t202 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t203 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t204 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t205 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t206 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t207 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t208 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t209 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t210 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t211 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t212 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t213 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t214 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t215 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t216 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t217 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t218 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t219 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t220 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t221 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t222 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t223 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t224 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t225 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t226 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t227 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t228 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t229 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t230 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t231 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t232 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t233 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t234 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t235 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t236 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t237 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t238 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t239 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t240 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t241 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t242 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t243 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t244 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t245 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t246 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t247 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t248 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t249 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t250 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t251 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t252 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t253 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t254 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t255 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t256 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t257 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t258 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t259 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t260 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t261 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t262 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t263 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t264 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t265 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t266 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t267 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t268 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t269 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t270 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t271 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t272 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t273 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t274 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t275 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t276 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t277 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t278 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t279 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t280 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t281 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t282 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t283 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t284 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t285 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t286 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t287 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t288 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t289 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t290 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t291 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t292 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t293 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t294 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t295 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t296 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t297 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t298 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t299 TABLESPACE roll;
CREATE ROLLBACK SEGMENT t300 TABLESPACE roll;
```

tpcc_stored_proc.sh

```
#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
#============================================================+
#       Copyright (c) 1998  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#============================================================+
# NAME
#     benchsetup
# DESCRIPTION
#     Usage: benchsetup.sh [options]
#=============================================================
#

BUILD_HOME=$ORACLE_HOME/bench/tpc/tpcc/scripts/1450w4k
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BUILD_HOME/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BUILD_HOME/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_BLOCKS=$BENCH_HOME/tpcc/blocks
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$BUILD_HOME/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
```

```
BUILD_SQL=sql
LOAD_SCRIPTS=${BUILD_HOME}
OUTDIR=${BUILD_HOME}/outdir
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
MULT=1450

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH


sqlplus tpcc/tpcc @$TPCC_BLOCKS/views
sqlplus tpcc/tpcc @$TPCC_BLOCKS/initpay
sqlplus tpcc/tpcc @$TPCC_BLOCKS/tkvcinin
# sqlplus tpcc/tpcc @$TPCC_BLOCKS/pay
```

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

cust.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 =============================================================
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)
      )
                  single                          table
      hashkeys       43500000
      hash is        (c_w_id * 30000 + c_d_id * 3000 + c_id)
      size           850
      initrans       3
      pctfree        0
      tablespace     cust
                  storage (buffer_pool recycle initial 770M next 770M minextents 58);

      create table customer (
      c_id          number(5,0),
      c_d_id        number(2,0),
      c_w_id        number(5,0),
      c_discount    number,
      c_credit      char(2),
      c_last        varchar2(16),
      c_first       varchar2(16),
      c_credit_lim  number,
      c_balance     number,
      c_ytd_payment number,
      c_payment_cnt number,
      c_delivery_cnt number,
      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_middle      char(2),
      c_data        varchar2(500)
      )
      cluster ccluster (c_id, c_d_id, c_w_id);

rem
rem done
rem

      exit;
```

dist.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      dist.sql
rem  DESCRIPTION
rem      Create customer table for TPC-C database.
rem =============================================================
rem

rem
rem DROP all first
rem
      drop table district;
                  drop cluster dcluster including tables;

set timing on

rem
rem DISTRICT table
rem

                  create cluster dcluster (
                              d_w_id        number(5,0),
                              d_id          number(2,0)
                  )
                  single                    table
                  hashkeys      14500
                  hash is                   (d_w_id) * 10 + d_id
                  size                      1536
                  initrans      3
                  pctfree                   0
      tablespace                misc;

      create table district (
                  d_id          number(2,0),
                  d_w_id        number(5,0),
                  d_ytd         number,
                  d_tax         number,
                  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)
                  )
                  cluster dcluster (d_w_id, d_id);

rem
rem done
rem

      exit;
```

hist.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      hist.sql
rem  DESCRIPTION
rem      Create customer table for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
      drop table history;

set timing on

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,
      h_data          varchar2(24)
      )
      tablespace hist
      initrans 4
      pctfree  5
      storage (freelist groups 40 freelists 9
                                    buffer_pool recycle);

rem
rem done
rem

      exit;
```

icust.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      icust.sql
rem  DESCRIPTION
rem      Create customer index for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
      drop index icustomer;

set timing on

rem
rem ICUST1 index
rem

     create unique index icustomer on customer(c_w_id, c_d_id, c_id)
      tablespace unused
      initrans 3
                parallel 2
     storage (initial 20M next 20M  pctincrease 0) pctfree 1;

rem
rem done
rem

      exit;
```

icust2.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      icust2.sql
rem  DESCRIPTION
rem      Create customer index 2 for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
      drop index icustomer2;

set timing on

rem
rem ICUST2 index
rem

     create unique index icustomer2 on customer(c_last, c_d_id, c_w_id, c_first)
      tablespace icust2
      initrans 3
                parallel 10
      pctfree  1;

rem
rem done
rem

      exit;
```

idist.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      dist.sql
rem  DESCRIPTION
rem      Create customer table for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
      drop index idistrict;

set timing on

rem
rem DISTRICT index
rem

     create unique index idistrict on district(d_w_id, d_id)
      tablespace unused
      initrans 3
      pctfree  5;

rem
rem done
rem

      exit;
```

iitem.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      item.sql
rem  DESCRIPTION
rem      Create customer table for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
      drop index iitem;

set timing on

rem
rem ITEM index
```

```
rem

      create unique index iitem on item(i_id)
       tablespace misc
       initrans 3
                  parallel 10
       pctfree  1;

rem
rem done
rem

       exit;
```

iordr.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       iordr.sql
rem  DESCRIPTION
rem       Create orders index for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
       drop index iorders;

set timing on

rem
rem ORDERS index
rem

      create unique index iorders on orders(o_w_id, o_d_id, o_id)
       tablespace iord1
       initrans 3
                  parallel 10;

rem
rem done
rem

       exit;
```

iordr2.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       iordr2.sql
rem  DESCRIPTION
rem       Create orders index 2 for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
       drop index iorders2;

set timing on

rem
rem ORDERS index 2
rem

      create unique index iorders2 on orders(o_w_id, o_d_id, o_c_id, o_id)
       tablespace iord2
       initrans 4
                  parallel 10
       storage (initial 6M next 6M  pctincrease 0
                                    maxextents unlimited freelist groups 13 freelists 22)
       pctfree  1;

rem
rem done
rem

       exit;
```

istok.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       istok.sql
rem  DESCRIPTION
rem       Create stock index for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
       drop index istock;

set timing on

rem
rem STOCK index
rem

      create unique index istock on stock(s_i_id, s_w_id)
       tablespace unused
       initrans 3
                  parallel 10;

rem
rem done
rem

       exit;
```

item.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       item.sql
rem  DESCRIPTION
rem       Create ITEM table for TPC-C database.
rem ================================================================
rem

rem
rem DROP item cluster and table
rem
       drop cluster icluster including tables;
       drop table item;

       set timing on;

rem
rem ITEM table
rem

       create cluster icluster (
       i_id           number (6,0)
       )
                  single                              table
       hashkeys       100000
       hash is        i_id
       size           120
       initrans       3
       pctfree        0
       tablespace     misc
                  storage (buffer_pool keep);

       create table item (
       i_id           number(6,0),
       i_name         varchar2(24),
       i_price        number,
       i_data         varchar2(50),
       i_im_id        number
       )
       cluster icluster(i_id);


rem
rem done
rem

       exit;
```

iware.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      iware.sql
rem  DESCRIPTION
rem      Create warehouse index for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
      drop index iwarehouse;

set timing on

rem
rem WAREHOUSE index
rem

    create unique index iwarehouse on warehouse (w_id)
      tablespace unused
      initrans 3
                pctfree 1;

rem
rem done
rem

      exit;
```

nord.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      nord.sql
rem  DESCRIPTION
rem      Create customer table for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
      drop table new_order;

set timing on

rem
rem NEW_ORDER table
rem

     create table new_order (
           no_w_id        number,
           no_d_id        number,
           no_o_id        number,
                          constraint inord primary key (no_w_id, no_d_id, no_o_id)
      )
                organization index tablespace nord
      initrans 4
      pctfree  5
                storage ( freelist groups 40 freelists 9);


rem
rem done
rem

      exit;
```

ordl.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      ordr.sql
rem  DESCRIPTION
rem      Create customer table for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
      drop table order_line;

set timing on

rem
rem ORDER_LINE table
rem

     create table order_line (
      ol_w_id         number,
      ol_d_id         number,
      ol_o_id         number,
      ol_number       number,
      ol_i_id         number,
      ol_delivery_d   date,
      ol_amount       number,
      ol_supply_w_id  number,
      ol_quantity     number,
      ol_dist_info    char(24),
                constraint iordl primary key (ol_w_id, ol_d_id, ol_o_id, ol_number)
      )
                organization index tablespace ordl
      initrans 4
      pctfree  5
      storage ( freelist groups 40 freelists 9);

rem
rem done
rem

      exit;
```

ordr.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      ordr.sql
rem  DESCRIPTION
rem      Create customer table for TPC-C database.
rem ================================================================
rem

rem
rem DROP all first
rem
      drop table orders;

set timing on

rem
rem ORDERS table
rem

     create table orders (
      o_id           number,
      o_w_id         number,
      o_d_id         number,
      o_c_id         number,
      o_carrier_id   number,
      o_ol_cnt       number,
      o_all_local    number,
      o_entry_d      date
      )
      tablespace ordr
      initrans 3
      pctfree  5
      storage (initial 40K next 50M  pctincrease 0
            freelist groups 40 freelists 9);


rem
rem done
rem

      exit;
```

stok.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 ================================================================
```

```
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)
      )
                  single                              table
      hashkeys       145000000
      hash is        (s_i_id * 1450 + s_w_id)
      size           350
      initrans       3
      pctfree        0
      tablespace     stock
                  storage (buffer_pool keep initial 1810M next 1810M minextents 30);

      create table stock (
      s_i_id          number(6,0),
      s_w_id          number(5,0),
      s_quantity      number,
      s_ytd           number,
      s_order_cnt     number,
      s_remote_cnt    number,
      s_data          varchar2(50),
      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)
      )
      cluster scluster (s_i_id, s_w_id);

rem
rem done
rem

      exit;
```

ware.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       ware.sql
rem  DESCRIPTION
rem       Create customer table for TPC-C database.
rem =================================================================
rem

rem
rem DROP all first
rem
      drop table warehouse;
                  drop cluster wcluster including tables;

set timing on

rem
rem WAREHOUSE table
rem

                  create cluster wcluster (
                              w_id                        number (5,0)
                  )
                  single                                  table
                  hashkeys                    1450
                  hash is                                 w_id
                  size                                    1536
                  initrans                    3
                  pctfree                                 0
      tablespace              misc;

      create table warehouse (
            w_id          number(5,0),
            w_ytd         number,
            w_tax         number,
            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)
      )
                  cluster wcluster (w_id);

rem
rem done
rem

      exit;
```

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;
rem analyze table new_order estimate statistics;
rem 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;
quit;
rem analyze index inew_order estimate statistics;
rem analyze index iorder_line estimate statistics;
```

C.3 Data Generation Code

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
+================================================================*/

#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 (w_id, w_ytd, w_tax, w_name, w_street_1,
w_street_2, w_city, w_state, w_zip) 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 (d_id, d_w_id, d_ytd, d_tax, d_next_o_id, d_name,
d_street_1, d_street_2, d_city, d_state, d_zip) 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 (C_ID, C_D_ID, C_W_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_YTD_PAYMENT,
C_PAYMENT_CNT, C_DELIVERY_CNT, C_DATA) 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 (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id, h_date,
h_amount, h_data) 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 (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, s_ytd, s_order_cnt,
s_remote_cnt, s_data) \
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 (I_ID,I_IM_ID,I_NAME,I_PRICE,I_DATA) VALUES
(:i_id, :i_im_id, :i_name, :i_price, \
  :i_data)"

#define SQLTXTO1 "INSERT INTO orders (O_ID,
O_D_ID,O_W_ID,O_C_ID,O_ENTRY_D,O_CARRIER_ID,O_OL_CNT,O_ALL_LOCAL) \
  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 (O_ID,
O_D_ID,O_W_ID,O_C_ID,O_ENTRY_D,O_CARRIER_ID,O_OL_CNT,O_ALL_LOCAL) \
  VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
  SYSDATE, 11, :o_ol_cnt, 1)"

#define SQLTXTOL1 "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, SYSDATE, :ol_i_id, :ol_supply_w_id, 5, 0, \
  :ol_dist_info)"

#define SQLTXTOL2 "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, to_date('01-Jan-1811'), :ol_i_id, :ol_supply_w_id, 5, :ol_amount, \
  :ol_dist_info)"

#define SQLTXTNO "INSERT INTO new_order (no_o_id, no_d_id, no_w_id) 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];
  float 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];
  float 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];
  float 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];
#define FIRSTNAME_WITH_CLAST
#ifdef FIRSTNAME_WITH_CLAST
  char firstname_with_clast[100];
  sprintf(firstname_with_clast, "C_LAST=%d", CNUM1);
#endif /* FIRSTNAME_WITH_CLAST */


/*---------------------------------------------------------------+
 | 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_FLT, -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_FLT, -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_FLT, -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 (SEED);
  srand48 (SEED);
  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';

      if (gen) {
        printf ("%d 30000000 %6.4f %s %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 %6.4f 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);
#ifdef FIRSTNAME_WITH_CLAST
      if ((c_id[i] == 1) && (c_d_id[i] == 1) && (c_w_id[i] == 1 ))
        strcpy(c_first[i], firstname_with_clast);
      else
#endif

      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 %cC 5000000 %6.4f -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 %s 0 0 0 %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 0 0 0 %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 1000 %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 */
            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 5 %ld %s\n", cid,
                    cdid,  cwid, j + 1, sdate, ol_i_id[j], cwid,
                    ol_amount[j], str24[j]);
            }
            else {
                      /* Insert a default date instead of null date */
              fprintf (olfp, "%d %d %d %d 01-Jan-1811 %d %d 5 %ld %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);

}
```

C4. Loader

Dist.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 district
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY '"'
(

 D_ID                           integer external,
 D_W_ID                          integer external,
 D_YTD                          integer external,
 D_TAX                          float external,
 D_NEXT_O_ID                          integer external,
 D_NAME                        CHAR(10),
 D_STREET_1                       CHAR(20),
 D_STREET_2                       CHAR(20),
 D_CITY                       CHAR(20),
 D_STATE                          CHAR(2),
 D_ZIP                         CHAR(9)
)
```

Hist.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.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
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 "DD-Mon-YYYY",
  h_amount         integer external,
  h_data           char(24)
)
```

Neword.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
--      neword.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>
-- ============================================================*/

OPTIONS (DIRECT = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE new_order
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY '"'
(
  no_o_id          integer external,
  no_d_id          integer external,
  no_w_id          integer external
)
```

Order.ctl

```
--
-- $Header: order.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.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>
-- ============================================================*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE orders
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 "DD-Mon-YYYY",
  o_carrier_id     integer external,
  o_ol_cnt         integer external,
  o_all_local      integer external
)
```

Ordline.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.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>
-- ============================================================*/

OPTIONS (DIRECT = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE order_line
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)
)
```

Stock.ctl

```
--
-- $Header: stock.ctl 7030100.1 95/08/07 15:54:18 plai Osd<base> $ Copyr (c) 1994 Oracle
--
-- ============================================================+
--       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
--                OPEN SYSTEMS PERFORMANCE GROUP                  |
--                   All Rights Reserved                          |
-- ============================================================+
-- FILENAME
--      stock.ctl
-- DESCRIPTION
--      This is a SQL*Loader control file.  It is used for
--      loading stocks 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 stock
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY '"'
(
  s_i_id           integer external,
  s_w_id           integer external,
  s_quantity       integer external,
  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            integer external,
  s_order_cnt      integer external,
  s_remote_cnt     integer external,
  s_data           char(50)
)
```

Ware.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 warehouse
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY '"'
(

 W_ID                           integer external,
 W_YTD                            integer external,
 W_TAX                            float external,
 W_NAME                             CHAR(10)
 W_STREET_1                           CHAR(20)
 W_STREET_2                           CHAR(20)
 W_CITY                           CHAR(20)
 W_STATE                            CHAR(2)
 W_ZIP                          CHAR(9)
)
```

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 */
LASTC=86
MEASUREMENT="1"
MASTER "lmaster"
SUT   = "suta"

MAIN_CLIENT   = "client1"
#if 1
SLAVES drv1,drv2,drv3,drv4,drv5,drv6,drv7,drv8,drv9,drv10,drv11,drv12,drv13,drv14,drv15,drv16
CLIENT_REAL   = "client1 client2 client3 client4 client5 client6 client7 client8"
#endif
#if 0
SLAVES
drv1,drv1a,drv1b,drv1c,drv2,drv2a,drv2b,drv2c,drv3,drv3a,drv3b,drv3c,drv4,drv4a,drv4b,drv4c,drv
7,drv7a,drv7b,drv7c,drv8,drv8a,drv8b,drv8c,drv9,drv9a,drv9b,drv9c,drv10,drv10a,drv10b,drv10c,dr
v11,drv11a,drv11b,drv11c,drv12,drv12a,drv12b,drv12c,drv13,drv13a,drv13b,drv13c,drv14,drv14a,dr
v14b,drv14c,drv15,drv15a,drv15b,drv15c,drv16,drv16a,drv16b,drv16c
CLIENT_REAL   = "client1 client2 client4 client5 client6 client7 client8"
#endif
#if 0
SLAVES drv1,drv2,drv3,drv4,drv5,drv6,drv7,drv8,drv9,drv10,drv11,drv12,drv13,drv14,drv15,drv16
CLIENT_REAL   = "client1 client2 client3 client4 client5 client6 client7 client8 "
#endif

CLIENT client1x oracle orif1db
CLIENT client1y oracle orif1db
CLIENT client2x oracle orif1db
CLIENT client2y oracle orif1db
CLIENT client3x oracle orif1db
CLIENT client3y oracle orif1db
CLIENT client4x oracle orif1db
CLIENT client4y oracle orif1db
CLIENT client5x oracle orif1db
CLIENT client5y oracle orif1db
CLIENT client6x oracle orif1db
CLIENT client6y oracle orif1db
CLIENT client7x oracle orif1db
CLIENT client7y oracle orif1db
CLIENT client8x oracle orif1db
CLIENT client8y oracle orif1db

TELNET telnet 23
SOCKET socket 199703

SOCKET_NETWORK socket1  6700 drv3
SOCKET_NETWORK socket2  6701 drv4
SOCKET_NETWORK socket3  6702 drv3
SOCKET_NETWORK socket4  6703 drv4
SOCKET_NETWORK socket5  6704 drv3
SOCKET_NETWORK socket6  6705 drv4
SOCKET_NETWORK socket7  6706 drv3
SOCKET_NETWORK socket8  6707 drv4
#if 0
SOCKET_NETWORK socket9  6708 drv3
SOCKET_NETWORK socket10 6709 drv4
#endif

SOCKET_NETWORK socket11  6700 drv5
SOCKET_NETWORK socket12  6701 drv6
SOCKET_NETWORK socket13  6702 drv5
SOCKET_NETWORK socket14  6703 drv6
SOCKET_NETWORK socket15  6704 drv5
SOCKET_NETWORK socket16  6705 drv6
SOCKET_NETWORK socket17  6706 drv5
SOCKET_NETWORK socket18  6707 drv6
#if 0
SOCKET_NETWORK socket19  6708 drv5
SOCKET_NETWORK socket20  6709 drv6
#endif

SOCKET_NETWORK socket21  6700 drv7
SOCKET_NETWORK socket22  6701 drv8
SOCKET_NETWORK socket23  6702 drv7
SOCKET_NETWORK socket24  6703 drv8
SOCKET_NETWORK socket25  6704 drv7
SOCKET_NETWORK socket26  6705 drv8
SOCKET_NETWORK socket27  6706 drv7
SOCKET_NETWORK socket28  6707 drv8
#if 0
SOCKET_NETWORK socket29  6708 drv7
SOCKET_NETWORK socket30  6709 drv8
#endif

SOCKET_NETWORK socket31  6700 drv9
SOCKET_NETWORK socket32  6701 drv10
SOCKET_NETWORK socket33  6702 drv9
SOCKET_NETWORK socket34  6703 drv10
SOCKET_NETWORK socket35  6704 drv9
SOCKET_NETWORK socket36  6705 drv10
SOCKET_NETWORK socket37  6706 drv9
SOCKET_NETWORK socket38  6707 drv10
#if 0
SOCKET_NETWORK socket39  6708 drv9
SOCKET_NETWORK socket40  6709 drv10
#endif

SOCKET_NETWORK socket41  6700 drv11
SOCKET_NETWORK socket42  6701 drv12
SOCKET_NETWORK socket43  6702 drv11
SOCKET_NETWORK socket44  6703 drv12
SOCKET_NETWORK socket45  6704 drv11
SOCKET_NETWORK socket46  6705 drv12
SOCKET_NETWORK socket47  6706 drv11
SOCKET_NETWORK socket48  6707 drv12
#if 0
SOCKET_NETWORK socket49  6708 drv11
SOCKET_NETWORK socket50  6709 drv12
#endif

SOCKET_NETWORK socket51  6700 drv13
SOCKET_NETWORK socket52  6701 drv14
SOCKET_NETWORK socket53  6702 drv13
SOCKET_NETWORK socket54  6703 drv14
SOCKET_NETWORK socket55  6704 drv13
SOCKET_NETWORK socket56  6705 drv14
SOCKET_NETWORK socket57  6706 drv13
SOCKET_NETWORK socket58  6707 drv14
#if 0
SOCKET_NETWORK socket59  6708 drv13
SOCKET_NETWORK socket60  6709 drv14
#endif

SOCKET_NETWORK socket61  6700 drv15
SOCKET_NETWORK socket62  6701 drv16
SOCKET_NETWORK socket63  6702 drv15
SOCKET_NETWORK socket64  6703 drv16
SOCKET_NETWORK socket65  6704 drv15
SOCKET_NETWORK socket66  6705 drv16
SOCKET_NETWORK socket67  6706 drv15
SOCKET_NETWORK socket68  6707 drv16
#if 0
SOCKET_NETWORK socket69  6708 drv15
SOCKET_NETWORK socket70  6709 drv16
#endif

SOCKET_NETWORK socket71  6700 drv1
```

```
SOCKET_NETWORK socket72  6701 drv2
SOCKET_NETWORK socket73  6702 drv1
SOCKET_NETWORK socket74  6703 drv2
SOCKET_NETWORK socket75  6704 drv1
SOCKET_NETWORK socket76  6705 drv2
SOCKET_NETWORK socket77  6706 drv1
SOCKET_NETWORK socket78  6707 drv2
#if 0
SOCKET_NETWORK socket79  6708 drv1
SOCKET_NETWORK socket80  6709 drv2
#endif


OUTPUTNAME="../runs/creature"

CPU=4

#if 1
BEGIN_WAIT=15:00
RAMPUP=42:30
RUNTIME=30:00
RAMPDOWN_WAIT=2:00
RAMPDOWN=10:00
#else
BEGIN_WAIT=10:00
RAMPUP=15:00
RUNTIME=15:00
RAMPDOWN_WAIT=5:00
RAMPDOWN=10:00
#endif
INTERVAL=1:00   /* Interval to calculate mix from */

LOGIN_MAX_LOAD       = 4
LOGIN_BEGIN          = 0  /* skip login state if set to 1 */
NOBEGIN              = 0
KEYSTROKE_PACKET_SIZE = 0

MAX_CONCURRENT_SPAWN  = 10
SPAWN_COUNT          = 5

MIN_PORT             = 8088
MAX_PORT             = 8089

/* User variables.  Think, Emulex Delay, %desired, %min, %max */
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 "

START client1x socket71   220
START client1y socket72   220
START client1x socket73   220
START client1y socket74   220
START client1x socket75   220
START client1y socket76   220
START client1x socket77   220
START client1y socket78   220
#if 0
START client1x socket79   225
START client1y socket80   225
#endif

START client2x socket1    220
START client2y socket2    220
START client2x socket3    220
START client2y socket4    220
START client2x socket5    220
START client2y socket6    220
START client2x socket7    220
START client2y socket8    220
#if 0
START client2x socket9    225
START client2y socket10   225
#endif

START client3x socket11   220
START client3y socket12   220
START client3x socket13   220
START client3y socket14   220
START client3x socket15   220
START client3y socket16   220
START client3x socket17   220
START client3y socket18   220
#if 0
START client3x socket19   225
START client3y socket20   225
#endif

START client4x socket21   220
START client4y socket22   220
START client4x socket23   220
START client4y socket24   220
START client4x socket25   220
START client4y socket26   220
START client4x socket27   220
START client4y socket28   220
#if 0
START client4x socket29   225
START client4y socket30   225
#endif

START client5x socket31   220
START client5y socket32   220
START client5x socket33   220
START client5y socket34   220
START client5x socket35   220
START client5y socket36   220
START client5x socket37   220
START client5y socket38   220
#if 0
START client5x socket39   225
START client5y socket40   225
#endif

START client6x socket41   220
START client6y socket42   220
START client6x socket43   220
START client6y socket44   220
START client6x socket45   220
START client6y socket46   220
START client6x socket47   220
START client6y socket48   220
#if 0
START client6x socket49   225
START client6y socket50   225
#endif

START client7x socket51   220
START client7y socket52   220
START client7x socket53   220
START client7y socket54   220
START client7x socket55   220
START client7y socket56   220
START client7x socket57   220
START client7y socket58   220
#if 0
START client7x socket59   225
START client7y socket60   225
#endif

START client8x socket61   220
START client8y socket62   220
START client8x socket63   220
START client8y socket64   220
START client8x socket65   220
START client8y socket66   220
START client8x socket67   220
START client8y socket68   220
#if 0
START client8x socket69   225
START client8y socket70   225
#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 1.8 1996/11/27 19:58:49 channui Exp channui $";

#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 <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#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 1.9 1996/11/27 19:53:38 channui Exp channui $";

/*****************************************************************************/
/***              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 %s screen\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 1.6 1996/11/27 19:53:51 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;

#endif
```

Appendix E: Third Party Quotes

May 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	1,950	Compex 8 port10baset hub	32	62,400

Please note, prices are valid for 60 days (subject to availability). Thank you for your consideration.

Sincerely,

Claudia Segreto
Datacomm Warehouse
800-328-2261 ext20286
732-363-4823 FAX

Quotation No. 58
Terms: Net 30 days
Quote Duration: Sixty (60) days
FOB: FACTORY
Date: 5/21/1999

To:
D Pruitt
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
NH2012/R	8-port 10/100 Auto-Sensing Ethernet Switch with 2 Expansion/Uplink Slots	$3,095.00	2	30%	$4,333.00
NH2032	Powergroup Switch with 16 10/100Base-TX ports with 4 expansion slots.	$2,000.00	2	30%	$2,800.00
				Total Configuration Cost	**$7,133.00**
		Note: If we are favored with your order it should be made out to: NBase-Xyplex 3200 Southwest Frwy Suite 3300 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