TPC Benchmark™ C
Full Disclosure Report

BULL ESCALA
RL470

using

ORACLE 8

Special Notices

The following terms used in this publication are trademarks of the BULL company in the United States and/or other countries:

BULL
Escala

The following terms used in this publication are trademarks of the IBM company in the United States and/or other countries:

AIX
IBM
RISC System/6000
TX series, Encina

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

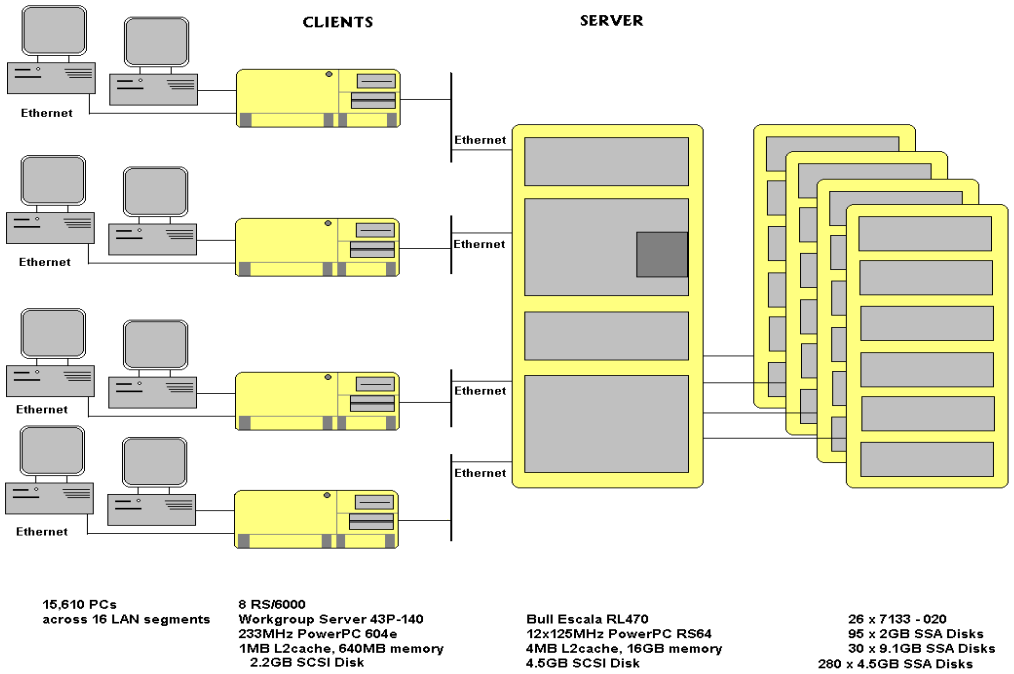
TPC Benchmark Trademark of the Transaction Processing Performance Council
ORACLE, Oracle 8, SQL*DBA, SQL*Loader, SQL*Net and SQL*Plus are registered trademarks of Oracle Corporation.

First Edition March 25, 1998

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

BULL S.A.
J.F. Lemerre
B.P.208 - 1 rue de Provence - 38 432 Echirolles - France
FAX Number 33 047 629 75 18

BULL S.A.	ESCALA RL470 C/S			TPC - C Rev.3.3 Report Date March 25, 98
Total System	TPC - C Throughput		Price / Performance	Availability
$ 2,177,811	**18,666.73 tpmC**		**$116.66/ tpmC**	**Now except: Oracle 8 : sep 2, 98**
Processors	Data base Manager	Operating System	Other Software	Number of users
12 Power PC RS64 @125MHz	Oracle8 Enterprise Edition 8.0	AIX4.3.0	IBM TX Series 4.2 for AIX	15,610



System Components	Clients		Server	
	Qty	Type	Qty	Type
Processors	8	PPC 604e @ 233 MHz	12	Power PC RS64 @ 125 MHz
Cache		w/ 1 MB L2 cache		w/ 4 MB L2 cache
Memory (MB)		640 MB each		16 GB
Disk Controllers	8	SCSI-2 Adapters	3	SCSI -2 Adapters
			5	SSA Adapters
Disk Drives	8	2.2 GB SCSI Disks	376	4.5GB SSA Disks
			30	9.1 SCSI F/W Disks
Total Storage (GB)		2.2GB each Client		1,598.57 GB
Terminals	3	System Console	16	System Consoles
Terminal Connect	863	Linksys 20 Port Ethernet Hubs + 10% spares		

BULL S.A.	Bull Escala RL470 C/S				TPC-C REV 3.3.2 Report Date: March 25,1998		
Description	**Part Number**	**Third Party**		**Unit Price**	**Quantity**	**Extended Price**	**5-year Maint. Price**
Server Hardware		**Brand**	**Pricing**				
RL470 Package incl.:12 x RS64 Proc., 16GB mem.,							
AIX 2-user license, 9GB disk	CPXG213-0000	BULL	1	372847	1	372 847	40 416
Async/Terminal Cable	CBLG104-1600	BULL	1	100	1	100	0
Ethernet PCI Adapter	DCCG122-0000	BULL	1	275	9	2 475	864
Remote I/O cable - Drawer to drawer	CBLG182-1000	BULL	1	595	1	595	0
Drawer to drawer power control cable	CBLG184-1000	BULL	1	60	1	60	0
SCSI2 F/W PCI Adapter	MSCG031-0000	BULL	1	360	1	360	96
SSA RAID PCI Adapter	MSCG036-0000	BULL	1	3000	5	15 000	4 800
Rack I/O Drawer	DRWG007-0000	BULL	1	14900	1	14 900	4 656
System Rack Model R00	RCKG004-0000	BULL	1	3500	4	14 000	4 416
SSA Subsystem including 16 x 4.5 GB disk	SSAG005-0000	BULL	1	28067	23	645 541	266 064
SSA Subsystem including 10 x 9.1 GB disk	SSAG006-0000	BULL	1	27689	3	83 067	34 344
4.5 GB Disk Drive Modules	MSUG068-0000	BULL	1	2100	7	14 700	8 064
SSA Cables	CBLG164-1000	BULL	1	35	26	910	0
					Subtotal	1 164 555	363 720
Server Software							
AIX C Compiler	CLGG017-RAAA	BULL	1	575	1	575	288
Performance Toolbox	UTSG136-PA00	BULL	1	950	1	950	192
AIX 4.3 Server Library	EXSG150-SRCE	BULL	1	100	1	100	0
AIX 4.3 Bonus Pack	EXSG083-RA00	BULL	1	50	1	50	0
Oracle Version 8.0.4		Oracle	3	222721	1	222 721	222 721
					Subtotal	224 396	223 201
Client Hardware							
RS/6000 Model 43P-140, 233 MHz	7043-140	IBM	2	8000	8	64 000	26 880
2.1gb Disk, Intg SCSI-2 FW, Intg Enet							
128MB DIMM Memory	4106	IBM	2	1280	8	10 240	0
128MB DIMM Memory Expansion	4115	IBM	2	2560	32	81 920	0
Async Terminal/Printer Cable	2934	IBM	2	45	8	360	0
Ethernet Adapter, PCI	2985	IBM	2	195	8	1 560	0
3Com Fast Ethernet XL	2986	IBM	2	275	8	2 200	0
SYSTEM Console BQ306	CSUG002-2100	BULL	1	620	9	5 580	432
					Subtotal	165 860	27 312
Client Software							
AIX 4.2.1 Unlimited Users	5756-C34	IBM	2	3660	8	29 280	0
IBM TXSeries 4.2 for AIX with 25% Discount	5697-D17	IBM	2	3034	8	18 206	34 355
					Subtotal	47 486	34 355
User Connectivity							
LInksys 20-prt 10Mbps HUB (+10% spares)	EW20HUB	Linksys	4	119	863	102 697	0
					Subtotal	102 697	0
			15%Discount on BULL HW/SW			-175 772	
					TOTAL	1 529 223	648 588

Notes:

Pricing: 1-Bull 2-IBM/Dickens 3-Oracle 4-CDW

Audited by Francois Raab of Information Paradigm, Inc

Five-Year Cost of Ownership: $2 177 811

tpmC Rating: 18666,73

$ / tpmC: 116,66

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 reflects 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 BULL ESCALA RL470

MQTH, Maximum Sustained Qualified Throughput: 18,666.73 tpmC

Repeatability Run: 18,580.40 tpmC 0.5% difference

Response Times (in seconds)	90th Percentile	Average
New Order	3.57	1.56
Payment	3.23	1.31
Order-Status	3.24	1.31
Delivery (interactive)	0.2	0.12
Delivery (deferred)	4.99	2.02
Stock-Level	3.04	1.16
Menu	0.02	0.01

Transaction Mix, in percent of total transactions	Percent
New Order	44.69%
Payment	43.17%
Order-Status	4.06%
Delivery	4.05%
Stock-Level	4.00%

Keying/Think Times (in seconds)	Min.	Average	Max.
New Order	18.00/0.01	18.01/12.09	18.11/121.01
Payment	3.00/0.01	3.01/12.12	3.12/121.02
Order-Status	2.00/0.01	2.01/10.11	2.10/101.00
Delivery	2.00/0.01	2.01/5.12	2.01/51.01
Stock-Level	2.00/0.01	2.01/5.12	2.10/51.00

Test Duration	
Ramp-up Time	42 min 45 sec
Measurement interval	30 minutes
Transactions during measurement interval (all types)	1,253,044
Ramp-down time	10 minutes

Checkpointing

Number of checkpoints	1
Checkpoint Interval	30

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 3.3.2 on June 25, 1997

This is the full disclosure report for benchmark testing of the BULL ESCALA RL470 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 defered 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 benchmarking 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 Escala application code for the five TPC Benchmark™ C transactions.

1.2 Benchmark Sponsor

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

This benchmark was sponsored by **BULL S.A.** 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.*

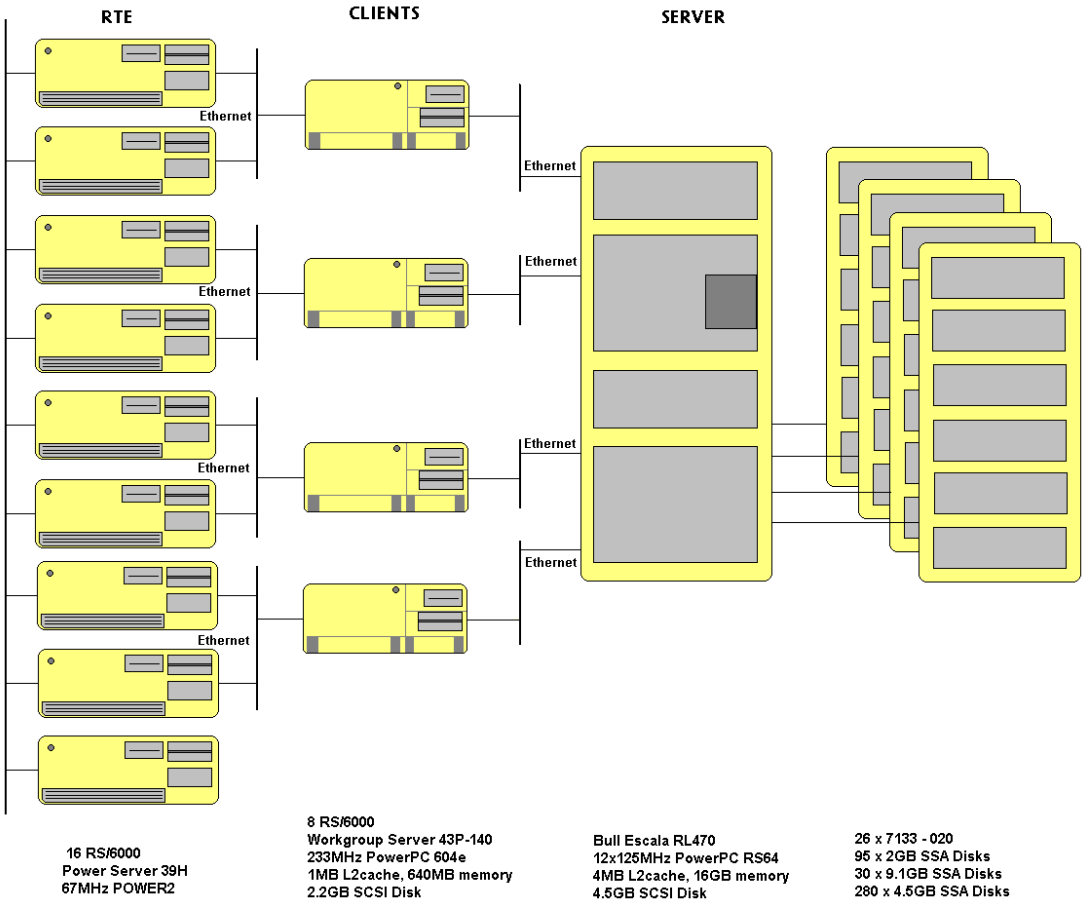
Appendix B contains the data base, application and system parameters changed from their default values used in these TPC Benchmark™ C tests.

1.4 Configuration Diagrams

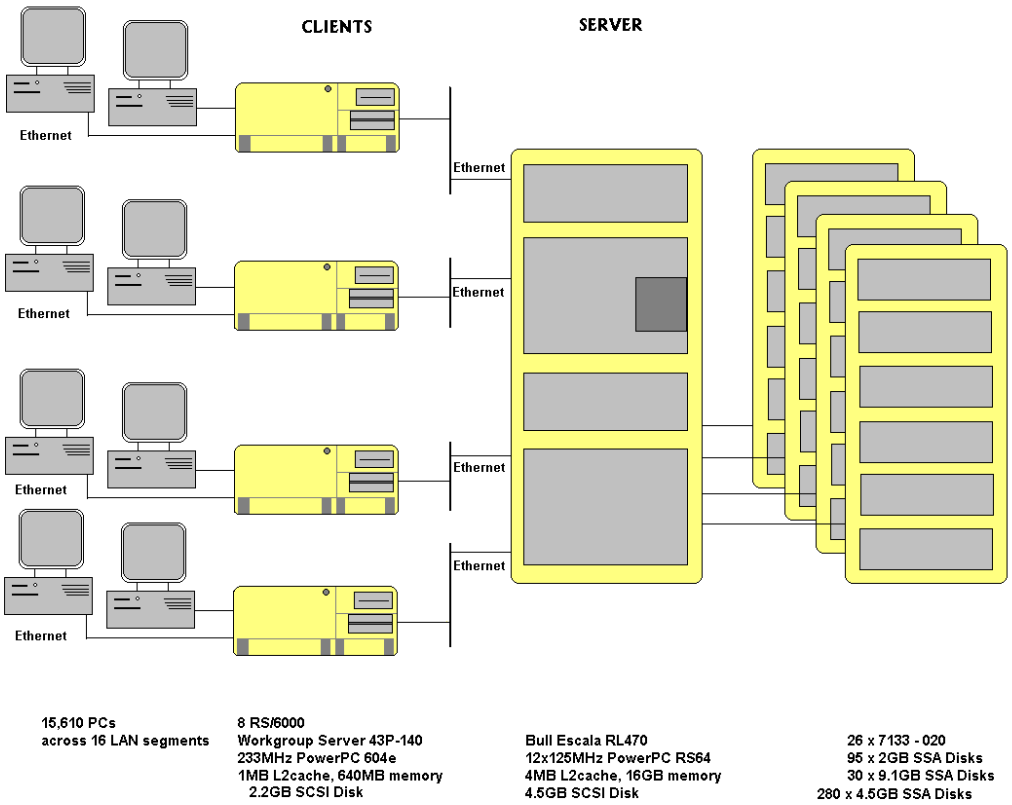
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *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.)*

BULL ESCALA RL 470 Benchmark Configuration



BULL ESCALA RL470 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 creation, index creation, and the data generation programs to load 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 on the server disks according to the details provided in Table 5-2 and in section C1 of appendix C. The size of the space segments on each disk was calculated to provide even distribution of data across the disk subsystem. Clustered indicies were defined on all tables except the history table. In addtion, 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.

The size of the space segments on each disk was calculated to provide even distribution of data across the disk subsystem.

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 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 measurements 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 smallint, D_ID smallint, C_LAST char(16), CNTR smallint);
- 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 correspond exactly to the layouts 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 priced terminals provide all the features described in Clause 2.2.2.4 must be disclosed.

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

3.4 Presentation Managers

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

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

3.5 Transactions Statistics

Table 3-1 shows the numerical quantities that clauses 8.1.3.5 to 8.1.3.11 require.

3.12 Queuing Mechanism of Delivery

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

The Delivery transaction was submitted using a RPC call to an IBM TX series version 4.2 Encina interface transaction Manager (TM). TX series returns 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 ree the application code in Appendix A for details.

Table 3-1 Transaction Statistics - ESCALA RL 470

New Order	Escala RL470
Percentage of Home order lines	99.01%
Percentage of Remote order lines	0.99%
Percentage of Rolled Back Transactions	1.01%
Number of Items per order	10
Payment	
Percentage of Home transactions	85.04%
Percentage of Remote transactions	14.96%
Non-Primary Key Access	
Percentage of Payment using C_LAST	59.90%
Percentage of Order-Status using C_LAST	60.22%
Delivery	
Delivery transactions skipped	0
Transaction Mix	
New-Order	44.69%
Payment	43.17%
Order-Status	4.06%
Delivery	4.05%
Stock-Level	4.00%

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. The Atomicity, Consistency, Isolation and all Durability tests were performed on the ESCALA RL470.

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 rooback 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.

4.2.1 Consistency Condition 1

Entries in the WAREHOUSE and DISTRICT tables 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 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 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 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.

The following steps were performed to satisfy the test of isolation for Order-Status and 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 commited.
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 commited without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 completed and was commited.
5. An order-status transaction T3 was started for the same customer used in T1. T3 returned the order inserted by T1.

This result demostrates 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 commited.
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 commited 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 commited.
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 commited.
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).

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 prio 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 commited.
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 commited.
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 commited.
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 commited.
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 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 suctomer. 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 commited 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 commited.

4.3.7 Isolation Test 9

This test demonstrates isolation for phantom protection between a delivery and a new-order tansaction.

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 commited.
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 commited 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 commited.
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 commited.

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 Instananeous 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 test was started and allowed to run for fifteen minutes.
4. The disk containing the TPCC table was powered off.
5. The run was aborted on the RTE.
6. The disk from step 5 was powered back on and the system was rebooted.
7. The failed disk was restored from the backup copy in step 2.
8. Oracle8 was restarted and its transaction log was used to roll forward the transactions that had completed but weren't written to disk before the failure.

9. Step 3 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.
10. 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 portrays the TPC Benchmark™ C defined tables and the number of rows for each table as they were built initially.

Table 5-1 Initial Cardinality of Tables (ESCALA RL470)

Table Name	Number of Rows
Warehouse	**1,561**
District	**17,000**
Customer	**51,000,000**
History	**51,000,000**
Orders	**51,000,000**
New_order	**15,300,000**
Order_line	**510,006,087**
Stock	**170,000,000**
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 diagram depict the data base configuration.

Figure 5-2 ESCALA RL470 Data Distribution Benchmark Configuration

Controller	Disk	Contents	Capacity
scsi1	hdisk0	OS, paging	4 GB
ssa0	hdisk5	lvcust1	4 GB
	hdisk6	lvcust2	4 GB
	hdisk7	lvcust3	4 GB
	hdisk8	lvcust4	4 GB
	hdisk9	lvcust5	4 GB
	hdisk10	lvcust6	4 GB
	hdisk11	lvcust7	4 GB
	hdisk12	lvcust8	4 GB
	hdisk13	lvcust9	4 GB
	hdisk14	lvcust10	4 GB
	hdisk15	lvcust11	4 GB
	hdisk16	lvcust12	4 GB
	hdisk17	lvcust13	4 GB
	hdisk18	lvcust14	4 GB
	hdisk19	lvcust15	4 GB
	hdisk20	lvcust16	4 GB

	hdisk21	lvcust17	4 GB
	hdisk22	lvcust18	4 GB
	hdisk23	lvcust19	4 GB
	hdisk24	lvcust20	4 GB
	hdisk25	lvcust21	4 GB
	hdisk26	lvcust22	4 GB
	hdisk27	lvcust23	4 GB
	hdisk28	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk29	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk30	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk31	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk32	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk33	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk34	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk35	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk36	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk37	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk38	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk39	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk40	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk41	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk42	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk43	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk44	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk45	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk46	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk47	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk48	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk49	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk50	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk51	lvstock1,lvstock2,lvstock3,lvstock4,lvstock5	4 GB
	hdisk52	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk53	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk54	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk55	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk56	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk57	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk58	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk59	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk60	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk61	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk62	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk63	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk64	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk65	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk66	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk67	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk68	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk69	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk70	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk71	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk72	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB

	hdisk73	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk74	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk75	lvstock6,lvstock7,lvstock8,lvstock9,lvstock10	4 GB
	hdisk76	lv*0,lvsys1,lvtpcc_cntl1,lvtpcc_cntl2,lvtpcc_cntl3	4 GB
	hdisk77	lvroll1	4 GB
	hdisk78	lvroll1	4 GB
	hdisk79	lvord1,lvord2,lvord3,lvord4	4 GB
	hdisk80	lvnord1	4 GB
	hdisk81	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk82	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk83	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk84	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk85	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk86	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk87	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk88	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk89	lviordl1,lviordl2,lviordl3	4 GB
	hdisk90	lviordl1,lviordl2,lviordl3	4 GB
	hdisk91	lvicust21,lvicust22,lvicust23,lvicust24	4 GB
	hdisk92	lvicust21,lvicust22,lvicust23,lvicust24	4 GB
	hdisk93	lvinord1,lvinord2,lvinord3	4 GB
	hdisk94	lvinord1,lvinord2,lvinord3	4 GB
	hdisk95	lvicust1	4 GB
	hdisk96	lvistk1	4 GB
	hdisk97	lviord21	4 GB
	hdisk98	lviord21	4 GB
	hdisk99	lviord21	4 GB
ssa1	hdisk100	lvcust24	4 GB
	hdisk101	lvcust25	4 GB
	hdisk102	lvcust26	4 GB
	hdisk103	lvcust27	4 GB
	hdisk104	lvcust28	4 GB
	hdisk105	lvcust29	4 GB
	hdisk106	lvcust30	4 GB
	hdisk107	lvcust31	4 GB
	hdisk108	lvcust32	4 GB
	hdisk109	lvcust33	4 GB
	hdisk110	lvcust34	4 GB
	hdisk111	lvcust35	4 GB
	hdisk112	lvcust36	4 GB
	hdisk113	lvcust37	4 GB
	hdisk114	lvcust38	4 GB
	hdisk115	lvcust39	4 GB
	hdisk116	lvcust40	4 GB
	hdisk117	lvcust41	4 GB
	hdisk118	lvcust42	4 GB
	hdisk119	lvcust43	4 GB
	hdisk120	lvcust44	4 GB
	hdisk121	lvcust45	4 GB
	hdisk122	lvcust46	4 GB
	hdisk123	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk124	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB

	hdisk125	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk126	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk127	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk128	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk129	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk130	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk131	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk132	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk133	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk134	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk135	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk136	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk137	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk138	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk139	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk140	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk141	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk142	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk143	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk144	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk145	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk146	lvstock11,lvstock12,lvstock13,lvstock14,lvstock15	4 GB
	hdisk147	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk148	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk149	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk150	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk151	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk152	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk153	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk154	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk155	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk156	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk157	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk158	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk159	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk160	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk161	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk162	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk163	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk164	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk165	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk166	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk167	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk168	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk169	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk170	lvstock16,lvstock17,lvstock18,lvstock19,lvstock20	4 GB
	hdisk171	lvnord1	4 GB
	hdisk172	lvware	4 GB
	hdisk173	lvord1,lvord2,lvord3,lvord4	4 GB
	hdisk174	lvnord1	4 GB
	hdisk175	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk176	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB

	hdisk177	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk178	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk179	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk180	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk181	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk182	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk183	lvordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk184	lviord11,lviord12,lviord13	4 GB
	hdisk185	lviord11,lviord12,lviord13	4 GB
	hdisk186	lvicust21,lvicust22,lvicust23,lvicust24	4 GB
	hdisk187	lvicust21,lvicust22,lvicust23,lvicust24	4 GB
	hdisk188	lvinord1,lvinord2,lvinord3	4 GB
	hdisk189	lvinord1,lvinord2,lvinord3	4 GB
	hdisk190	lvistk2	4 GB
	hdisk191	lvistk3	4 GB
	hdisk192	lviord22	4 GB
	hdisk193	lviord22	4 GB
	hdisk194	lviord22	4 GB
ssa3	hdisk195	lvcust47	2 GB
	hdisk196	lvcust48	2 GB
	hdisk197	lvcust49	2 GB
	hdisk198	lvcust50	2 GB
	hdisk199	lvcust51	2 GB
	hdisk200	lvcust52	2 GB
	hdisk201	lvcust53	2 GB
	hdisk202	lvcust54	2 GB
	hdisk203	lvcust55	2 GB
	hdisk204	lvcust56	2 GB
	hdisk205	lvcust57	2 GB
	hdisk206	lvcust58	2 GB
	hdisk207	lvcust59	2 GB
	hdisk208	lvcust60	2 GB
	hdisk209	lvcust61	2 GB
	hdisk210	lvcust62	2 GB
	hdisk211	lvcust63	2 GB
	hdisk212	lvcust64	2 GB
	hdisk213	lvcust65	2 GB
	hdisk214	lvcust66	2 GB
	hdisk215	lvcust67	2 GB
	hdisk216	lvcust68	2 GB
	hdisk217	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk218	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk219	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk220	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk221	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk222	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk223	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk224	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk225	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk226	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk227	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk228	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB

	hdisk229	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk230	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk231	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk232	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk233	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk234	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk235	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk236	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk237	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk238	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk239	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2GB
	hdisk240	lvstock21,lvstock22,lvstock23.lvstock24,lvstock25	2 GB
	hdisk241	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2GB
	hdisk242	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk243	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk244	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2GB
	hdisk245	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2GB
	hdisk246	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2GB
	hdisk247	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2GB
	hdisk248	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2GB
	hdisk249	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk250	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk251	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk252	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk253	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk254	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk255	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk256	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk257	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk258	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk259	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk260	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk261	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk262	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk263	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk264	lvstock26,lvstock27,lvstock28.lvstock29,lvstock30	2 GB
	hdisk265	lvitem	2 GB
	hdisk266	lvord1,lvord2,lvord3,lvord4	2 GB
	hdisk267	lvnord1	2 GB
	hdisk268	lvhist1	2 GB
	hdisk269	lvhist2	2 GB
	hdisk270	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	2 GB
	hdisk271	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	2 GB
	hdisk272	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	2 GB
	hdisk273	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	2 GB
	hdisk274	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	2 GB
	hdisk275	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	2 GB
	hdisk276	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	2 GB
	hdisk277	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	2 GB
	hdisk278	lviord11,lviord12,lviord13	2 GB
	hdisk279	lviord11,lviord12,lviord13	2 GB
	hdisk280	lvicust21,lvicust22,lvicust23,lvicust24	2 GB

	hdisk281	lvicust21,lvicust22,lvicust23,lvicust24	2 GB
	hdisk282	lvinord1,lvinord2,lvinord3	2 GB
	hdisk283	lvinord1,lvinord2,lvinord3	2 GB
	hdisk285	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	2 GB
	hdisk286	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	2 GB
	hdisk287	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	2 GB
	hdisk288	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	2 GB
	hdisk289	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	2 GB
	hdisk290	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
ssa5	hdisk291	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	2 GB
	hdisk292	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk293	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk294	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk295	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk296	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk297	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk298	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk299	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk300	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk301	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk302	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk303	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk304	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk305	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk306	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk307	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk308	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk309	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk310	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk311	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk312	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk313	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk314	lvstock31,lvstock32,lvstock33,lvstock34,lvstock35	4 GB
	hdisk315	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk316	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk317	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk318	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk319	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk320	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk321	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk322	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk323	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk324	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk325	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk326	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk327	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk328	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk329	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk330	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk331	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk332	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk333	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB

	hdisk334	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk335	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk336	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk337	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk338	lvstock36,lvstock37,lvstock38,lvstock39,lvstock40	4 GB
	hdisk339	lvord1,lvord2,lvord3,lvord4	4 GB
	hdisk340	lvord1,lvord2,lvord3,lvord4	4 GB
	hdisk341	lvord1,lvord2,lvord3,lvord4	4 GB
	hdisk342	lvhist3	4 GB
	hdisk343	lvhist4	4 GB
	hdisk344	lvnord1	4 GB
	hdisk345	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	4 GB
	hdisk346	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	4 GB
	hdisk347	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	4 GB
	hdisk348	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,32,32	4 GB
	hdisk349	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31.32	4 GB
	hdisk350	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	4 GB
	hdisk351	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	4 GB
	hdisk352	lvordl17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32	4 GB
	hdisk353	lvicust21,lvicust22,lvicust23,lvicust24	4 GB
	hdisk354	lvicust21,lvicust22,lvicust23,lvicust24	4 GB
	hdisk360	lviord23	4 GB
	hdisk361	lviord23	4 GB
	hdisk362	lviord23	4 GB
	hdisk363	lviord23	4 GB
	hdisk364	lviord24	4 GB
	hdisk365	lviord24	4 GB
	hdisk366	lviord24	4 GB
	hdisk367	lviord24	4 GB
	hdisk368	lviord25	4 GB
	hdisk369	lviord25	4 GB
	hdisk370	lviord25	4 GB
	hdisk371	lviord25	4 GB
	hdisk372	lviord26	4 GB
	hdisk373	lviord26	4 GB
	hdisk374	lviord26	4 GB
	hdisk375	lviord26	4 GB
	hdisk376	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk377	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk378	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk379	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk380	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk381	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk382	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk383	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB

	hdisk384	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
	hdisk385	lviordl1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16	4 GB
ssa2	hdisk3	lvlog	16.9 GB
	hdisk4	lvlog	16.9 GB
	hdisk387	paging	9 GB
	hdisk388	paging	9 GB
	hdisk389	lvlog	16.9 GB
	hdisk393	lvlog	16.9 GB
	hdisk398	lvlog	16.9 GB
	hdisk399	lvlog	16.9 GB
	hdisk405	lvlog	16.9 GB
	hdisk409	lvlog	16.9 GB
	hdisk410	lvlog	16.9 GB

5.3 Data Base Model Implemented

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

The database manager used for this testing was Oracle8 Enterprise Edition 8 .0 from Oracle Inc. Oracle8 Enterprise Edition 8.0 is a relational DBMS.

5.4 Partitions/Replications Mapping

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

No horizontal or vertical partitioning were implemented for these TPC-C tests.

5.5 180 day space calculations ESCALA RL470

TPM	18666.73
Warehouses	1700.00

SEGMENT	TYPE	TSPACE	BLOCKS	FIVE_PCT	DAILY_GROW	TOTAL
CUSTOMER	TABLE	CUST	12750011.00	637500.55	0	13387511.55
DISTRICT	TABLE	WARE	17004.00	850.20	0	17854.20
HISTORY	TABLE	HIST	692005.00		121576.19	813581.19
ICUSTOMER	INDEX	ICUST1	284032.00	14201.60	0	298233.60
ICUSTOMER2	INDEX	ICUST2	634969.00	31748.45	0	666717.45
IDISTRICT	INDEX	WARE	3330.00	166.50	0	3496.50
IITEM	INDEX	ITEMS	2050.00	102.50	0	2152.50
INEW_ORDER	INDEX	INORD	90950.00	4547.50	0	95497.50
IORDERS	INDEX	IORD1	280998.00	14049.90	0	295047.90
IORDERS2	INDEX	IORD2	453718.00	22685.90	0	476403.90
IORDER_LINE	INDEX	IORDL	3240832.00	162041.60	0	3402873.60
ISTOCK	INDEX	ISTK	847254.00	42362.70	0	889616.70
ITEM	TABLE	ITEMS	3031.00	151.55	0	3182.55
IWAREHOUSE	INDEX	WARE	350.00	17.50	0	367.50
NEW_ORDER	TABLE	NORD	65399.00	3269.95	0	68668.95
ORDERS	TABLE	ORD	503424.00	0.00	88444.99	591868.99
ORDER_LINE	TABLE	ORDL	9184217.00	0.00	1613546.34	10797763.34
ROLL_SEG	SYS	ROLL	26368.00	0.00	0	26368.00
STOCK	TABLE	STOCKS	15454547.00	772727.35	0	16227274.35
SYSTEM	SYS	SYSTEM	110080.00	0.00	0	110080.00
WAREHOUSE	TABLE	WARE	1704.00	85.20	0	1789.20
Total			44646273.00	1706508.95	1823567.52	48176349.47

Dynamic space	10379646.00	
Static space	35973135.95	
Free space	1823567.52	
Daily growth	1823567.52	
Daily spread	0.00	Oracle may be configured such that daily spread is 0
180-day space (blk.)	364215290.19	
Block size (bytes)	4096.00	
180-day (GB)	1389.37	
Log block size	512.00	
Log blocks/tpmC	33.90	Number of log blocks used in one tpmC
8-hour log (GB)	144.85	

Disk Type	Disk ormatted Capacity	SUT # of disks	SUT Capacity(GB)	Priced # of disks	Priced Capacity(GB)
2.2	2148	96	201.38		
4.5	4296	281	1178.88	377	1581.63
9.1	8672	2	16.94	2	16.94
RAID(3-9.1GB)	17344.00	9.00	156096.00	9.00	152.44
8 hour log					1598.57
rollbacks	5667.00				
new order	560002.00				
payment	540985.00				
delivery	509830.00				
Commits per TpmC	2.86				
Redo blocks written	18985708.00				
Redo blocks written per commit		11.84			
Redo space per tpmC	33.90				

Space usage (GB)	
180-day	1389.37
RIAD	152.44
os+paging	4.20
Paging	16.94
Total Space	1562.95

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 lists the response times and the ninetieth percentiles for each of the transaction types for the measured systems.

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 lists the TPC-C keying and think times for the measured systems.

Table 6-1 Escala RL470 Response, Think and Keying Times

Response Times	New Order	Payment	Order Status	Delivery (int./def.)	Stock Level	Menus
90 %	3.57	3.23	3.24	0.20/4.99	3.04	0.02
Average	1.56	1.31	1.31	0.12/2.02	1.16	0.01
Maximum	11.43	11.3	10.85	2.23/17.8	10.77	1.51
			Think Times			
Minimum	0.01	0.01	0.01	0.01	0.01	N/A
Average	12.01	12.02	10.11	5.12	5.12	N/A
Maximum	121	121	101	51	51	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	18.11	3.12	2.1	2.1	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. Escala RL470 New-Order Response Time Distribution

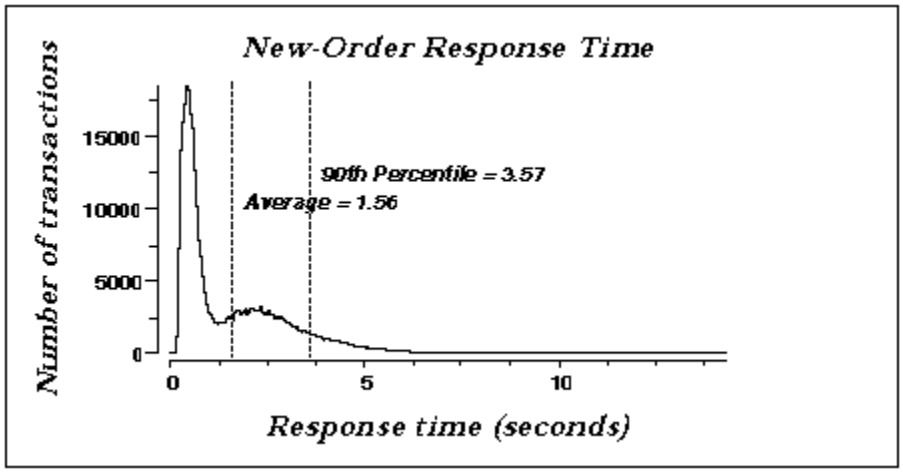


Figure 6-3-2. Escala RL470 Payment Response Time Distribution

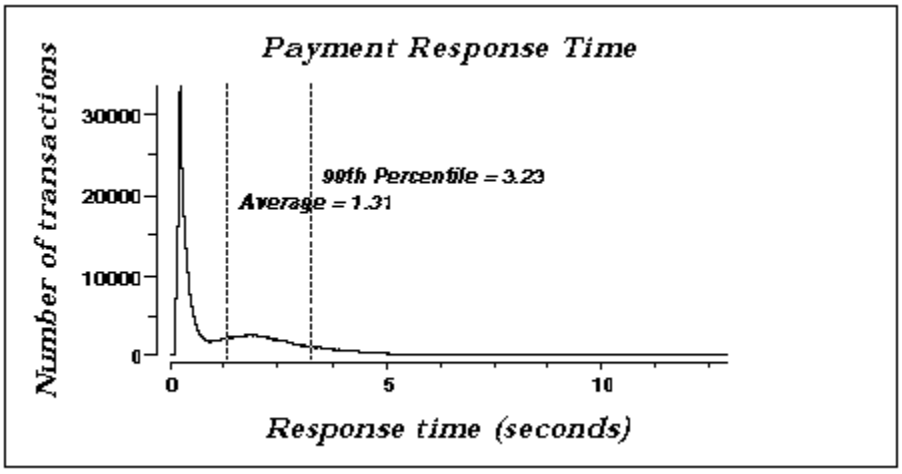


Figure 6-3-3. Escala RL470 Order-Status Response Time Distribution

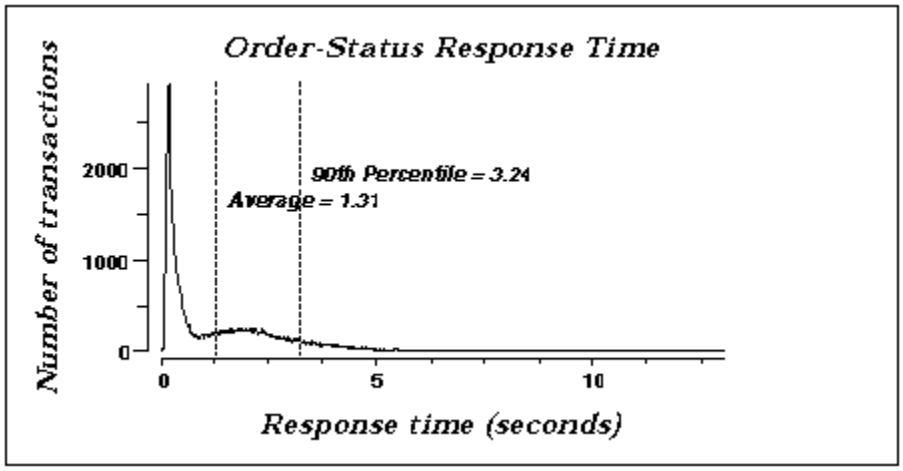


Figure 6-3-4. RL470 Delivery (Interactive) Response Time Distribution

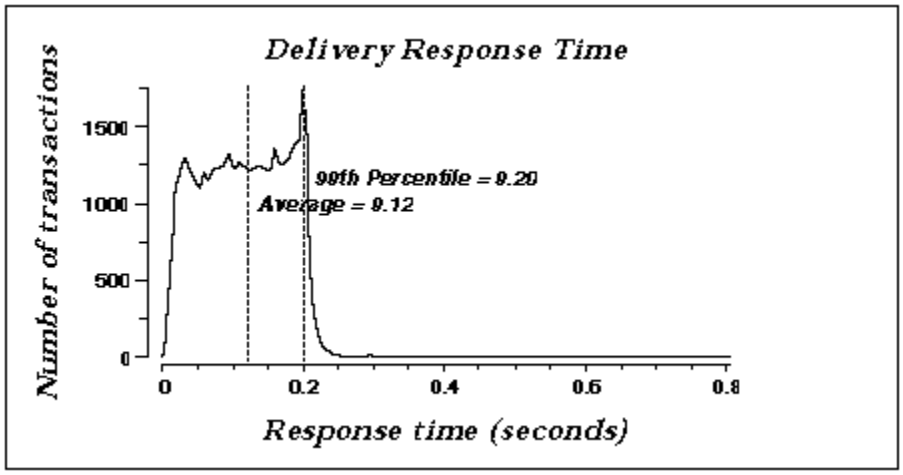


Figure 6-3-5. RL470 Delivery (Deferred) Response Time Distribution

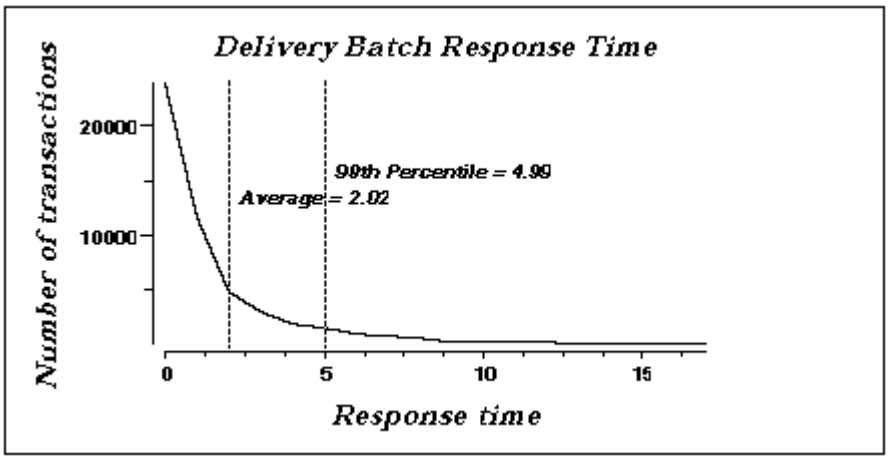
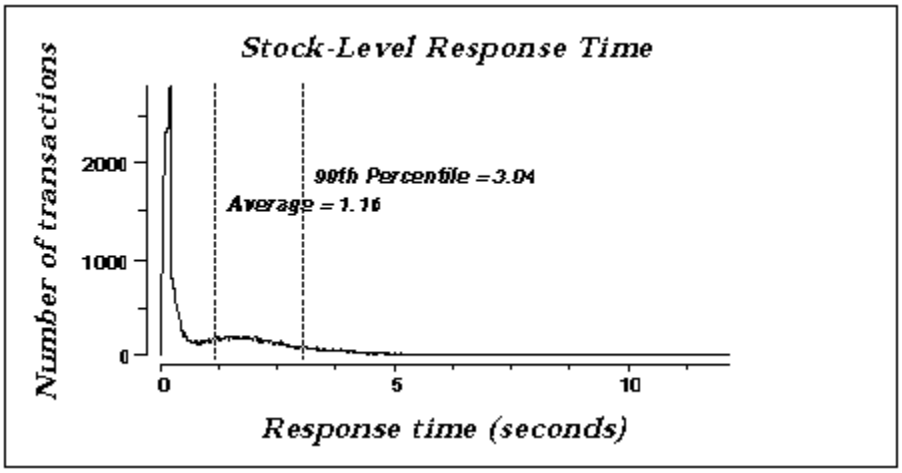


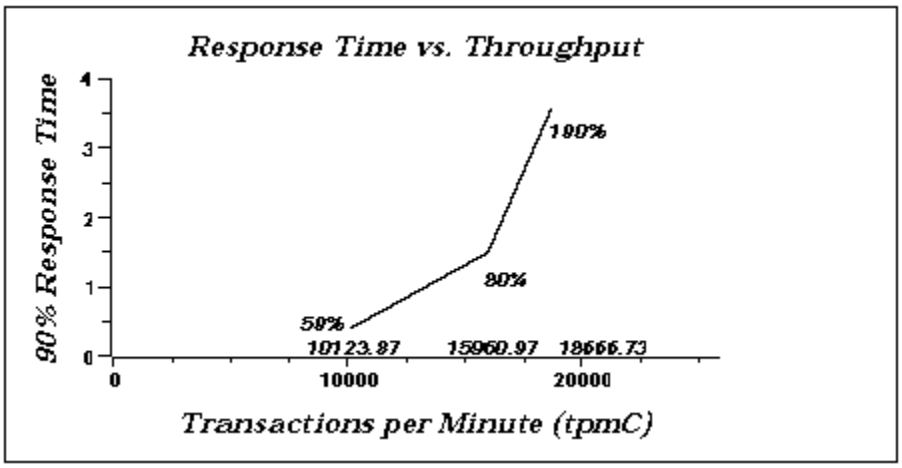
Figure 6-3-6. Escala RL470 Stock-Level Response Time Distribution



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.

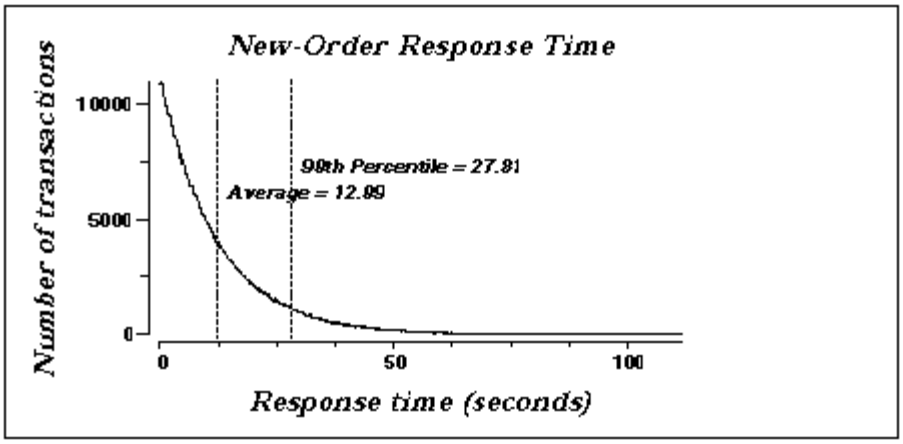
Figure 6-4-1. RL470 New-Order Response Time vs. Throughput



6.5 Think Time Frequency Distribution

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

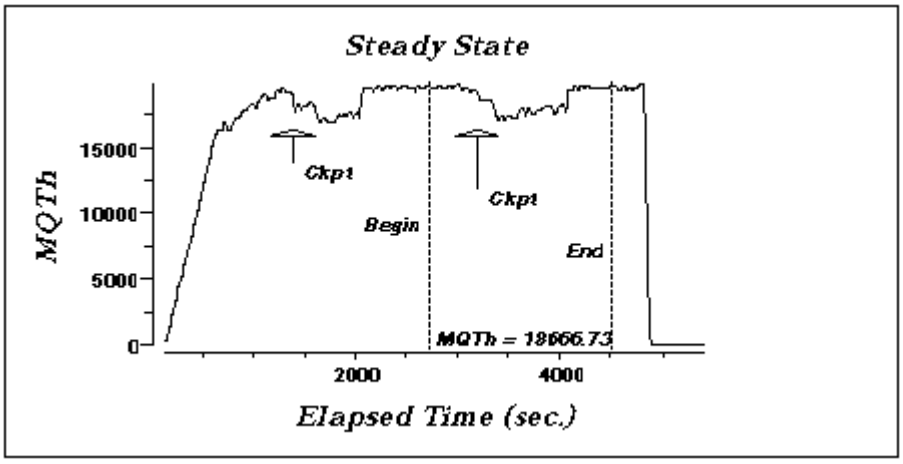
Figure 6-5-1. Escala RL470 New-Order Think Time Distribution



6.7 Throughput versus Elapsed Time

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

Figure 6-7-1. Escala RL470 New-Order Throughput vs. Elapsed Time



6.8 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 timestamping interval was set to start after several minutes of rampup. Refer to the Numerical Quantities Summary pages for the rampup times for the measured system. Figure 6.7.1 New-Order throughput versus Elapsed Time graph shows that the system was in steady state at the beginning of the Measurement Interval.

6.9 Work Performed During Steady State

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

6.9.1 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: a front end program which handled all screen I/O, a database client program which connected to the database and served as a TXSeries server server(a backend 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 listed 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 dilivery transactions and one all other transaction.
- The delivery server is configured with one processing agent with 3 server manager DCE threads, and 3 background threads to process deffered deliveries. Each background thread has one connection to the database.
- The other server is configured with 6 processing agents. Each processing agent has 5 server manager DCE threads. Each thread has one connection to the database.
- When the Encinaclients 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 ter minal.
- The TPC-C user chooses the transaction type and proceeds to fill the screen fields required for transaction.
- The process_terminal accepts all values entered by the user and transmites 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 dilivery 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 outpu t parameters. The RPC is then sent back to the client program.
- When the RPC returns to the client, the process_terminal routine writes the transaction out on the user terminal.

6.9.2 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 addtional 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.

6.9.3 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 ocurring during the measurement interval.

6.10 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 18,580.40 tpmC.

6.11 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.

The RTE used for these tests was an internal IBM tool. 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 IBM RS/6000 Model 39H emulates a network of 1,100 RS/6000 Model 43P-140 workstations. The communications mechanism used in the benchmarked 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 pr iced configuration is used to connect 734 workstations.

7.3 Network Bandwidth

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

The Ethernet used in the LAN complies with the IEEE 802.3 standard and has a bandwidth of 10 Megabits per second. Each LAN segment in the RS/6000 Enterprise Server S70 configuration connected 734 workstations.

7.4 Operator Intervention

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

The Escala RL470 configuration reported do not require any operator intervention 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 BULL are available the same day as product or feature availability.

The detailed list of hardware and programs for the priced configuration is listed in the pricing sheet (refer to the executive summary statement). Prices for all Bull S.A. products are US list prices. Each priced configuration consists of an integrated system package, additional components and third party components.

The Escala RL 470 Packaged system referenced by the MI CPXG213-0000 includes :

12 CPU model Power PC RS64 @125Mhz-4MB L2 cache/CPU
16GB of memory
1x9 GB disk
1x SCSI-2 F/W DE Ext Disk Adapter
1 CD ROM
1 floppy disk 1.44MB 3”1/2
1 IO Rack, 1 IO drawer

Pricing for IBM products have been officially disclosed in a recent IBM TPC-C FDR in a quotation made by Dickens Data Systems on February 27th, valid 90 days.
So No new quotation concerning IBM products is usefull in this FDR.

Pricing for Linksys 20 port Ethernet Hubs was quoted by Computer Discount Warehouse.

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 Escala RL470 are contained on the first pages.
A Bull S.A. 15% dollar volume discount on Hardware and Software is applicable to Hardware configurations above $500,000.
The five years support pricing for Bull S.A.consists of one year warranty included in the system package price and four years support price.

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 currently generally available today except for the following products (general availability dates are provided):

Product	Availability Date
Oracle 8 Enterprise Edition 8.0	**September 2, 1998**

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	tmpC	5-year SystemCost	$/tmpC	Availability Date
Escala RL470	**18,666.73tpmC**	**$2,177,811**	**$116.66/tpmC**	**now except parts noted in section 8.3**

8. 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 below.

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

Sponsor: Jean-François Lemerre
Bull S.A.
1, rue de Provence
Echirolles, 38432
FRANCE

March 25, 1998

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

Platform: ESCALA RL470 c/s
Operating system: AIX 4.3
Database Manager: Oracle8 Enterprise Edition 8.0
Transaction Manager: IBM TX Series 4.2 for AIX

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: ESCALA RL470 c/s				
12 x PowerPC RS64 (125 MHz)	16.0 GB (4 MB L2 Cache per processor)	95 x 2.2 GB 282 x 4.5 GB 29 x 9.1 GB	3.57 Seconds	18666.73
Eight Clients: RISC System/6000 Workgroup Server 43P-140 (Specification for each)				
1 x PowerPC 604e (233 MHz)	640 MB	1 x 2.2 GB	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for Revision 3.3 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

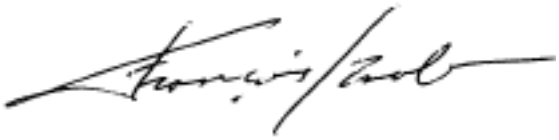
1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office : 719/473-7555 • Fax : 719/473-7554

- 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 (95) 2.2 GB disks used in the tested configuration were substituted with 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 0
#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 = 60;
} else if (ran > 990) {
timeP->tv_sec = 10;
} else if (ran > 950) {
timeP->tv_sec = 2;
} else {
timeP->tv_sec = 0;
}
timeP->tv_sec *= 10;
timeP->tv_nsec = 150000000;
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)
char *dbName;
int flag;
{
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);
rc = TPCinit (serverIdNumber, "tpcc", "tpcc");
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++) {
I_CP(i, ol_amount, ol_amount);
I_CP(i, ol_i_id, ol_i_id);
I_CP(i, ol_supply_w_id, ol_supply_w_id);
I_CP(i, ol_quantity, ol_quantity);
I_SCP(i, ol_delivery_d, delivery_date, 11);
}
#undef I_CP
#undef I_SCP
TPCCP_RET_SCP(ordout.c_first, c_first, 17);
TPCCP_RET_SCP(ordout.c_middle, c_middle, 3);
TPCCP_RET_SCP(ordout.c_last, c_last, 17);
TPCCP_RET_SCP(ordout.o_entry_d, entry_date, 20);
}
void do_order_status(dataP)
orderStatus_data_t *dataP;
{
struct ordstruct oraStruct;
int i, rc;
AUDITLOG(("> do_order_status\n"));
if (server_null_test) {
sim_order_status(dataP);
return;
}
TPCC_CP(ordin.w_id, w_id);
TPCC_CP(ordin.d_id, d_id);
TPCC_CP(ordin.c_id, c_id);
oraStruct.ordin.bylastname = ((dataP->c_id == 0) ? 1 : 0);
TPCC_SCP(ordin.c_last, c_last, 17);
DEBUGP(("Calling TPCord: w_id %d, d_id %d, c_id %d, bylastname %d, c_last %s\n",
oraStruct.ordin.w_id, oraStruct.ordin.d_id, oraStruct.ordin.c_id,
oraStruct.ordin.bylastname, oraStruct.ordin.c_last));
rc = TPCord(&oraStruct);
if ( rc != 0 ) {
err_printf("Error TPCord: terror %d, rc %d, retry %d, w_id %d, d_id %d, c_id %d,
bylastname %d, c_last %s\n ",
oraStruct.ordout.terror, rc, oraStruct.ordout.retry,
oraStruct.ordin.w_id, oraStruct.ordin.d_id, oraStruct.ordin.c_id,
oraStruct.ordin.bylastname, oraStruct.ordin.c_last);
}
copyout_order_status(dataP, &oraStruct);
dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.ordout.terror;
AUDITLOG(("< do_order_stats rc %d\n", dataP->header.returncode));
}
void do_stock_level(dataP)
stockLevel_data_t *dataP;
{
struct stostruct oraStruct;
/* What's this comment?? -- srs: i only did this one to check the links */
int rc;
AUDITLOG(("> do_stock_level\n"));
if (server_null_test) {
sim_stock_level(dataP);
return;
}
TPCC_CP(stoin.w_id, w_id);
TPCC_CP(stoin.d_id, d_id);
TPCC_CP(stoin.threshold, threshold);
DEBUGP(("Calling TPCsto: w_id %d, d_id %d, threashold %d\n",
oraStruct.stoin.w_id, oraStruct.stoin.d_id,
oraStruct.stoin.threshold));
rc = TPCsto( &oraStruct );
if ( rc != 0 ) {
err_printf("Error TPCsto : terror %d, rc %d, retry %d, w_id %d, d_id %d, threashold
%d\n",
oraStruct.stoout.terror, rc, oraStruct.stoout.retry,
oraStruct.stoin.w_id, oraStruct.stoin.d_id,
oraStruct.stoin.threshold);
}
TPCC_RET_CP(stoout.low_stock, stock_count);
dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.stoout.terror;
DEBUGP(("do_stock_lev returning %d\n", dataP->header.returncode));
AUDITLOG(("< do_stock_level rc %d\n", dataP->header.returncode));
}
void copyin_payment(dataP, oraStructP)
payment_data_t *dataP;
struct paystruct *oraStructP;
{
TPCCP_CP(payin.w_id, w_id);
TPCCP_CP(payin.d_id, d_id);
TPCCP_CP(payin.c_w_id, c_w_id);
TPCCP_CP(payin.c_d_id, c_d_id);
TPCCP_CP(payin.c_id, c_id);
oraStructP->payin.bylastname = ((dataP->c_id == 0) ? 1 : 0);
```

```
TPCCP_CP(payin.h_amount, h_amount);
TPCCP_SCP(payin.c_last, c_last, 17);
}
void copyout_payment(dataP, oraStructP)
payment_data_t *dataP;
struct paystruct *oraStructP;
{
TPCCP_RET_SCP(payout.w_street_1, w_street_1, 21);
TPCCP_RET_SCP(payout.w_street_2, w_street_2, 21);
TPCCP_RET_SCP(payout.w_city, w_city, 21);
TPCCP_RET_SCP(payout.w_state, w_state, 3);
TPCCP_RET_SCP(payout.w_zip, w_zip, 10);
TPCCP_RET_SCP(payout.d_street_1, d_street_1, 21);
TPCCP_RET_SCP(payout.d_street_2, d_street_2, 21);
TPCCP_RET_SCP(payout.d_city, d_city, 21);
TPCCP_RET_SCP(payout.d_state, d_state, 3);
TPCCP_RET_SCP(payout.d_zip, d_zip, 10);
TPCCP_RET_CP(payout.c_id, c_id);
TPCCP_RET_SCP(payout.c_first, c_first, 17);
TPCCP_RET_SCP(payout.c_middle, c_middle, 3);
TPCCP_RET_SCP(payout.c_last, c_last, 17);
TPCCP_RET_SCP(payout.c_street_1, c_street_1, 21);
TPCCP_RET_SCP(payout.c_street_2, c_street_2, 21);
TPCCP_RET_SCP(payout.c_city, c_city, 21);
TPCCP_RET_SCP(payout.c_state, c_state, 3);
TPCCP_RET_SCP(payout.c_zip, c_zip, 10);
TPCCP_RET_SCP(payout.c_phone, c_phone, 17);
TPCCP_RET_SCP(payout.c_since, c_date, 11);
TPCCP_RET_SCP(payout.c_credit, c_credit, 3);
TPCCP_RET_CP(payout.c_credit_lim, c_credit_lim);
TPCCP_RET_CP(payout.c_discount, c_discount);
TPCCP_RET_CP(payout.c_balance, c_balance);
TPCCP_RET_SCP(payout.c_data, c_data, 201);
TPCCP_RET_SCP(payout.h_date, pay_date, 20);
strcpy((char *)dataP->w_name, "W_NAME");
strcpy((char *)dataP->d_name, "D_NAME");
/* Ignore c_ytd_payment, c_payment_cnt */
}
void do_payment(dataP, xaFlag)
payment_data_t *dataP;
int xaFlag;
{
struct paystruct oraStruct;
int firstWh, secondWh;
int rc;
AUDITLOG(("> do_payment\n"));
if (server_null_test) {
sim_payment(dataP);
return;
}
copyin_payment(dataP, &oraStruct);
#if 0
err_printf("TPCpay: w_id %d, D_id %d, C_w_id %d, c_id %d, bylastname %d, amount
%.2f, c_last %s (%s)\n",
oraStruct.payin.w_id,
oraStruct.payin.d_id,
oraStruct.payin.c_w_id,
oraStruct.payin.c_id,
oraStruct.payin.bylastname,
oraStruct.payin.h_amount,
oraStruct.payin.c_last,
dataP->c_last);
#endif
rc = TPCpay(&oraStruct);
#if 0
err_printf("< TPCpay terror %d, rc %d, retry %d\n",
oraStruct.payout.terror, rc, oraStruct.payout.retry);
#endif
dataP->header.num_rms = 1;
if (rc != 0 ) {
err_printf("Error TPCpay: terror %d, rc %d, retry %d, w_id %d, D_id %d, C_w_id %d,
c_id %d, bylastname %d, amount %.2f, c_last %s (%s)\n",
oraStruct.payout.terror, rc, oraStruct.payout.retry,
oraStruct.payin.w_id,
oraStruct.payin.d_id,
oraStruct.payin.c_w_id,
oraStruct.payin.c_id,
oraStruct.payin.bylastname,
oraStruct.payin.h_amount,
oraStruct.payin.c_last ? oraStruct.payin.c_last : "-NULL-",
(char *)dataP->c_last ? (char *)dataP->c_last : "-NULL-");
}
copyout_payment(dataP, &oraStruct);
dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.payout.terror;
AUDITLOG(("< do_payment rc %d\n", dataP->header.returncode));
}
static void copyin_new_order(dataP, oraStructP)
newOrder_data_t *dataP;
struct newstruct *oraStructP;
{
int i;
TPCCP_CP(newin.w_id, w_id);
TPCCP_CP(newin.d_id, d_id);
TPCCP_CP(newin.c_id, c_id);
#define NO_I_CP(ind,a,b) oraStructP->a[ind] = dataP->item[ind].b
#define NO_I_SCP(ind,a,b,len) strncpy((char *)oraStructP->a[ind], (char
*)dataP->item[ind].b, len)
/* tpccpl.c loops over 15 items, we do the same */
for (i=0; i<15; i++) {
NO_I_CP(i, newin.ol_i_id, ol_i_id);
NO_I_CP(i, newin.ol_supply_w_id, ol_supply_w_id);
NO_I_CP(i, newin.ol_quantity, ol_quantity);
#ifdef DEBUG_SERVER
fprintf(stderr, "NewOrder: Item %d, supplyWh %d (local %d)\n",
i,
oraStructP->newin.ol_supply_w_id[i],
oraStructP->newin.w_id);
#endif
}
/* Ignore all_local field, total_items,
* tpccpl.c doesnt use them
*/
#undef NO_I_CP
#undef NO_I_SCP
}
void copyout_new_order(dataP, oraStructP)
newOrder_data_t *dataP;
struct newstruct *oraStructP;
{
int i;
TPCCP_RET_CP(newout.o_id, o_id);
TPCCP_RET_CP(newout.o_ol_cnt, o_ol_cnt);
TPCCP_RET_SCP(newout.c_last, c_last, 17);
TPCCP_RET_SCP(newout.c_credit, c_credit, 3);
TPCCP_RET_CP(newout.c_discount, c_discount);
TPCCP_RET_CP(newout.w_tax, w_tax);
TPCCP_RET_CP(newout.d_tax, d_tax);
TPCCP_RET_SCP(newout.o_entry_d, entry_date, 20);
TPCCP_RET_CP(newout.total_amount, total);
TPCCP_RET_SCP(newout.status, statusline, 26);
#define NO_RET_CP(ind,a,b) dataP->item[ind].b = oraStructP->newout.a[ind]
#define NO_RET_SCP(ind,a,b,len) strncpy((char *)dataP->item[ind].b, (char
*)oraStructP->newout.a[ind], len)
for (i=0; i<oraStructP->newout.o_ol_cnt && i<15; i++) {
NO_RET_SCP(i, i_name, name_i, 25);
NO_RET_CP(i, s_quantity, s_quantity);
dataP->item[i].brand_generic[0] = oraStructP->newout.brand_generic[i];
dataP->item[i].brand_generic[1] = '\0';
NO_RET_CP(i, i_price, price);
NO_RET_CP(i, ol_amount, ol_amount);
/* Ignore s_idx and s_dist */
}
if (oraStructP->newout.status[0] != '\0') {
DEBUGP(("TPCnew: status -- %s\n", oraStructP->newout.status));
dataP->items_valid = 0;
} else {
dataP->items_valid = 1;
}
#undef NO_RET_CP
#undef NO_RET_SCP
}
void do_new_order(dataP, xaFlag)
newOrder_data_t *dataP;
int xaFlag;
{
static int num_calls = 0;
int i;
struct newstruct oraStruct;
int rc;
AUDITLOG(("> do_new_order\n"));
if (server_null_test) {
sim_new_order(dataP);
return;
}
/* Copy the structure into the TPCC structure. */
copyin_new_order(dataP, &oraStruct);
DEBUGP(("-> TPCnew %d items to wh %d\n",
dataP->o_ol_cnt, dataP->w_id));
dataP->header.num_rms = 1;
#if 0
err_printf("Error TPCnew : w_id %d, d_id %d, c_id %d, o_ol_cnt %d (out cnt %d)\n",
oraStruct.newin.w_id, oraStruct.newin.d_id,
oraStruct.newin.c_id, dataP->o_ol_cnt, oraStruct.newout.o_ol_cnt);
for (i=0; i<15; i++) {
err_printf("ol_i_id %d, ol_supply_w_id %d, ol_quantity %d\n",
oraStruct.newin.ol_i_id[i], oraStruct.newin.ol_supply_w_id[i],
oraStruct.newin.ol_quantity[i]);
}
#endif
rc = TPCnew(&oraStruct);
#if 0
err_printf("< TPCnew terror %d, rc %d, retry %d\n",
oraStruct.newout.terror, rc, oraStruct.newout.retry);
#endif
if (rc != 0) {
err_printf("Error TPCnew : terror %d, rc %d, retry %d, w_id %d, d_id %d, c_id %d,
o_ol_cnt %d (out cnt %d)\n",
oraStruct.newout.terror, rc, oraStruct.newout.retry,
oraStruct.newin.w_id, oraStruct.newin.d_id,
oraStruct.newin.c_id, dataP->o_ol_cnt, oraStruct.newout.o_ol_cnt);
for (i=0; i<15; i++) {
err_printf("ol_i_id %d, ol_supply_w_id %d, ol_quantity %d\n",
oraStruct.newin.ol_i_id[i], oraStruct.newin.ol_supply_w_id[i],
oraStruct.newin.ol_quantity[i]);
}
}
DEBUGP(("-< TPCnew %d\n", rc));
/* copy out results */
copyout_new_order(dataP, &oraStruct);
if (rc == 0) {
dataP->header.returncode =
dataP->items_valid ? TPCC_SUCCESS : INVALID_NEWO;
#if 0
if (dataP->items_valid && (++num_calls % 500) == 0) {
int i;
err_printf("TPCnew Success: w_id %d, d_id %d, c_id %d, o_ol_cnt %d, Oid %d\n",
oraStruct.newin.w_id, oraStruct.newin.d_id,
oraStruct.newin.c_id, oraStruct.newout.o_ol_cnt,
oraStruct.newout.o_id);
for (i=0; i<15 && i<oraStruct.newout.o_ol_cnt; i++) {
topr_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)
char *dbName;
int flag;
int num_cn;
{
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: 1.2 $
* $Date: 1998/01/26 20:37:34 $
* $Log: $
*
*
* $TALog: client_bg_thread.c,v $
* 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>
#include <dce/pthread.h>
#include <tpm/mon/mon.h>
#include <tpm/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"
#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);
static struct timeval *client_last_time(thread_descr_t *descrP);
/*
* 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)
{
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);
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;
sleep(60);
}
}
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;
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));
for (i=0; i<info_list_len; i++) {
struct timeval cur_time;
struct timeval *client_timeP;
int time_diff;
thread_descr_t *descrP;
int delta;
if (info_list[i] == NULL || !info_list[i]->initialized) {
continue;
}
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_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_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;
}
}
```

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_init (int infd, int outfd, void *contextP) {
int rc = 0;
Thread_data *threadP = new Thread_data(infd, outfd, contextP);
Field *menuField = new IntField(threadP, 8);
User_data user_data;
Login log(&user_data, threadP);
Menu menu(&user_data, threadP);
log.handle();
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 *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 *thread_init(void);
#endif /* TPCC_CLIENT_H */
```

client_listen.c

```
/*
* client_listen.c
*
* $Revision: 1.13 $
* $Date: 1998/01/26 16:19:22 $
* $Lo: $
*
*
* $TALog: client_listen.c,v $
* 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>
#include <stdio.h>
#include <sys/types.h>
#include <tc/tc.h>
#include <do_tpcc.h>
#include <tpcc_type.h>
#include <cnm/cnm.h>
#if defined (solaris)
#include <dce/pthread.h>
#else /* solaris */
#include <pthread.h>
#endif /* solaris */
#include "client_utils.h"
/*
* 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, void *);
/*
* 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)
{
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;
tid_t tid = thread_self();
char *host_name = getenv("HOST");
logprintf("Tid: %d (0x%x) pid %d\n", tid, tid, getpid());
fdIn = fileno(argP->stream);
/*
* Default terminal context
* This may be updated later by the terminal
*/
terminal_context.terminal_id = -1;
terminal_context.profiling = 0;
/* Initialize the server handle and other thread structures */
terminal_context.handle_contextP = (void *)thread_init();
client_init(fdIn, fdIn, terminal_context.handle_contextP);
/* Sleep a random amount so that not all the threads
* close the sockets at the same time
*/
close(fdIn);
logprintf("Thread done - Tid: %d (0x%x)\n", tid, tid);
thread_done(terminal_context.handle_contextP);
}
/*
* 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_utils.c

```
/*
*
* client_utils.c
*
* $Revision: 1.5 $
* $Date: 1998/01/24 14:17:04 $
* $Log: $
*
*
*
*
* $TALog: client_utils.c,v $
* 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);
}
```

client_utils.h

```
/*
*
* client_utils.h
*
* $Revision: 1.6 $
* $Date: 1998/01/26 16:43:32 $
* $Log: $
*
*
*
*
* $TALog: client_utils.h,v $
* 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 "tpcc_type.h"
#include <stdio.h>
#include <time.h>
#include "client.h"
/*
* 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, ...);
/* 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
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;
} 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];
} 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__ */
```

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
#
#
### The following definitions are used to compile
### Using the Transarc Standard (internal) environment
### At Transarc
##ENC_DIR = /afs/transarc.com/kansas/rTO/internal
ENC_DIR = /usr/lpp/encina
DB2_HOME = /database/home/db2v2lcl
CC = xlc_r4
CXX=xlC_r4
DEBUG=-O
CFLAGS=$(DEBUG) -DUSE_ALLOCA
CXXFLAGS=$(DEBUG) -DUSE_ALLOCA
##CC = xlC_r4 -g -DUSE_ALLOCA
OBJS=client.o screen_data.o screen.o inout.o format.o field.o encina.o
client: libClient.a client_main.o debug.o
$(CXX) client_main.o debug.o -o client -L. -lClient
libClient.a: $(OBJS)
ar crv libClient.a $(OBJS)
clean:
rm -f core *.o client format_test libClient.a
```

mon_client.c

```
/*
* mon_client.c
*
* $Revision: 1.19 $
* $Date: 1998/01/26 20:37:35 $
* $Log: $
*
*
* $TALog: mon_client.c,v $
* 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 <dce/pthread.h>
#include <tpm/mon/mon.h>
#include <tpm/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"
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).
*/
#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); \
} \
}
/*
* 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);
}
mon_ExitClient( err );
exit( err );
}
/*
* 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 *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->initialized = 0;
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]);
} else {
int ind = infoP->thread_index;
for (j=1; j<=MAX_TRAN_TYPE; j++) {
total_counts.tran[j].num += info_list[ind]->tran[j].num;
total_counts.tran[j].errs += info_list[ind]->tran[j].errs;
total_counts.tran[j].RT[0] += info_list[ind]->tran[j].RT[0];
total_counts.tran[j].RT[1] += info_list[ind]->tran[j].RT[1];
total_counts.errors += info_list[ind]->tran[j].errs;
total_counts.tran[0].num += info_list[ind]->tran[j].num;
}
err_printf("> So far, %d trans %d errs\n",
total_counts.tran[0].num, total_counts.errors);
}
MUTEX_UNLOCK(&init_lock);
if (all_done) {
static char *names[] = {"0", "newo", "pay ", "ord ", "dvry", "stok"};
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;
fprintf(stderr, "Client Done. Total of %d trans and %d errors\n",
total_counts.tran[0].num, total_counts.errors);
for (j=1; j<=MAX_TRAN_TYPE; j++) {
fprintf(stderr, " Tran %s, num %d, errs %d, RT avg %.3f\n",
names[j],
total_counts.tran[j].num,
total_counts.tran[j].errs,
(total_counts.tran[j].RT[0] / 1000.) / total_counts.tran[j].num);
}
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;
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;
memset(&total_counts, 0, sizeof(total_counts));
gettimeofday(&a_time, &tz);
srand48(a_time.tv_sec ^ a_time.tv_usec);
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;
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.
*
*/
#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 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;
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 < 1 || 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 {
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) {
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;
inFunction("server_Init");
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);
set_scheduling(scheduling);
register_interfaces(interface_type);
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);
if ((rc = get_db_ready("tpcc/tpcc", 0)) != 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 ++;
err_printf(">> 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);
dataP->start_queue = (double)now.tv_sec + (now.tv_usec / 1000000.0);
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 > 29) {
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);
queue_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_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/sp-tpcc/ora8MT_encMT/RC
S/pldel.c,v 1.2 1998/01/23 15:08:05 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */
/*+=====================================================================+
| Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=====================================================================+
| FILENAME
| pldel.c
| DESCRIPTION
| OCI version of DELIVERY transaction in TPC-C benchmark.
+=====================================================================*/
#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
#define SQLTXT0 "SELECT substr(value,1,5) FROM v$parameter \
WHERE name = 'instance_number'"
#endif
#define SQLTXT1 "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 1, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 1 AND o_w_id = :w_id AND o_d_id = 1 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 2, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 2 AND o_w_id = :w_id AND o_d_id = 2 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 3, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 3 AND o_w_id = :w_id AND o_d_id = 3 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 4, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 4 AND o_w_id = :w_id AND o_d_id = 4 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 5, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 5 AND o_w_id = :w_id AND o_d_id = 5 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 6, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
```

```
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 6 AND o_w_id = :w_id AND o_d_id = 6 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 7, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 7 AND o_w_id = :w_id AND o_d_id = 7 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 8, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 8 AND o_w_id = :w_id AND o_d_id = 8 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 9, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 9 AND o_w_id = :w_id AND o_d_id = 9 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 10, no_o_id, new_order.rowid,
o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 10 AND o_w_id = :w_id AND o_d_id = 10 AND \
o_id = no_o_id AND rownum <= 1"
#define SQLTXTTEST "INSERT INTO vv VALUES (:no_rowid, :o_rowid)"
#define SQLTXT2 "DELETE FROM new_order WHERE rowid = :no_rowid"
/*
#define SQLTXT2 "DELETE FROM new_order WHERE no_o_id = :o_id and no_d_id = :d_id
and \
no_w_id = :w_id"
*/
#define SQLTXT3 "UPDATE orders SET o_carrier_id = :carrier_id \
WHERE rowid = :o_rowid"
/*
#define SQLTXT3 "UPDATE orders SET o_carrier_id = :carrier_id \
WHERE o_id = :o_id and o_d_id = :d_id and o_w_id = :w_id \
and o_c_id = :c_id"
*/
#define SQLTXT4 "UPDATE order_line SET ol_delivery_d = :cr_date \
WHERE ol_w_id = :w_id AND ol_d_id = :d_id AND ol_o_id = :o_id"
#define SQLTXT5 "\
SELECT :d_id1, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id1 AND ol_o_id = :o_id1 UNION ALL \
SELECT :d_id2, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id2 AND ol_o_id = :o_id2 UNION ALL \
SELECT :d_id3, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id3 AND ol_o_id = :o_id3 UNION ALL \
SELECT :d_id4, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id4 AND ol_o_id = :o_id4 UNION ALL \
SELECT :d_id5, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id5 AND ol_o_id = :o_id5 UNION ALL \
SELECT :d_id6, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id6 AND ol_o_id = :o_id6 UNION ALL \
SELECT :d_id7, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id7 AND ol_o_id = :o_id7 UNION ALL \
SELECT :d_id8, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id8 AND ol_o_id = :o_id8 UNION ALL \
SELECT :d_id9, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id9 AND ol_o_id = :o_id9 UNION ALL \
SELECT :d_id10, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id10 AND ol_o_id = :o_id10"
#define SQLTXT6 "UPDATE customer SET c_balance = c_balance + :amt, \
c_delivery_cnt = c_delivery_cnt + 1 WHERE c_w_id = :w_id AND \
c_d_id = :d_id AND c_id = :c_id"
pldelinit (ora_cn_data_t *ora_SlotDataP)
{
delctx *dctx;
global_delivery_t *delP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
int i,j;
char bstr1[10];
char bstr2[10];
text stmbuf[4096];
dctx = (delctx *) malloc (sizeof(delctx));
memset(dctx,(char)0,sizeof(delctx));
ora_SlotDataP->dctx = dctx;
delP = (global_delivery_t *)malloc(sizeof(global_delivery_t));
memset(delP, (char)0, sizeof(global_delivery_t));
ora_SlotDataP->delP = delP;
dctx->norow = 0;
for(i=0;i<NDISTS;i++) {
/*
dctx->o_rowid_ptr[i] = &(dctx->o_rowid[i][0]);
dctx->no_rowid_ptr[i] = &(dctx->no_rowid[i][0]);
*/
OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&dctx->o_rowid_ptr[i],
OCI_DTYPE_ROWID,0,(dvoid**)0));
OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&dctx->no_rowid_ptr[i],
OCI_DTYPE_ROWID,0,(dvoid**)0));
}
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd0), OCI_HTYPE_STMT, 0, (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT0);
OCIStmtPrepare(dctx->curd0, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);
OCIDFNRA(dctx->curd0, dctx->inum_dp,errhp,1,dctx->inum,SIZ(dctx->inum),SQLT_STR,
&(dctx->inum_ind),&(dctx->inum_len),&(dctx->inum_rcode));
#endif
/* open first cursor */
OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd1), OCI_HTYPE_STMT, 0, (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT1);
OCIStmtPrepare(dctx->curd1, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);
OCIERROR(errhp,
OCIAttrSet(dctx->curd1,OCI_HTYPE_STM,(dvoid*)&dctx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
/* bind variables */
OCIBND(dctx->curd1,
dctx->w_id_bp,errhp,":w_id",ADR(delP->w_id),SIZ(delP->w_id),SQLT_INT);
OCIDFNRA(dctx->curd1, dctx->d_id_dp,errhp,1,dctx->d_id,SIZ(dctx->d_id[0]),SQLT_INT,
dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
OCIDFNRA(dctx->curd1,
dctx->del_o_id_dp,errhp,2,dctx->del_o_id,SIZ(dctx->del_o_id[0]),
SQLT_INT,dctx->del_o_id_ind,dctx->del_o_id_len,
dctx->del_o_id_rcode);
OCIDFNRA(dctx->curd1, dctx->no_rowid_dp,errhp,3,dctx->no_rowid_ptr,
sizeof(dctx->no_rowid_ptr[0]), SQLT_RDD,dctx->no_rowid_ind,
dctx->no_rowid_len, dctx->no_rowid_rcode);
OCIDFNRA(dctx->curd1, dctx->c_id_dp,errhp,4,dctx->c_id,SIZ(dctx->c_id[0]),SQLT_INT,
dctx->c_id_ind,dctx->c_id_len,dctx->c_id_rcode);
OCIDFNRA(dctx->curd1, dctx->o_rowid_dp,errhp,5,dctx->o_rowid_ptr,
sizeof(dctx->o_rowid_ptr[0]), SQLT_RDD,dctx->o_rowid_ind,
dctx->o_rowid_len, dctx->o_rowid_rcode);
/* open test cursor */
/* OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curdtest), OCI_HTYPE_STMT, 0, (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXTTEST);
OCIStmtPrepare(dctx->curdtest, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);
OCIBNDRA(dctx->curdtest,dctx->no_rowid_bp,errhp,":no_rowid",&(dctx->no_rowid_ptr[0])
,
SIZ(dctx->no_rowid_ptr[0]),SQLT_RDD,dctx->no_rowid_ind,
dctx->no_rowid_len,dctx->no_rowid_rcode);
OCIBNDRA(dctx->curdtest,dctx->o_rowid_bp,errhp,":o_rowid",&(dctx->o_rowid_ptr[0]),
SIZ(dctx->o_rowid_ptr[0]),SQLT_RDD,dctx->o_rowid_ind,
dctx->o_rowid_len,dctx->o_rowid_rcode);
*/
/* open second cursor */
OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd2), OCI_HTYPE_STMT, 0, (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT2);
OCIStmtPrepare(dctx->curd2, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
/* bind variables */
OCIBNDRA(dctx->curd2,
dctx->no_rowid_bp,errhp,":no_rowid",&(dctx->no_rowid_ptr[0]),
SIZ(dctx->no_rowid_ptr[0]),SQLT_RDD,dctx->no_rowid_ind,
dctx->no_rowid_len,dctx->no_rowid_rcode);
/*
OCIBNDRA(dctx->curd2, dctx->w_id_bp, errhp, ":w_id", dctx->w_id,SIZ(dctx->w_id[0]),
SQLT_INT,dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
OCIBNDRA(dctx->curd2, dctx->d_id_bp, errhp, ":d_id", dctx->d_id,SIZ(dctx->d_id[0]),
SQLT_INT,dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
OCIBNDRA(dctx->curd2, dctx->del_o_id_bp, errhp, ":o_id", dctx->del_o_id,
SIZ(dctx->del_o_id[0]),
SQLT_INT,dctx->del_o_id_ind,dctx->del_o_id_len,dctx->del_o_id_rcode);
*/
/* open third cursor */
OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd3), OCI_HTYPE_STMT, 0, (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT3);
OCIStmtPrepare(dctx->curd3, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
/* bind variables */
OCIBNDRA(dctx->curd3, dctx->carrier_id_bp,errhp,":carrier_id",dctx->carrier_id,
SIZ(dctx->carrier_id[0]),SQLT_INT,dctx->carrier_id_ind,
dctx->carrier_id_len,dctx->carrier_id_rcode);
/*
OCIBNDRA(dctx->curd3, dctx->w_id_bp, errhp, ":w_id", dctx->w_id,SIZ(dctx->w_id[0]),
SQLT_INT,dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
OCIBNDRA(dctx->curd3, dctx->d_id_bp, errhp, ":d_id", dctx->d_id,SIZ(dctx->d_id[0]),
SQLT_INT,dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
OCIBNDRA(dctx->curd3, dctx->del_o_id_bp, errhp, ":o_id", dctx->del_o_id,
SIZ(dctx->del_o_id[0]),
SQLT_INT,dctx->del_o_id_ind,dctx->del_o_id_len,dctx->del_o_id_rcode);
OCIBNDRA(dctx->curd3, dctx->c_id_bp, errhp, ":c_id", dctx->c_id,SIZ(dctx->c_id[0]),
SQLT_INT,dctx->c_id_ind,dctx->c_id_len,dctx->c_id_rcode);
*/
OCIBNDRA(dctx->curd3, dctx->o_rowid_bp,errhp,":o_rowid",&(dctx->o_rowid_ptr[0]),
SIZ(dctx->o_rowid_ptr[0]),SQLT_RDD,dctx->o_rowid_ind,
dctx->o_rowid_ptr_len,dctx->o_rowid_rcode);
/* open fourth cursor */
OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd4), OCI_HTYPE_STMT, 0, (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT4);
OCIStmtPrepare(dctx->curd4, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
/* bind variables */
OCIBNDRA(dctx->curd4, dctx->w_id_bp,errhp,":w_id",dctx->w_id,SIZ(dctx->w_id[0]),
SQLT_INT,dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
OCIBNDRA(dctx->curd4, dctx->d_id_bp,errhp,":d_id",dctx->d_id,SIZ(dctx->d_id[0]),
SQLT_INT,dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
OCIBNDRA(dctx->curd4, dctx->o_id_bp,errhp,":o_id",dctx->del_o_id,
SIZ(dctx->del_o_id[0]),SQLT_INT,dctx->del_o_id_ind,
dctx->del_o_id_len,dctx->del_o_id_rcode);
OCIBNDRA(dctx->curd4, dctx->cr_date_bp,errhp,":cr_date",
dctx->del_date,DEL_DATE_LEN,
SQLT_DAT,dctx->del_date_ind, dctx->del_date_len,
dctx->del_date_rcode);
/* open fifth cursor */
OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd5), OCI_HTYPE_STMT, 0, (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT5);
OCIStmtPrepare(dctx->curd5, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
OCIERROR(errhp,
OCIAttrSet(dctx->curd5,OCI_HTYPE_STM,(dvoid*)&dctx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
/* bind variables */
OCIBND(dctx->curd5,dctx->w_id_bp,errhp,":w_id",ADR(delP->w_id),SIZ(delP->w_id),SQLT
_INT);
for (i = 0; i < NDISTS; i++) {
sprintf (bstr1, ":d_id%d", i + 1);
sprintf (bstr2, ":o_id%d", i + 1);
OCIBNDRA(dctx->curd5,dctx->bstr1_bp[i],errhp,bstr1,ADR(dctx->d_id[i]),
SIZ(dctx->d_id[0]),SQLT_INT, &(dctx->d_id_ind[i]),
&(dctx->d_id_len[i]),&(dctx->d_id_rcode[i]));
OCIBNDRA(dctx->curd5,dctx->bstr2_bp[i],errhp,bstr2,ADR(dctx->del_o_id[i]),
SIZ(dctx->del_o_id[0]),SQLT_INT, &(dctx->del_o_id_ind[i]),
&(dctx->del_o_id_len[i]),&(dctx->del_o_id_rcode[i]));
}
```

```
OCIDFNRA(dctx->curd5,dctx->cons_dp,errhp,1,dctx->cons,SIZ(dctx->cons[0]),SQLT_INT,
dctx->cons_ind,dctx->cons_len,dctx->cons_rcode);
OCIDFNRA(dctx->curd5,dctx->amt_dp,errhp,2,dctx->amt,SIZ(dctx->amt[0]),SQLT_INT,
dctx->amt_ind,dctx->amt_len,dctx->amt_rcode);
/* open sixth cursor */
OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd6), OCI_HTYPE_STMT, 0, (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT6);
OCIStmtPrepare(dctx->curd6, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
/* bind variables */
OCIBNDRA(dctx->curd6,dctx->amt_bp,errhp,":amt",dctx->amt,SIZ(dctx->amt[0]),SQLT_INT
,
dctx->amt_ind,dctx->amt_len,dctx->amt_rcode);
OCIBNDRA(dctx->curd6,dctx->w_id_bp,errhp,":w_id",dctx->w_id,SIZ(dctx->w_id[0]),
SQLT_INT,dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
OCIBNDRA(dctx->curd6,dctx->d_id_bp,errhp,":d_id",dctx->d_id,SIZ(dctx->d_id[0]),
SQLT_INT,dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
OCIBNDRA(dctx->curd6,dctx->c_id_bp,errhp,":c_id",dctx->c_id,SIZ(dctx->c_id[0]),
SQLT_INT, dctx->c_id_ind,dctx->c_id_len,dctx->c_id_rcode);
return (0);
}
pldel (ora_cn_data_t *ora_SlotDataP)
{
delctx *dctx = ora_SlotDataP->dctx;
global_delivery_t *delP = ora_SlotDataP->delP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
int i, j,v;
int rpc,rcount;
int invalid;
int tmp_id;
/* float tmp_amt; changed form float to int */
int tmp_amt;
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
int hasno;
int reread;
char sdate[30];
/* oexfet (&curd0, 1, 0, 0); */
OCIStmtExecute(tpcsvc,dctx->curd0,errhp,1,0,0,0,OCI_DEFAULT);
sysdate (sdate);
printf ("Delivery started at %s on node %s\n", sdate, dctx->inum);
#endif
retry:
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
reread = 1;
#endif
iso:
invalid = 0;
/* initialization for array operations */
for (i = 0; i < NDISTS; i++) {
dctx->del_o_id_ind[i] = TRUE;
dctx->cons_ind[i] = TRUE;
dctx->w_id_ind[i] = TRUE;
dctx->d_id_ind[i] = TRUE;
dctx->c_id_ind[i] = TRUE;
dctx->del_date_ind[i] = TRUE;
dctx->carrier_id_ind[i] = TRUE;
dctx->amt_ind[i] = TRUE;
dctx->no_rowid_ind[i] = TRUE;
dctx->o_rowid_ind[i] = TRUE;
dctx->del_o_id_len[i] = SIZ(dctx->del_o_id[0]);
dctx->cons_len[i] = SIZ(dctx->cons[0]);
dctx->w_id_len[i] = SIZ(dctx->w_id[0]);
dctx->d_id_len[i] = SIZ(dctx->d_id[0]);
dctx->c_id_len[i] = SIZ(dctx->c_id[0]);
dctx->del_date_len[i] = DEL_DATE_LEN;
dctx->carrier_id_len[i] = SIZ(dctx->carrier_id[0]);
dctx->amt_len[i] = SIZ(dctx->amt[0]);
dctx->no_rowid_len[i] = ROWIDLEN;
dctx->o_rowid_len[i] = ROWIDLEN;
dctx->o_rowid_ptr_len[i] = SIZ(dctx->o_rowid_ptr[0]);
dctx->no_rowid_ptr_len[i] = SIZ(dctx->no_rowid_ptr[0]);
dctx->w_id[i] = delP->w_id;
dctx->carrier_id[i] = delP->o_carrier_id;
memcpy(dctx->del_date[i],delP->cr_date,DEL_DATE_LEN);
}
/* array select from new_order and orders tables */
delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd1,errhp,NDISTS,0,0,0,OCI_DEFAULT)
;
if((delP->execstatus != OCI_SUCCESS) && (delP->execstatus != OCI_NO_DATA)) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
delP->errcode = OCIERROR(errhp,delP->execstatus);
if(delP->errcode == NOT_SERIALIZABLE) {
delP->retries++;
goto retry;
} else if (delP->errcode == RECOVERR) {
delP->retries++;
goto retry;
} else {
return -1;
}
}
/* mark districts with no new order */
OCIAttrGet(dctx->curd1,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);
rpc = rcount;
invalid = NDISTS - rcount;
for (i = rpc; i < NDISTS; i++) {
dctx->del_o_id_ind[i] = NA;
dctx->w_id_ind[i] = NA;
dctx->d_id_ind[i] = NA;
dctx->c_id_ind[i] = NA;
dctx->carrier_id_ind[i] = NA;
dctx->no_rowid_ind[i] = NA;
dctx->o_rowid_ind[i] = NA;
}
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
if (invalid) {
sysdate (sdate);
for (i = 1; i <= NDISTS; i++) {
hasno = 0;
for (j = 0; j < rpc; j++) {
if (dctx->d_id[j] == i) {
hasno = 1;
break;
}
}
if (!hasno)
printf ("Delivery [dist %d] found no new order at %s\n", i, sdate);
}
if (reread) {
sleep (60);
sysdate (sdate);
printf ("Delivery wake up at %s\n", sdate);
reread = 0;
goto iso;
}
}
#endif
/* array delete of new_order table */
delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd2,errhp,rpc,0,0,0,OCI_DEFAULT);
if(delP->execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
delP->errcode = OCIERROR(errhp,delP->execstatus);
if(delP->errcode == NOT_SERIALIZABLE) {
delP->retries++;
goto retry;
} else if (delP->errcode == RECOVERR) {
delP->retries++;
goto retry;
} else {
return -1;
}
}
/* mark districts with no new order */
OCIAttrGet(dctx->curd2,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);
if (rcount != rpc) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (DEL_ERROR);
}
delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd3,errhp,rpc,0,0,0,OCI_DEFAULT);
if (delP->execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
delP->errcode = OCIERROR(errhp,delP->execstatus);
if(delP->errcode == NOT_SERIALIZABLE) {
delP->retries++;
goto retry;
} else if (delP->errcode == RECOVERR) {
delP->retries++;
goto retry;
} else {
return -1;
}
}
OCIAttrGet(dctx->curd3,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);
if (rcount != rpc) {
fprintf (stderr,
"Error in TPC-C server %d: %d rows selected, %d ords updated\n",
proc_no, rpc, rcount);
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-1);
}
/* array update of order_line table */
delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd4,errhp,rpc,0,0,0,OCI_DEFAULT);
if(delP->execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
delP->errcode = OCIERROR(errhp,delP->execstatus);
if(delP->errcode == NOT_SERIALIZABLE) {
delP->retries++;
goto retry;
} else if (delP->errcode == RECOVERR) {
delP->retries++;
goto retry;
} else {
return -1;
}
}
/* array select from order_line table */
delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd5,errhp,rpc,0,0,0,OCI_DEFAULT);
if((delP->execstatus != OCI_SUCCESS) && (delP->execstatus != OCI_NO_DATA)) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
delP->errcode = OCIERROR(errhp,delP->execstatus);
if(delP->errcode == NOT_SERIALIZABLE) {
delP->retries++;
goto retry;
} else if (delP->errcode == RECOVERR) {
delP->retries++;
goto retry;
} else {
return -1;
}
}
OCIAttrGet(dctx->curd5,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);
if (rcount != rpc) {
fprintf (stderr,
"Error in TPC-C server %d: %d rows selected, %d ordl selected\n",
delP->proc_no, rpc, rcount);
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-1);
}
/* reorder amount selected if necessary */
for (i = 0; i < rpc; i++) {
if (dctx->cons[i] != dctx->d_id[i]) {
fprintf (stderr, "TPC-C server %d: reordering amount\n", proc_no);
for (j = i + 1; j < rpc; j++) {
if (dctx->cons[j] == dctx->d_id[i]) {
tmp_id = dctx->cons[i];
```

```
dctx->cons[i] = dctx->cons[j];
dctx->cons[j] = tmp_id;
tmp_amt = dctx->amt[i];
dctx->amt[i] = dctx->amt[j];
dctx->amt[j] = tmp_amt;
break;
}
}
if (j >= rpc) {
fprintf (stderr,
"Error in TPC-C server %d: missing ordl?\n", delP->proc_no);
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-1);
}
}
}
#if defined(ISO5) || defined(ISO6)
printf ("d_id:amount\n");
for (i = 0; i < rpc; i++)
printf ("%d:%.2f ", dctx->d_id[i], (float)dctx->amt[i]/100);
printf ("\n");
#endif
/* array update of customer table */
#if defined(ISO5) || defined (ISO6)
delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,0,0,
OCI_DEFAULT);
#else
delP->execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,0,0,
OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
#endif
if(delP->execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
delP->errcode = OCIERROR(errhp,delP->execstatus);
if(delP->errcode == NOT_SERIALIZABLE) {
delP->retries++;
goto retry;
} else if (delP->errcode == RECOVERR) {
delP->retries++;
goto retry;
} else {
return -1;
}
}
OCIAttrGet(dctx->curd6,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROWCNT,errhp);
if (rcount != rpc) {
fprintf (stderr,
"Error in TPC-C server %d: %d rows selected, %d cust updated\n",
delP->proc_no, rpc, rcount);
OCITransRollback(tpcsvc, errhp, OCI_DEFAULT);
return (-1);
}
#if defined(ISO5) || defined(ISO6)
sysdate (sdate);
#ifdef ISO5
printf ("Delivery sleep before commit at %s\n", sdate);
#else
printf ("Delivery sleep before abort at %s\n", sdate);
#endif
sleep (60);
sysdate (sdate);
printf ("Delivery wake up at %s\n", sdate);
#endif
#ifdef ISO6
printf("Delivery ISO6 Rolling back.\n");
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
#endif
#ifdef ISO5
OCITransCommit(tpcsvc, errhp, OCI_DEFAULT);
#endif
#if defined(ISO5) || defined(ISO6)
sysdate (sdate);
printf ("Delivery completed at: %s\n", sdate);
#endif
/* return o_id's in district id order */
for (i = 0; i < NDISTS; i++)
delP->del_o_id[i] = 0;
for (i = 0; i < rpc; i++)
delP->del_o_id[dctx->d_id[i] - 1] = dctx->del_o_id[i];
return (0);
}
void pldeldone (ora_cn_data_t *ora_SlotDataP)
{
delctx *dctx = ora_SlotDataP->dctx;
global_delivery_t *delP = ora_SlotDataP->delP;
if (dctx)
{
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
OCIHandleFree((dvoid *)dctx->curd0,OCI_HTYPE_STMT);
#endif
OCIHandleFree((dvoid *)dctx->curd1,OCI_HTYPE_STMT);
OCIHandleFree((dvoid *)dctx->curd2,OCI_HTYPE_STMT);
OCIHandleFree((dvoid *)dctx->curd3,OCI_HTYPE_STMT);
OCIHandleFree((dvoid *)dctx->curd4,OCI_HTYPE_STMT);
OCIHandleFree((dvoid *)dctx->curd5,OCI_HTYPE_STMT);
OCIHandleFree((dvoid *)dctx->curd6,OCI_HTYPE_STMT);
free (dctx);
ora_SlotDataP->dctx = NULL;
}
if (delP) {
free(delP);
ora_SlotDataP->delP = NULL;
}
}
```

pldel.h

```
#ifndef TPCC_PLDEL_H
#define TPCC_PLDEL_H
#define NDISTS 10
#define ROWIDLEN 20
struct delctx {
sb2 del_o_id_ind[NDISTS];
sb2 cons_ind[NDISTS];
sb2 w_id_ind[NDISTS];
sb2 d_id_ind[NDISTS];
sb2 c_id_ind[NDISTS];
sb2 del_date_ind[NDISTS];
sb2 carrier_id_ind[NDISTS];
sb2 amt_ind[NDISTS];
sb2 no_rowid_ind[NDISTS];
sb2 o_rowid_ind[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
sb2 inum_ind;
#endif
ub2 del_o_id_len[NDISTS];
ub2 cons_len[NDISTS];
ub2 w_id_len[NDISTS];
ub2 d_id_len[NDISTS];
ub2 c_id_len[NDISTS];
ub2 del_date_len[NDISTS];
ub2 carrier_id_len[NDISTS];
ub2 amt_len[NDISTS];
ub2 no_rowid_len[NDISTS];
ub2 no_rowid_ptr_len[NDISTS];
ub2 o_rowid_len[NDISTS];
ub2 o_rowid_ptr_len[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
ub2 inum_len;
#endif
ub2 del_o_id_rcode[NDISTS];
ub2 cons_rcode[NDISTS];
ub2 w_id_rcode[NDISTS];
ub2 d_id_rcode[NDISTS];
ub2 c_id_rcode[NDISTS];
ub2 del_date_rcode[NDISTS];
ub2 carrier_id_rcode[NDISTS];
ub2 amt_rcode[NDISTS];
ub2 no_rowid_rcode[NDISTS];
ub2 o_rowid_rcode[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
ub2 inum_rcode;
#endif
int del_o_id[NDISTS];
int cons[NDISTS];
int w_id[NDISTS];
int d_id[NDISTS];
int c_id[NDISTS];
int carrier_id[NDISTS];
/* float amt[NDISTS]; Changed to int */
int amt[NDISTS];
/*
OCIRowid *no_rowid[NDISTS]; */
OCIRowid *no_rowid_ptr[NDISTS];
/*
OCIRowid *o_rowid[NDISTS]; */
OCIRowid *o_rowid_ptr[NDISTS];
unsigned char del_date[NDISTS][DEL_DATE_LEN];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
char inum[10];
#endif
OCIStmt *curd0;
OCIStmt *curd1;
OCIStmt *curd2;
OCIStmt *curd3;
OCIStmt *curd4;
OCIStmt *curd5;
OCIStmt *curd6;
OCIStmt *curdtest;
OCIBind *w_id_bp;
OCIBind *d_id_bp;
OCIBind *o_id_bp;
OCIBind *cr_date_bp;
OCIBind *c_id_bp;
OCIBind *no_rowid_bp;
OCIBind *carrier_id_bp;
OCIBind *o_rowid_bp;
OCIBind *del_o_id_bp;
OCIBind *amt_bp;
OCIBind *bstr1_bp[10];
OCIBind *bstr2_bp[10];
OCIDefine *inum_dp;
OCIDefine *d_id_dp;
OCIDefine *del_o_id_dp;
OCIDefine *no_rowid_dp;
OCIDefine *c_id_dp;
OCIDefine *o_rowid_dp;
OCIDefine *cons_dp;
OCIDefine *amt_dp;
int norow;
};
typedef struct delctx delctx;
struct global_delivery_t {
int w_id;
int o_carrier_id;
int retries;
int del_o_id[10];
int errcode;
int execstatus;
int proc_no;
unsigned char cr_date[7];
};
typedef struct global_delivery_t global_delivery_t;
#endif /* TPCCPLDEL_H */
```

plnew.c

```
#ifdef RCSID
static char *RCSid =
"$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/plnew.c,v 1.2 1998/01/23 15:08:07 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */
/*+=================================================================+
| Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=================================================================+
| FILENAME
| plnew.c
| DESCRIPTION
| OCI version (using PL/SQL stored procedure) of
| NEW ORDER transaction in TPC-C benchmark.
+=================================================================*/
#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"
extern void err_printf(char *format, ...);
#ifdef OPS
#define SQLTXT2 "UPDATE stock SET s_order_cnt = s_order_cnt + 1, \
s_ytd = s_ytd + :ol_quantity, s_remote_cnt = s_remote_cnt + :s_remote, \
s_quantity = s_quantity - :ol_quantity + \
DECODE (SIGN (s_quantity - :ol_quantity - 10), -1, 91, 0) \
WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id"
#else
#define SQLTXT2 "UPDATE stock SET s_order_cnt = s_order_cnt + 1, \
s_ytd = s_ytd + :ol_quantity, s_remote_cnt = s_remote_cnt + :s_remote, \
s_quantity = :s_quantity \
WHERE rowid = :s_rowid"
#endif
#define SQLTXT3 "\
SELECT 0,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :10 AND s_w_id = :30 AND s_i_id = i_id UNION ALL \
SELECT 1,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :11 AND s_w_id = :31 AND s_i_id = i_id UNION ALL \
SELECT 2,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :12 AND s_w_id = :32 AND s_i_id = i_id UNION ALL \
SELECT 3,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :13 AND s_w_id = :33 AND s_i_id = i_id UNION ALL \
SELECT 4,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :14 AND s_w_id = :34 AND s_i_id = i_id UNION ALL \
SELECT 5,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :15 AND s_w_id = :35 AND s_i_id = i_id UNION ALL \
SELECT 6,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :16 AND s_w_id = :36 AND s_i_id = i_id UNION ALL \
SELECT 7,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :17 AND s_w_id = :37 AND s_i_id = i_id UNION ALL \
SELECT 8,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :18 AND s_w_id = :38 AND s_i_id = i_id UNION ALL \
SELECT 9,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :19 AND s_w_id = :39 AND s_i_id = i_id UNION ALL \
SELECT 10,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :20 AND s_w_id = :40 AND s_i_id = i_id UNION ALL \
SELECT 11,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :21 AND s_w_id = :41 AND s_i_id = i_id UNION ALL \
SELECT 12,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :22 AND s_w_id = :42 AND s_i_id = i_id UNION ALL \
SELECT 13,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :23 AND s_w_id = :43 AND s_i_id = i_id UNION ALL \
SELECT 14,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :24 AND s_w_id = :44 AND s_i_id = i_id"
#define SQLTXT4 "INSERT INTO order_line \
(ol_o_id, ol_d_id, ol_w_id, oL_number, ol_delivery_d, ol_i_id, \
ol_supply_w_id, ol_quantity,ol_amount,ol_dist_info)\
VALUES (:ol_o_id, :ol_d_id, \
:ol_w_id, :ol_number, :null_date, :ol_i_id, :ol_supply_w_id, :ol_quantity, \
:ol_amount, :ol_dist_info)"
static int SelItemStk (newctx *nctx,
global_newOrder_t *newP,
OCISvcCtx *tpcsvc,
OCIError *errhp);
static int UpdStk2 (newctx *nctx,
global_newOrder_t *newP,
OCISvcCtx *tpcsvc,
OCIError *errhp);
#define ROWIDLEN 20
#define OCIROWLEN 20
/*
* plnewinit: called once per thread to initialize the new order
* specific thread global data structures.
*/
plnewinit (ora_cn_data_t *ora_SlotDataP)
{
int i, j;
char *ora_home = getenv("ORACLE_HOME");
char sql_file_name[256];
text stmbuf[SQL_BUF_SIZE];
char id[4];
char sd[4];
newctx *nctx;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
global_newOrder_t *newP;
ora_SlotDataP->nctx = (newctx *) malloc (sizeof(newctx));
nctx = ora_SlotDataP->nctx;
memset(nctx,(char)0,sizeof(newctx));
ora_SlotDataP->globals = (global_newOrder_t *)malloc(sizeof(global_newOrder_t)
);
newP = ora_SlotDataP->globals;
memset(ora_SlotDataP->globals,(char)0,sizeof(global_newOrder_t));
nctx->cs = 1;
nctx->norow=0;
for(i=0;i<NITEMS;i++) {
OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&nctx->s_rowid_ptr[i],
OCI_DTYPE_ROWID,0,(dvoid**)0));
}
nctx->w_id_ind = TRUE;
nctx->w_id_len = sizeof(newP->w_id);
nctx->d_id_ind = TRUE;
nctx->d_id_len = sizeof(newP->d_id);
nctx->c_id_ind = TRUE;
nctx->c_id_len = sizeof(newP->c_id);
nctx->o_all_local_ind = TRUE;
nctx->o_all_local_len = sizeof(newP->o_all_local);
nctx->o_ol_cnt_ind = TRUE;
nctx->o_ol_cnt_len = sizeof(newP->o_ol_cnt);
nctx->w_tax_ind = TRUE;
nctx->w_tax_len = 0;
nctx->d_tax_ind = TRUE;
nctx->d_tax_len = 0;
nctx->o_id_ind = TRUE;
nctx->o_id_len = sizeof(newP->o_id);
nctx->c_discount_ind = TRUE;
nctx->c_discount_len = 0;
nctx->c_credit_ind = TRUE;
nctx->c_credit_len = 0;
/* nctx->c_credit_len = 0; */
nctx->c_last_ind = TRUE;
nctx->c_last_len = 0;
nctx->retries_ind = TRUE;
nctx->retries_len = sizeof(newP->retries);
nctx->cr_date_ind = TRUE;
nctx->cr_date_len = sizeof(newP->cr_date);
/* open first cursor */
OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&nctx->curn1),
OCI_HTYPE_STMT, 0, (dvoid**)0));
if (!ora_home) {
err_printf("Cannot find env variable ORACLE_HOME\n");
exit(13);
}
sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/new.sql", ora_home);
sqlfile(sql_file_name,stmbuf);
/* sqlfile("../blocks/new.sql",stmbuf); */
OCIERROR(errhp,OCIStmtPrepare(nctx->curn1, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT));
/* bind variables */
OCIBNDR(nctx->curn1, nctx->w_id_bp, errhp,
":w_id",ADR(newP->w_id),SIZ(newP->w_id),
SQLT_INT, &nctx->w_id_ind, &nctx->w_id_len, &nctx->w_id_rc);
OCIBNDR(nctx->curn1, nctx->d_id_bp, errhp, ":d_id",ADR(newP->d_id),SIZ(newP->d_id),
SQLT_INT, &nctx->d_id_ind, &nctx->d_id_len, &nctx->d_id_rc);
OCIBNDR(nctx->curn1, nctx->c_id_bp, errhp, ":c_id",ADR(newP->c_id),SIZ(newP->c_id),
SQLT_INT, &nctx->c_id_ind, &nctx->c_id_len, &nctx->c_id_rc);
OCIBNDR(nctx->curn1, nctx->o_all_local_bp, errhp, ":o_all_local",
ADR(newP->o_all_local), SIZ(newP->o_all_local),SQLT_INT, &nctx->o_all_local_ind,
&nctx->o_all_local_len, &nctx->o_all_local_rc);
OCIBNDR(nctx->curn1, nctx->o_all_cnt_bp, errhp, ":o_ol_cnt",ADR(newP->o_ol_cnt),
SIZ(newP->o_ol_cnt),SQLT_INT,
&nctx->o_ol_cnt_ind, &nctx->o_ol_cnt_len, &nctx->o_ol_cnt_rc);
OCIBNDR(nctx->curn1, nctx->w_tax_bp, errhp,
":w_tax",ADR(newP->w_tax),SIZ(newP->w_tax),
SQLT_INT, &nctx->w_tax_ind, &nctx->w_tax_len, &nctx->w_tax_rc);
OCIBNDR(nctx->curn1, nctx->d_tax_bp, errhp,
":d_tax",ADR(newP->d_tax),SIZ(newP->d_tax),
SQLT_INT, &nctx->d_tax_ind, &nctx->d_tax_len, &nctx->d_tax_rc);
OCIBNDR(nctx->curn1, nctx->o_id_bp, errhp, ":o_id",ADR(newP->o_id),SIZ(newP->o_id),
SQLT_INT, &nctx->o_id_ind, &nctx->o_id_len, &nctx->o_id_rc);
OCIBNDR(nctx->curn1, nctx->c_discount_bp, errhp, ":c_discount",
ADR(newP->c_discount), SIZ(newP->c_discount),SQLT_INT,
&nctx->c_discount_ind, &nctx->c_discount_len, &nctx->c_discount_rc);
OCIBNDR(nctx->curn1, nctx->c_credit_bp, errhp, ":c_credit",newP->c_credit,
SIZ(newP->c_credit),SQLT_CHR,
&nctx->c_credit_ind, &nctx->c_credit_len, &nctx->c_credit_rc);
OCIBNDR(nctx->curn1, nctx->c_last_bp, errhp, ":c_last",newP->c_last,SIZ(newP->c_last),
SQLT_STR, &nctx->c_last_ind, &nctx->c_last_len, &nctx->c_last_rc);
OCIBNDR(nctx->curn1, nctx->retries_bp, errhp, ":retry",ADR(newP->retries),
SIZ(newP->retries),SQLT_INT,
&nctx->retries_ind, &nctx->retries_len, &nctx->retries_rc);
OCIBNDR(nctx->curn1, nctx->cr_date_bp, errhp,
":cr_date",newP->cr_date,SIZ(newP->cr_date), SQLT_DAT, &nctx->cr_date_ind,
&nctx->cr_date_len, &nctx->cr_date_rc);
/* open second cursor */
OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&nctx->curn2), OCI_HTYPE_STMT,
0, (dvoid**)0));
sprintf ((char *) stmbuf, SQLTXT2);
OCIERROR(errhp,OCIStmtPrepare(nctx->curn2, errhp, stmbuf,
strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
/* bind variables */
#ifdef OPS
OCIBNDRA(nctx->curn2, nctx->ol_i_id_bp,errhp,":ol_i_id",newP->nol_i_id,
SIZ(int), SQLT_INT, nctx->nol_i_id_ind,nctx->nol_i_id_len,
nctx->nol_i_id_rcode);
OCIBNDRA(nctx->curn2, nctx->ol_supply_w_id_bp, errhp, ":ol_supply_w_id",
newP->nol_supply_w_id,SIZ(int),SQLT_INT, nctx->nol_supply_w_id_ind,
nctx->nol_supply_w_id_len, nctx->nol_supply_w_id_rcode);
#else
OCIBNDRA(nctx->curn2, nctx->s_quantity_bp,errhp,":s_quantity",newP->s_quantity,
SIZ(int), SQLT_INT,nctx->s_quant_ind,nctx->s_quant_len,
nctx->s_quant_rcode);
OCIBNDRA(nctx->curn2, nctx->s_rowid_bp, errhp, ":s_rowid",nctx->s_rowid_ptr,
sizeof(nctx->s_rowid_ptr[0]),SQLT_RDD,nctx->s_rowid_ind,
nctx->s_rowid_len,nctx->s_rowid_rcode);
#endif
OCIBNDRA(nctx->curn2, nctx->ol_quantity_bp,errhp,":ol_quantity",newP->nol_quantity,
SIZ(int),SQLT_INT,nctx->nol_quantity_ind,nctx->nol_quantity_len,
nctx->nol_quantity_rcode);
OCIBNDRA(nctx->curn2, nctx->s_remote_bp, errhp, ":s_remote",nctx->s_remote,
SIZ(int), SQLT_INT,nctx->s_remote_ind,nctx->s_remote_len,
nctx->s_remote_rcode);
/* open third cursor and bind variables */
for (i = 0; i < 10; i++)
{
j = i + 1;
```

```
OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)&((nctx->curn3)[i]),
OCI_HTYPE_STMT, 0, (dvoid**)0));
sprintf ((char *) stmbuf, SQLTXT3, j, j, j, j, j, j, j, j, j, j, j,
j, j, j);
OCIERROR(errhp,OCIStmtPrepare((nctx->curn3)[i], errhp, stmbuf,
strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT));
OCIERROR(errhp,
OCIAttrSet(nctx->curn3[i],OCI_HTYPE_STM,(dvoid*)&nctx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
for (j = 0; j < NITEMS; j++)
{
sprintf (id, ":%d", j + 10);
sprintf (sd, ":%d", j + 30);
OCIBNDRA((nctx->curn3)[i],(nctx->id_bp)[i][j],errhp,id,ADR(newP->nol_i_id[j]),
SIZ(int),SQLT_INT,
&nctx->nol_i_id_ind[j],&nctx->nol_i_id_len[j],
&nctx->nol_i_id_rcode[j]);
OCIBNDRA((nctx->curn3)[i],(nctx->sd_bp)[i][j],errhp,sd,
ADR(newP->nol_supply_w_id[j]),SIZ(int),SQLT_INT,
&nctx->nol_supply_w_id_ind[j],&nctx->nol_supply_w_id_len[j],
&nctx->nol_supply_w_id_rcode[j]);
nctx->nol_i_id_ind[j] = NA;
nctx->nol_supply_w_id_ind[j] = NA;
nctx->nol_i_id_len[j] = sizeof(int);
nctx->nol_supply_w_id_len[j] = sizeof(int);
}
OCIDFNRA((nctx->curn3)[i],(nctx->Dcons)[i],errhp,1,&(nctx->cons[0]),
SIZ(nctx->cons[0]),SQLT_INT,
nctx->cons_ind,nctx->cons_len, nctx->cons_rcode);
OCIDFNRA((nctx->curn3)[i], (nctx->Ds_rowid)[i],errhp,2,
nctx->s_rowid_ptr,
sizeof(nctx->s_rowid_ptr[0]),
SQLT_RDD,nctx->s_rowid_ind,nctx->s_rowid_len,
nctx->s_rowid_rcode);
OCIDFNRA((nctx->curn3)[i], (nctx->Di_price)[i],errhp,3,newP->i_price,SIZ(int),
SQLT_INT, nctx->i_price_ind,nctx->i_price_len,nctx->i_price_rcode);
OCIDFNRA((nctx->curn3)[i], (nctx->Di_name)[i],errhp,4,newP->i_name,
SIZ(newP->i_name[0]),SQLT_STR, nctx->i_name_ind,nctx->i_name_len,
nctx->i_name_rcode);
OCIDFNRA((nctx->curn3)[i], (nctx->Di_data)[i],errhp,5,nctx->i_data,
SIZ(nctx->i_data[0]),
SQLT_STR,nctx->i_data_ind,nctx->i_data_len,nctx->i_data_rcode);
OCIDFNRA((nctx->curn3)[i], (nctx->Ds_dist_info)[i],errhp,6,
nctx->s_dist_info, SIZ(nctx->s_dist_info[0]),SQLT_STR,
nctx->s_dist_info_ind, nctx->s_dist_info_len,
nctx->s_dist_info_rcode);
OCIDFNRA((nctx->curn3)[i],(nctx->Ds_data)[i],errhp,7,nctx->s_data,
SIZ(nctx->s_data[0]),SQLT_STR,nctx->s_data_ind,
nctx->s_data_len,nctx->s_data_rcode);
OCIDFNRA((nctx->curn3)[i],(nctx->Ds_quantity)[i],errhp,8,newP->s_quantity,
SIZ(int),SQLT_INT, nctx->s_quantity_ind,nctx->s_quantity_len,
nctx->s_quantity_rcode);
}
/* open fourth cursor */
OCIHandleAlloc(tpcenv, (dvoid **)(&nctx->curn4), OCI_HTYPE_STMT, 0,
(dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT4);
OCIStmtPrepare(nctx->curn4, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
/* bind variables */
OCIBNDRA(nctx->curn4, nctx->ol_o_id_bp,errhp,":ol_o_id",nctx->ol_o_id,
SIZ(int),SQLT_INT, nctx->ol_o_id_ind,nctx->ol_o_id_len,
nctx->ol_o_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_d_id_bp,errhp,":ol_d_id",nctx->ol_d_id,
SIZ(int),SQLT_INT, nctx->ol_d_id_ind,nctx->ol_d_id_len,
nctx->ol_d_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_w_id_bp,errhp,":ol_w_id",nctx->ol_w_id,
SIZ(int),SQLT_INT, nctx->ol_w_id_ind,nctx->ol_w_id_len,
nctx->ol_w_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_number_bp,errhp,":ol_number",nctx->ol_number,
SIZ(int),SQLT_INT, nctx->ol_number_ind,nctx->ol_number_len,
nctx->ol_number_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_i_id_bp,errhp,":ol_i_id",newP->nol_i_id,SIZ(int),
SQLT_INT, nctx->nol_i_id_ind,nctx->nol_i_id_len,
nctx->nol_i_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_supply_w_id_bp,errhp,":ol_supply_w_id",
newP->nol_supply_w_id,SIZ(int),SQLT_INT, nctx->nol_supply_w_id_ind,
nctx->nol_supply_w_id_len, nctx->nol_supply_w_id_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_quantity_bp,errhp,":ol_quantity",newP->nol_quantity,
SIZ(int),SQLT_INT, nctx->nol_quantity_ind,nctx->nol_quantity_len,
nctx->nol_quantity_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_amount_bp,errhp,":ol_amount",newP->nol_amount,
SIZ(int),SQLT_INT, nctx->nol_amount_ind,nctx->nol_amount_len,
nctx->nol_amount_rcode);
OCIBNDRA(nctx->curn4, nctx->ol_dist_info_bp,errhp,":ol_dist_info",
nctx->s_dist_info, SIZ(nctx->s_dist_info[0]),SQLT_AFC,
nctx->ol_dist_info_ind, nctx->ol_dist_info_len,
nctx->ol_dist_info_rcode);
OCIBNDRA(nctx->curn4, nctx->null_date_bp,errhp,":null_date",nctx->null_date,
SIZ(nctx->null_date[0]), SQLT_DAT,nctx->null_date_ind,
nctx->null_date_len, nctx->null_date_rc);
/* set up the null date Null date is 15-sep-11 */
for (i=0;i<NITEMS;i++)
{
nctx->null_date[i][0] = 118;
nctx->null_date[i][1] = 111;
nctx->null_date[i][2] = 1;
nctx->null_date[i][3] = 1;
nctx->null_date[i][4] = 1;
nctx->null_date[i][5] = 1;
nctx->null_date[i][6] = 1;
}
return (0);
}
plnew (ora_cn_data_t *ora_SlotDataP)
{
int i, j, k;
int rpc, rpc3, rowoff, iters,rcount;
ub4 flags;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
global_newOrder_t *newP = ora_SlotDataP->globals;
newctx *nctx = ora_SlotDataP->nctx;
#if defined(ISO1) || defined(ISO7)
int reread;
char sdate[30];
sysdate (sdate);
printf ("New Order started at: %s\n", sdate);
#endif
retry:
#ifdef ISO7
reread = 1;
#endif
newP->status = 0; /* number of invalid items */
/* get number of order lines, and check if all are local */
newP->o_ol_cnt = NITEMS;
newP->o_all_local = 1;
for (i = 0; i < NITEMS; i++) {
if (newP->nol_i_id[i] == 0) {
newP->o_ol_cnt = i;
break;
}
if (newP->nol_supply_w_id[i] != newP->w_id) {
nctx->s_remote[i] = 1;
newP->o_all_local = 0;
}
else
nctx->s_remote[i] = 0;
}
nctx->w_id_ind = TRUE;
nctx->w_id_len = sizeof(newP->w_id);
nctx->d_id_ind = TRUE;
nctx->d_id_len = sizeof(newP->d_id);
nctx->c_id_ind = TRUE;
nctx->c_id_len = sizeof(newP->c_id);
nctx->o_all_local_ind = TRUE;
nctx->o_all_local_len = sizeof(newP->o_all_local);
nctx->o_ol_cnt_ind = TRUE;
nctx->o_ol_cnt_len = sizeof(newP->o_ol_cnt);
nctx->w_tax_ind = TRUE;
nctx->w_tax_len = 0;
nctx->d_tax_ind = TRUE;
nctx->d_tax_len = 0;
nctx->o_id_ind = TRUE;
nctx->o_id_len = sizeof(newP->o_id);
nctx->c_discount_ind = TRUE;
nctx->c_discount_len = 0;
nctx->c_credit_ind = TRUE;
nctx->c_credit_len = 0;
nctx->c_last_ind = TRUE;
nctx->c_last_len = 0;
nctx->retries_ind = TRUE;
nctx->retries_len = sizeof(newP->retries);
nctx->cr_date_ind = TRUE;
nctx->cr_date_len = sizeof(newP->cr_date);
newP->execstatus = OCIStmtExecute(tpcsvc,nctx->curn1,errhp,1,0,0,0,OCI_DEFAULT);
if(newP->execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
newP->errcode = OCIERROR(errhp,newP->execstatus);
if(newP->errcode == NOT_SERIALIZABLE) {
newP->retries++;
goto retry;
} else if (newP->errcode == RECOVERR) {
newP->retries++;
goto retry;
} else {
return -1;
}
}
/* initialization for array operations */
for (i = 0; i < newP->o_ol_cnt; i++) {
nctx->ol_w_id[i] = newP->w_id;
nctx->ol_d_id[i] = newP->d_id;
nctx->ol_number[i] = i + 1;
nctx->null_date_ind[i]= TRUE;
nctx->nol_i_id_ind[i] = TRUE;
nctx->nol_supply_w_id_ind[i] = TRUE;
nctx->nol_quantity_ind[i] = TRUE;
nctx->nol_amount_ind[i] = TRUE;
nctx->ol_w_id_ind[i] = TRUE;
nctx->ol_d_id_ind[i] = TRUE;
nctx->ol_o_id_ind[i] = TRUE;
nctx->ol_number_ind[i] = TRUE;
nctx->ol_dist_info_ind[i] = TRUE;
nctx->s_remote_ind[i] = TRUE;
nctx->s_quant_ind[i] = TRUE;
nctx->cons_ind[i] = TRUE;
nctx->s_rowid_ind[i] = TRUE;
nctx->nol_i_id_len[i] = sizeof(int);
nctx->nol_supply_w_id_len[i] = sizeof(int);
nctx->nol_quantity_len[i] = sizeof(int);
nctx->nol_amount_len[i] = sizeof(int);
nctx->ol_w_id_len[i] = sizeof(int);
nctx->ol_d_id_len[i] = sizeof(int);
nctx->ol_o_id_len[i] = sizeof(int);
nctx->ol_number_len[i] = sizeof(int);
/* nctx->ol_dist_info_len[i] = sizeof(nctx->s_dist_info[0]);*/
nctx->ol_dist_info_len[i] = nctx->s_dist_info_len[i];
nctx->null_date_len[i]=sizeof(nctx->null_date[0]);
nctx->s_remote_len[i] = sizeof(int);
nctx->s_quant_len[i] = sizeof(int);
nctx->s_rowid_len[i] = sizeof(nctx->s_rowid_ptr[0]);
nctx->cons_len[i] = sizeof(int);
}
for (i = newP->o_ol_cnt; i < NITEMS; i++) {
```

```
nctx->nol_i_id_ind[i] = NA;
nctx->nol_supply_w_id_ind[i] = NA;
nctx->nol_quantity_ind[i] = NA;
nctx->nol_amount_ind[i] = NA;
nctx->ol_w_id_ind[i] = NA;
nctx->ol_d_id_ind[i] = NA;
nctx->ol_o_id_ind[i] = NA;
nctx->ol_number_ind[i] = NA;
nctx->ol_dist_info_ind[i] = NA;
nctx->null_date_ind[i]= NA;
nctx->s_remote_ind[i] = NA;
nctx->s_quant_ind[i] = NA;
nctx->cons_ind[i] = NA;
nctx->s_rowid_ind[i] = NA;
nctx->nol_i_id_len[i] = 0;
nctx->nol_supply_w_id_len[i] = 0;
nctx->nol_quantity_len[i] = 0;
nctx->nol_amount_len[i] = 0;
nctx->ol_w_id_len[i] = 0;
nctx->ol_d_id_len[i] = 0;
nctx->ol_o_id_len[i] = 0;
nctx->ol_number_len[i] = 0;
nctx->ol_dist_info_len[i] = 0;
nctx->null_date_len[i]= 0;
nctx->s_remote_len[i] = 0;
nctx->s_quant_len[i] = 0;
nctx->s_rowid_len[i] = 0;
nctx->cons_len[i] = 0;
}
#ifdef OPS
rpc = UpdStk ();
if (rpc == -2)
goto retry;
else if (rpc == -1)
return (-1);
#endif
#ifdef ISO7
iso7:
#endif
rpc3 = SelItemStk (nctx, newP, tpcsvc, errhp);
if (rpc3 == -2)
goto retry;
else if (rpc3 == -1)
return (-1);
#ifdef ISO7
sysdate (sdate);
printf ("Item table read at: %s\n", sdate);
for (i = 0; i < newP->o_ol_cnt; i++) {
if (nctx->nol_i_id_ind[i] != NA)
printf (" i_id = %d, i_price = %d\n", newP->nol_i_id[i], newP->i_price[i]);
}
if (reread) {
sleep (30);
reread = 0;
status = 0;
goto iso7;
}
#endif
/* compute order line amounts, total amount and stock quantities */
newP->total_amount = 0.0;
for (i = 0; i < newP->o_ol_cnt; i++) {
nctx->ol_o_id[i] = newP->o_id;
if (nctx->nol_i_id_ind[i] != NA) {
#ifndef OPS
newP->s_quantity[i] -= newP->nol_quantity[i];
if (newP->s_quantity[i] < 10)
newP->s_quantity[i] += 91;
#endif
newP->nol_amount[i] = (newP->nol_quantity[i] * newP->i_price[i]);
newP->total_amount += newP->nol_amount[i];
if (strstr (nctx->i_data[i], "ORIGINAL") &&
strstr (nctx->s_data[i], "ORIGINAL"))
newP->brand_gen[i] = 'B';
else
newP->brand_gen[i] = 'G';
}
}
newP->total_amount *= ((float)(10000 - newP->c_discount)/10000) * (1.0 +
((float)(newP->d_tax)/10000) + ((float)(newP->w_tax)/10000));
newP->total_amount = newP->total_amount/100;
#ifndef OPS
rpc = UpdStk2 (nctx, newP, tpcsvc, errhp);
if (rpc == -2)
goto retry;
else if (rpc == -1)
return (-1);
#endif
/* number of items selected != number of stock updated */
if (rpc3 != rpc) {
fprintf (stderr, "Error in TPC-C server %d: %d rows of item read, ",
newP->proc_no, rpc3);
fprintf (stderr, " but %d rows of stock update\n", rpc);
/* rollback */
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-1);
}
/* array insert into order line table */
#ifdef ISO1
flags= OCI_DEFAULT;
#else
flags= (newP->status ? OCI_DEFAULT : (OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS));
#endif
if ((newP->o_ol_cnt - newP->status) > 0)
{
newP->execstatus = OCIStmtExecute(tpcsvc,nctx->curn4,errhp,newP->o_ol_cnt -newP->
status,
0,0,0,flags);
if(newP->execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
newP->errcode = OCIERROR(errhp,newP->execstatus);
if(newP->errcode == NOT_SERIALIZABLE) {
newP->retries++;
goto retry;
} else if (newP->errcode == RECOVERR) {
newP->retries++;
goto retry;
} else {
return -1;
}
}
OCIAttrGet(nctx->curn4,OCI_HTYPE_STMT,&rcount,NULL,
OCI_ATTR_ROWCNT, errhp);
if (rcount != (newP->o_ol_cnt - newP->status))
{
fprintf (stderr, "Error in TPC-C server %d: array insert failed\n",
newP->proc_no);
/* rollback */
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-1);
}
}
#ifdef ISO1
sysdate (sdate);
printf ("Sleep before commit/rollback at: %s\n", sdate);
sleep (30);
sysdate (sdate);
printf ("Wake up after sleep at: %s\n", sdate);
#endif
/* commit if no invalid item */
if (newP->status) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
}
#ifdef ISO1
else {
OCITransCommit(tpcsvc, errhp, OCI_DEFAULT);
}
#endif
#if defined(ISO1) || defined(ISO7)
sysdate (sdate);
printf ("New Order completed at: %s\n", sdate);
#endif
return (0);
}
void plnewdone (ora_cn_data_t *ora_SlotDataP)
{
int i;
newctx *nctx;
global_newOrder_t *newP;
if (nctx = ora_SlotDataP->nctx)
{
err_printf("free_handles> OCIHandleFree curn s\n");
OCIHandleFree((dvoid *)nctx->curn1,OCI_HTYPE_STMT);
OCIHandleFree((dvoid *)nctx->curn2,OCI_HTYPE_STMT);
for (i = 0; i < 10; i++)
OCIHandleFree((dvoid *)(nctx->curn3)[i],OCI_HTYPE_STMT);
OCIHandleFree((dvoid *)nctx->curn4,OCI_HTYPE_STMT);
err_printf("free_handles> free nctx (0x%x)\n", nctx);
free (nctx);
ora_SlotDataP->nctx = NULL;
}
if (newP = ora_SlotDataP->globals) {
err_printf("free_handles> newP: 0x%x\n", newP);
free(newP);
ora_SlotDataP->globals = NULL;
}
}
/* the arrays are initialized based on a successful select from */
/* stock/item. We need to shift the values in the orderline array */
/* one position up to compensate when we have an invalid item */
shiftitemstock (i, j, nctx, newP)
int i, j;
newctx *nctx;
global_newOrder_t *newP;
{
/* shift up the values for the stock table */
nctx->s_remote[i] = nctx->s_remote[j];
/* shift up the order_line values */
nctx->nol_i_id_ind[i]=nctx->nol_i_id_ind[j];
newP->nol_i_id[i] = newP->nol_i_id[j];
nctx->nol_quantity_ind[i] = nctx->nol_quantity_ind[j];
newP->nol_quantity[i] = newP->nol_quantity[j];
nctx->nol_supply_w_id_ind [i] = nctx->nol_supply_w_id_ind[j];
newP->nol_supply_w_id[i] = newP->nol_supply_w_id[j];
}
#if 0
/* TODO - this routine is not ever called. So, no changes for now */
swapitemstock (i, j)
int i, j;
{
int k;
int tempi;
int tempf;
char tempstr[52];
ub2 tempub2;
sb2 tempsb2;
OCIRowid *tmprid;
tempsb2 = nctx->cons_ind[i];
nctx->cons_ind[i] = nctx->cons_ind[j];
nctx->cons_ind[j] = tempsb2;
tempub2 = nctx->cons_len[i];
nctx->cons_len[i] = nctx->cons_len[j];
nctx->cons_len[j] = tempub2;
tempub2 = nctx->cons_rcode[i];
nctx->cons_rcode[i] = nctx->cons_rcode[j];
nctx->cons_rcode[j] = tempub2;
tempi = nctx->cons[i];
nctx->cons[i] = nctx->cons[j];
nctx->cons[j] = tempi;
tempsb2 = nctx->s_rowid_ind[i];
nctx->s_rowid_ind[i] = nctx->s_rowid_ind[j];
```

```
nctx->s_rowid_ind[j] = tempsb2;
tempub2 = nctx->s_rowid_len[i];
nctx->s_rowid_len[i] = nctx->s_rowid_len[j];
nctx->s_rowid_len[j] = tempub2;
tempub2 = nctx->s_rowid_rcode[i];
nctx->s_rowid_rcode[i] = nctx->s_rowid_rcode[j];
nctx->s_rowid_rcode[j] = tempub2;
tmprid = nctx->s_rowid_ptr[i];
nctx->s_rowid_ptr[i]= nctx->s_rowid_ptr[j];
nctx->s_rowid_ptr[j]=tmprid;
tempsb2 = nctx->i_price_ind[i];
nctx->i_price_ind[i] = nctx->i_price_ind[j];
nctx->i_price_ind[j] = tempsb2;
tempub2 = nctx->i_price_len[i];
nctx->i_price_len[i] = nctx->i_price_len[j];
nctx->i_price_len[j] = tempub2;
tempub2 = nctx->i_price_rcode[i];
nctx->i_price_rcode[i] = nctx->i_price_rcode[j];
nctx->i_price_rcode[j] = tempub2;
tempf = i_price[i];
i_price[i] = i_price[j];
i_price[j] = tempf;
tempsb2 = nctx->i_name_ind[i];
nctx->i_name_ind[i] = nctx->i_name_ind[j];
nctx->i_name_ind[j] = tempsb2;
tempub2 = nctx->i_name_len[i];
nctx->i_name_len[i] = nctx->i_name_len[j];
nctx->i_name_len[j] = tempub2;
tempub2 = nctx->i_name_rcode[i];
nctx->i_name_rcode[i] = nctx->i_name_rcode[j];
nctx->i_name_rcode[j] = tempub2;
strncpy (tempstr, i_name[i], 25);
strncpy (i_name[i], i_name[j], 25);
strncpy (i_name[j], tempstr, 25);
tempsb2 = nctx->i_data_ind[i];
nctx->i_data_ind[i] = nctx->i_data_ind[j];
nctx->i_data_ind[j] = tempsb2;
tempub2 = nctx->i_data_len[i];
nctx->i_data_len[i] = nctx->i_data_len[j];
nctx->i_data_len[j] = tempub2;
tempub2 = nctx->i_data_rcode[i];
nctx->i_data_rcode[i] = nctx->i_data_rcode[j];
nctx->i_data_rcode[j] = tempub2;
strncpy (tempstr, nctx->i_data[i], 51);
strncpy (nctx->i_data[i], nctx->i_data[j], 51);
strncpy (nctx->i_data[j], tempstr, 51);
tempsb2 = nctx->s_quantity_ind[i];
nctx->s_quantity_ind[i] = nctx->s_quantity_ind[j];
nctx->s_quantity_ind[j] = tempsb2;
tempub2 = nctx->s_quantity_len[i];
nctx->s_quantity_len[i] = nctx->s_quantity_len[j];
nctx->s_quantity_len[j] = tempub2;
tempub2 = nctx->s_quantity_rcode[i];
nctx->s_quantity_rcode[i] = nctx->s_quantity_rcode[j];
nctx->s_quantity_rcode[j] = tempub2;
tempi = s_quantity[i];
s_quantity[i] = s_quantity[j];
s_quantity[j] = tempi;
tempsb2 = nctx->s_dist_info_ind[i];
nctx->s_dist_info_ind[i] = nctx->s_dist_info_ind[j];
nctx->s_dist_info_ind[j] = tempsb2;
tempub2 = nctx->s_dist_info_len[i];
nctx->s_dist_info_len[i] = nctx->s_dist_info_len[j];
nctx->s_dist_info_len[j] = tempub2;
tempub2 = nctx->s_dist_info_rcode[i];
nctx->s_dist_info_rcode[i] = nctx->s_dist_info_rcode[j];
nctx->s_dist_info_rcode[j] = tempub2;
strncpy (tempstr, nctx->s_dist_info[i], 25);
strncpy (nctx->s_dist_info[i], nctx->s_dist_info[j], 25);
strncpy (nctx->s_dist_info[j], tempstr, 25);
tempsb2 = nctx->s_data_ind[i];
nctx->s_data_ind[i] = nctx->s_data_ind[j];
nctx->s_data_ind[j] = tempsb2;
tempub2 = nctx->s_data_len[i];
nctx->s_data_len[i] = nctx->s_data_len[j];
nctx->s_data_len[j] = tempub2;
tempub2 = nctx->s_data_rcode[i];
nctx->s_data_rcode[i] = nctx->s_data_rcode[j];
nctx->s_data_rcode[j] = tempub2;
strncpy (tempstr, nctx->s_data[i], 51);
strncpy (nctx->s_data[i], nctx->s_data[j], 51);
strncpy (nctx->s_data[j], tempstr, 51);
}
#endif /* end of TODO */
static int SelItemStk (nctx, newP, tpcsvc, errhp)
newctx *nctx;
global_newOrder_t *newP;
OCISvcCtx *tpcsvc;
OCIError *errhp;
{
int i, j, rpc3,rcount;
/* array select from item and stock tables */
newP->execstatus=OCIStmtExecute(tpcsvc,(nctx->curn3)[newP->d_id-1],errhp,newP->o_ol
_cnt,
0,0,0,OCI_DEFAULT);
if((newP->execstatus != OCI_SUCCESS) && (newP->execstatus != OCI_NO_DATA)) {
newP->errcode = OCIERROR(errhp,newP->execstatus);
if(newP->errcode == NOT_SERIALIZABLE) {
newP->retries++;
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-2);
} else if (newP->errcode == RECOVERR) {
/* In case of NO_DATA this should NOT return, but simply fall through */
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
newP->retries++;
return (-2);
} else {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-1);
}
}
/* mark invalid items */
OCIAttrGet((nctx->curn3)[newP->d_id-1], OCI_HTYPE_STMT,&rcount,NULL,
OCI_ATTR_ROWCNT, errhp);
rpc3 = rcount;
/* the result is in order, so we have to shift up to fill */
/* the slot for the line with the invalid item. */
/* If more than one item is wrong, this is not an simulated */
/* error and we'll blow off */
if ((newP->status = newP->o_ol_cnt - rcount) >1)
{
fprintf (stderr, "TPC-C server %d: more than 1 invalid item?\n", newP->proc_no);
return (rpc3);
}
if (newP->status == 0) return (rpc3);
/* find the invalid item, transfer the rowid information */
for (i = 0; i < newP->o_ol_cnt; i++) {
if (nctx->cons[i] != i) break; /* this item is invalid */
}
/*
fprintf (stderr, "TPC-C server %d: reordering items and stocks\n",
newP->proc_no); */
/* not the last item - shift up */
for (j = i; j < newP->o_ol_cnt-1; j++)
{
shiftitemstock (j, j+1, nctx, newP);
}
/* zero the last item */
i = newP->o_ol_cnt-1;
nctx->nol_i_id_ind[i] = NA;
nctx->nol_supply_w_id_ind[i] = NA;
nctx->nol_quantity_ind[i] = NA;
nctx->nol_amount_ind[i] = NA;
nctx->ol_w_id_ind[i] = NA;
nctx->ol_d_id_ind[i] = NA;
nctx->ol_o_id_ind[i] = NA;
nctx->null_date_ind[i]= NA;
nctx->ol_number_ind[i] = NA;
nctx->ol_dist_info_ind[i] = NA;
nctx->s_remote_ind[i] = NA;
nctx->s_quant_ind[i] = NA;
nctx->nol_i_id_len[i] = 0;
nctx->nol_supply_w_id_len[i] = 0;
nctx->nol_quantity_len[i] = 0;
nctx->nol_amount_len[i] = 0;
nctx->ol_w_id_len[i] = 0;
nctx->ol_d_id_len[i] = 0;
nctx->ol_o_id_len[i] = 0;
nctx->ol_number_len[i] = 0;
nctx->ol_dist_info_len[i] = 0;
nctx->null_date_ind[i] = 0;
nctx->s_remote_len[i] = 0;
nctx->s_quant_len[i] = 0;
return (rpc3);
}
/* TODO - no changes yet for OPS code */
#ifdef OPS
UpdStk ()
{
int rcount;
/* array update of stock table */
execstatus = OCIStmtExecute(tpcsvc,nctx->curn2,errhp,o_ol_cnt,
0,0,0,OCI_DEFAULT);
if(execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
errcode = OCIERROR(errhp,execstatus);
if(errcode == NOT_SERIALIZABLE) {
retries++;
return (-2);
} else if (errcode == RECOVERR) {
retries++;
return (-2);
} else {
return -1;
}
}
OCIAttrGet(nctx->curn2,OCI_HTYPE_STMT,&rcount,NULL,
OCI_ATTR_ROWCNT, errhp);
if (rcount != (o_ol_cnt) ) {
fprintf (stderr, "Error in TPC-C server %d: array update failed in UpdStk()\n",
proc_no);
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-1);
}
return (rcount);
}
#endif
#ifndef OPS
static int UpdStk2 (nctx, newP, tpcsvc, errhp)
newctx *nctx;
global_newOrder_t *newP;
OCISvcCtx *tpcsvc;
OCIError *errhp;
{
int rpc, rowoff, iters,rcount;
/* array update of stock table */
newP->execstatus =
OCIStmtExecute(tpcsvc,nctx->curn2,errhp,newP->o_ol_cnt-newP->status,0,0,0,
OCI_DEFAULT);
if(newP->execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
newP->errcode = OCIERROR(errhp,newP->execstatus);
if(newP->errcode == NOT_SERIALIZABLE) {
newP->retries++;
return (-2);
} else if (newP->errcode == RECOVERR) {
newP->retries++;
return (-2);
} else {
return -1;
```

```
}
}
OCIAttrGet(nctx->curn2,OCI_HTYPE_STMT,&rcount,NULL, OCI_ATTR_ROWCNT, errhp);
rpc = rcount;
if (rpc != (newP->o_ol_cnt - newP->status)) {
fprintf (stderr, "Error in TPC-C server %d: array update failed\n",
newP->proc_no);
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-1);
}
return (rpc);
}
#endif
```

plnew.h

```
#ifndef TPCC_PLNEW_H
#define TPCC_PLNEW_H
#include <oci.h>
struct newctx {
sb2 nol_i_id_ind[NITEMS];
sb2 nol_supply_w_id_ind[NITEMS];
sb2 nol_quantity_ind[NITEMS];
sb2 nol_amount_ind[NITEMS];
sb2 i_name_ind[NITEMS];
sb2 s_quantity_ind[NITEMS];
sb2 i_price_ind[NITEMS];
sb2 ol_w_id_ind[NITEMS];
sb2 ol_d_id_ind[NITEMS];
sb2 ol_o_id_ind[NITEMS];
sb2 ol_number_ind[NITEMS];
sb2 cons_ind[NITEMS];
sb2 s_rowid_ind[NITEMS];
sb2 s_remote_ind[NITEMS];
sb2 s_quant_ind[NITEMS];
sb2 i_data_ind[NITEMS];
sb2 s_data_ind[NITEMS];
sb2 s_dist_info_ind[NITEMS];
sb2 ol_dist_info_ind[NITEMS];
sb2 null_date_ind[NITEMS];
ub2 nol_i_id_len[NITEMS];
ub2 nol_supply_w_id_len[NITEMS];
ub2 nol_quantity_len[NITEMS];
ub2 nol_amount_len[NITEMS];
ub2 i_name_len[NITEMS];
ub2 s_quantity_len[NITEMS];
ub2 i_price_len[NITEMS];
ub2 ol_w_id_len[NITEMS];
ub2 ol_d_id_len[NITEMS];
ub2 ol_o_id_len[NITEMS];
ub2 ol_number_len[NITEMS];
ub2 cons_len[NITEMS];
ub2 s_rowid_len[NITEMS];
ub2 s_remote_len[NITEMS];
ub2 s_quant_len[NITEMS];
ub2 i_data_len[NITEMS];
ub2 s_data_len[NITEMS];
ub2 s_dist_info_len[NITEMS];
ub2 ol_dist_info_len[NITEMS];
ub2 null_date_len[NITEMS];
ub2 nol_i_id_rcode[NITEMS];
ub2 nol_supply_w_id_rcode[NITEMS];
ub2 nol_quantity_rcode[NITEMS];
ub2 nol_amount_rcode[NITEMS];
ub2 i_name_rcode[NITEMS];
ub2 s_quantity_rcode[NITEMS];
ub2 i_price_rcode[NITEMS];
ub2 ol_w_id_rcode[NITEMS];
ub2 ol_d_id_rcode[NITEMS];
ub2 ol_o_id_rcode[NITEMS];
ub2 ol_number_rcode[NITEMS];
ub2 cons_rcode[NITEMS];
ub2 s_rowid_rcode[NITEMS];
ub2 s_remote_rcode[NITEMS];
ub2 s_quant_rcode[NITEMS];
ub2 i_data_rcode[NITEMS];
ub2 s_data_rcode[NITEMS];
ub2 s_dist_info_rcode[NITEMS];
ub2 ol_dist_info_rcode[NITEMS];
ub2 null_date_rc[NITEMS];
int ol_w_id[NITEMS];
int ol_d_id[NITEMS];
int ol_o_id[NITEMS];
int ol_number[NITEMS];
int cons[NITEMS];
OCIRowid *s_rowid_ptr[NITEMS];
int s_remote[NITEMS];
char i_data[NITEMS][51];
char s_data[NITEMS][51];
char s_dist_info[NITEMS][25];
unsigned char null_date[NITEMS][7]; /* base date for null date entry */
OCIStmt *curn;
OCIStmt *curn1;
OCIStmt *curn2;
OCIStmt *curn3[10];
OCIStmt *curn4;
OCIBind *w_id_bp;
OCIBind *d_id_bp;
OCIBind *c_id_bp;
OCIBind *o_all_local_bp;
OCIBind *o_all_cnt_bp;
OCIBind *w_tax_bp;
OCIBind *d_tax_bp;
OCIBind *o_id_bp;
OCIBind *c_discount_bp;
OCIBind *c_credit_bp;
OCIBind *c_last_bp;
OCIBind *retries_bp;
OCIBind *cr_date_bp;
OCIBind *ol_i_id_bp;
OCIBind *ol_supply_w_id_bp;
OCIBind *s_quantity_bp;
OCIBind *s_rowid_bp;
OCIBind *ol_quantity_bp;
OCIBind *s_remote_bp;
OCIBind *id_bp[10][15];
OCIBind *sd_bp[10][15];
OCIDefine *Dcons[10];
OCIDefine *Ds_rowid[10];
OCIDefine *Di_price[10];
OCIDefine *Di_data[10];
OCIDefine *Ds_dist_info[10];
OCIDefine *Ds_data[10];
OCIDefine *Ds_quantity[10];
OCIDefine *Di_name[10];
OCIBind *ol_o_id_bp;
OCIBind *ol_d_id_bp;
OCIBind *ol_w_id_bp;
OCIBind *ol_number_bp;
OCIBind *ol_amount_bp;
OCIBind *ol_dist_info_bp;
OCIBind *null_date_bp;
sb2 w_id_ind;
ub2 w_id_len;
ub2 w_id_rc;
sb2 d_id_ind;
ub2 d_id_len;
ub2 d_id_rc;
sb2 c_id_ind;
ub2 c_id_len;
ub2 c_id_rc;
sb2 o_all_local_ind;
ub2 o_all_local_len;
ub2 o_all_local_rc;
sb2 o_ol_cnt_ind;
ub2 o_ol_cnt_len;
ub2 o_ol_cnt_rc;
sb2 w_tax_ind;
ub2 w_tax_len;
ub2 w_tax_rc;
sb2 d_tax_ind;
ub2 d_tax_len;
ub2 d_tax_rc;
sb2 o_id_ind;
ub2 o_id_len;
ub2 o_id_rc;
sb2 c_discount_ind;
ub2 c_discount_len;
ub2 c_discount_rc;
sb2 c_credit_ind;
ub2 c_credit_len;
ub2 c_credit_rc;
sb2 c_last_ind;
ub2 c_last_len;
ub2 c_last_rc;
sb2 retries_ind;
ub2 retries_len;
ub2 retries_rc;
sb2 cr_date_ind;
ub2 cr_date_len;
ub2 cr_date_rc;
int cs;
int norow;
};
typedef struct newctx newctx;
/* struct to copy-in/out neworder vars */
struct global_newOrder_t {
int w_id;
int d_id;
int c_id;
int nol_i_id[15];
int nol_supply_w_id[15];
int nol_quantity[15];
int retries;
char o_entry_d[20];
int o_id;
int o_ol_cnt;
char c_last[17];
char c_credit[3];
int c_discount;
int w_tax;
int d_tax;
float total_amount;
char i_name[15][25];
int s_quantity[15];
char brand_gen[15];
int i_price[15];
int nol_amount[15];
int status;
int o_all_local;
int errcode;
int execstatus;
int proc_no;
unsigned char cr_date[7];
};
typedef struct global_newOrder_t global_newOrder_t;
#endif /* TPCC_PLNEW_H */
```

plora.h

```
#ifndef TPCC_PLORA_H
#define TPCC_PLORA_H
#include <dce/pthread.h>
#include "tpcc.h"
#include "tpccpl.h"
#include "plnew.h"
```

```
#include "plpay.h"
#include "plord.h"
#include "plsto.h"
#include "pldel.h"
#include "tpcc_info.h"
#define MAX_ORA_CONNECTIONS 100
#define NEWO_TRANS (1)
#define PAYMENT_TRANS (2)
#define ORDER_STAT_TRANS (3)
#define DELIVERY_TRANS (4)
#define STOCK_TRANS (5)
#define DIST_NEWO_TRANS (6)
#define DIST_PAY_TRANS (7)
#define MAX_TRAN_TYPE (7)
#define NUM_STATES 4
#define SVR_STATE_NONE 0
#define SVR_STATE_SENT 1
#define SVR_STATE_REPLIED 2
#define SVR_STATE_ERR 3
typedef struct {
int num;
int errs;
double RT;
} tran_info_t;
/*
* total_tran_count_t
*
* structure that holds the total count of transaction of each type
* as well as the reposne times.
*
*/
typedef struct {
tran_info_t tran[MAX_TRAN_TYPE + 1];
double dvry_queue_time; /* The average time a
* delivery request was in the queue
*/
int errors;
} total_tran_count_t;
typedef struct {
int tran_type;
union {
struct newstruct no;
struct paystruct pa;
struct ordstruct os;
struct delstruct dl;
struct stostruct sl;
} tran_info;
} tran_spec_info_t;
/* Oracle handles and rest of thread specific vars(thread slot data ) */
struct ora_cn_data_t {
OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
OCIStmt *curi;
dvoid *xmem;
global_newOrder_t *globals;
global_payment_t *payP;
global_order_t *ordP;
global_delivery_t *delP;
global_stock_t *stoP;
newctx *nctx;
payctx *pctx;
ordctx *octx;
stoctx *sctx;
delctx *dctx;
int calls; /* Number of times it was used */
int errors; /* Total number of errors on this connection */
int calls_last_err; /* Number of calls when the last error occured */
int consecutive_errs; /* Number of consecutive errs */
int connect_time; /* Time (seconds) connections was created */
/* For debug */
int state; /* State of the connection */
struct timeval tran_time; /* Time this tran started */
int cur_tran_type;
void *cur_tran_dataP;
total_tran_count_t stat;
int printed;
};
typedef struct ora_cn_data_t ora_cn_data_t;
extern int plnewinit (ora_cn_data_t *);
extern int plpayinit (ora_cn_data_t *);
extern int plordinit (ora_cn_data_t *);
extern int pldelinit (ora_cn_data_t *);
extern int plstoinit (ora_cn_data_t *);
extern int plnew (ora_cn_data_t *);
extern int plpay (ora_cn_data_t *);
extern int plord (ora_cn_data_t *);
extern int pldel (ora_cn_data_t *);
extern int plsto (ora_cn_data_t *);
extern void plnewdone (ora_cn_data_t *);
extern void plpaydone (ora_cn_data_t *);
extern void plorddone (ora_cn_data_t *);
extern void pldeldone (ora_cn_data_t *);
extern void plstodone (ora_cn_data_t *);
#endif /* TPCC_PLORA_H */
```

plord.c

```
#ifdef RCSID
static char *RCSid =
"$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/plord.c,v 1.2 1998/01/23 15:08:13 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */
/*+=================================================================+
| Copyright (c) 1995 Oracle Corp, Redwood Shores, CA |
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=================================================================+
| FILENAME
| plord.c
| DESCRIPTION
| OCI version (using PL/SQL anynomous block) of
| ORDER STATUS transaction in TPC-C benchmark.
+=================================================================*/
#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"
#ifdef ISO8
#define SQLTXT "BEGIN aorderstatus.agetstatus (:w_id, :d_id, :c_id, :byln, \
:c_last, :c_first, :c_middle, :c_balance, :o_id, :o_entry_d, :o_cr_id, \
:o_ol_cnt, :ol_s_w_id, :ol_i_id, :ol_quantity, :ol_amount, :ol_d_d); END;"
#endif
#define SQLCUR0 "SELECT rowid FROM customer \
WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last = :c_last \
ORDER BY c_w_id, c_d_id, c_last, c_first"
#define SQLCUR1 "SELECT c_id, c_balance, c_first, c_middle, \
o_id, o_entry_d, o_carrier_id, o_ol_cnt, c_last \
FROM customer, orders \
WHERE customer.rowid = :cust_rowid \
AND o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id = c_id \
ORDER BY o_w_id, o_d_id, o_c_id, o_id DESC"
#define SQLCUR2 "SELECT c_balance, c_first, c_middle, c_last, \
o_id, o_entry_d, o_carrier_id, o_ol_cnt \
FROM customer, orders \
WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id \
AND o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id = c_id \
ORDER BY o_w_id, o_d_id, o_c_id, o_id DESC"
#define SQLCUR3 "SELECT ol_i_id,ol_supply_w_id,ol_quantity,ol_amount, \
ol_delivery_d\
FROM order_line \
WHERE ol_d_id = :d_id AND ol_w_id = :w_id AND ol_o_id = :o_id"
plordinit (ora_cn_data_t *ora_SlotDataP)
{
int i;
text stmbuf[SQL_BUF_SIZE];
ordctx *octx;
global_order_t *ordP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
octx = (ordctx *) malloc (sizeof(ordctx));
memset(octx,(char)0,sizeof(ordctx));
ora_SlotDataP->octx = octx;
ora_SlotDataP->ordP = (global_order_t *)malloc(sizeof(global_order_t));
memset(ora_SlotDataP->ordP,(char)0,sizeof(global_order_t));
ordP = ora_SlotDataP->ordP;
octx->cs = 1;
octx->norow = 0;
/* get the rowid handles */
for(i=0;i<3000;i++) {
OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&octx->c_rowid_ptr[i],
OCI_DTYPE_ROWID,0,(dvoid**)0));
}
#ifdef ISO8
OCIERROR(errhp,
OCIHandleAlloc(tpcenv,(dvoid **)&octx->curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
#else
OCIERROR(errhp,
OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
OCIERROR(errhp,
OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo1,OCI_HTYPE_STMT,0,(dvoid**)0));
OCIERROR(errhp,
OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo2,OCI_HTYPE_STMT,0,(dvoid**)0));
OCIERROR(errhp,
OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo3,OCI_HTYPE_STMT,0,(dvoid**)0));
#endif
#ifdef ISO8
sprintf((char *) stmbuf, SQLTXT);
OCIERROR(errhp,
OCIStmtPrepare(octx->curo0,errhp,stmbuf,strlen((char *)stmbuf),
OCI_NTV_SYNTAX,OCI_DEFAULT));
#else
/* c_id = 0, use find customer by lastname. Get an array or rowid's back*/
sprintf((char *) stmbuf, SQLCUR0);
OCIERROR(errhp,
OCIStmtPrepare(octx->curo0,errhp,stmbuf,strlen((char *)stmbuf),
OCI_NTV_SYNTAX,OCI_DEFAULT));
OCIERROR(errhp,
OCIAttrSet(octx->curo0,OCI_HTYPE_STM,(dvoid*)&octx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
/* get order/customer info back based on rowid */
sprintf((char *) stmbuf, SQLCUR1);
OCIERROR(errhp,
OCIStmtPrepare(octx->curo1,errhp,stmbuf,strlen((char *)stmbuf),
OCI_NTV_SYNTAX,OCI_DEFAULT));
OCIERROR(errhp,
OCIAttrSet(octx->curo1,OCI_HTYPE_STM,(dvoid*)&octx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
/* c_id == 0, use lastname to find customer */
sprintf((char *) stmbuf, SQLCUR2);
OCIERROR(errhp,
OCIStmtPrepare(octx->curo2,errhp,stmbuf,strlen((char *)stmbuf),
OCI_NTV_SYNTAX,OCI_DEFAULT));
OCIERROR(errhp,
OCIAttrSet(octx->curo2,OCI_HTYPE_STM,(dvoid*)&octx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
sprintf((char *) stmbuf, SQLCUR3);
OCIERROR(errhp,
OCIStmtPrepare(octx->curo3,errhp,stmbuf,strlen((char *)stmbuf),
OCI_NTV_SYNTAX,OCI_DEFAULT));
OCIERROR(errhp,
OCIAttrSet(octx->curo3,OCI_HTYPE_STM,(dvoid*)&octx->norow,0,
```

```
OCI_ATTR_PREFETCH_ROWS,errhp));
#endif
for (i = 0; i < NITEMS; i++) {
octx->ol_supply_w_id_ind[i] = TRUE;
octx->ol_i_id_ind[i] = TRUE;
octx->ol_quantity_ind[i] = TRUE;
octx->ol_amount_ind[i] = TRUE;
octx->ol_delivery_d_ind[i] = TRUE;
octx->ol_supply_w_id_len[i] = sizeof(int);
octx->ol_i_id_len[i] = sizeof(int);
octx->ol_quantity_len[i] = sizeof(int);
octx->ol_amount_len[i] = sizeof(int);
octx->ol_delivery_d_len[i] = sizeof(ordP->ol_d_base[0]);
}
octx->ol_supply_w_id_csize = NITEMS;
octx->ol_i_id_csize = NITEMS;
octx->ol_quantity_csize = NITEMS;
octx->ol_amount_csize = NITEMS;
octx->ol_delivery_d_csize = NITEMS;
octx->ol_w_id_csize = NITEMS;
octx->ol_o_id_csize = NITEMS;
octx->ol_d_id_csize = NITEMS;
octx->ol_w_id_ind = TRUE;
octx->ol_d_id_ind = TRUE;
octx->ol_o_id_ind = TRUE;
octx->ol_w_id_len = sizeof(int);
octx->ol_d_id_len = sizeof(int);
octx->ol_o_id_len = sizeof(int);
/* bind variables */
#ifdef ISO8
OCIBND(octx->curo0,octx->w_id_bp0,errhp,":w_id",ADR(ordP->w_id),SIZ(ordP->w_id),
SQLT_INT);
OCIBND(octx->curo0, octx->d_id_bp0, errhp,":d_id",ADR(ordP->d_id),SIZ(ordP->d_id),
SQLT_INT);
OCIBND(octx->curo0, octx->c_id_bp, errhp,":c_id",ADR(ordP->c_id),SIZ(ordP->c_id),
SQLT_INT);
OCIBND(octx->curo0, octx->byln_bp, errhp,":byln",ADR(ordP->bylastname),
SIZ(ordP->bylastname),SQLT_INT);
OCIBND(octx->curo0, octx->c_last_bp, errhp,":c_last",ordP->c_last,
SIZ(ordP->c_last),SQLT_STR);
OCIBND(octx->curo0, octx->c_first_bp, errhp,":c_first",ordP->c_first,
SIZ(ordP->c_first),SQLT_STR);
OCIBND(octx->curo0, octx->c_middle_bp, errhp,":c_middle",ordP->c_middle,
SIZ(ordP->c_middle),SQLT_STR);
OCIBND(octx->curo0, octx->c_balance_bp, errhp,":c_balance",ADR(ordP->c_balance),
SIZ(ordP->c_balance),SQLT_FLT);
OCIBND(octx->curo0, octx->o_id_bp, errhp,":o_id",ADR(ordP->o_id),
SIZ(ordP->o_id),SQLT_INT);
OCIBND(octx->curo0, octx->o_entry_d_bp, errhp,":o_entry_d",
ordP->o_entry_d_base,SIZ(ordP->o_entry_d_base),SQLT_DAT);
OCIBND(octx->curo0, octx->o_cr_id_bp, errhp,":o_cr_id",ADR(ordP->o_carrier_id),
SIZ(ordP->o_carrier_id), SQLT_INT);
OCIBND(octx->curo0, octx->o_ol_cnt_bp, errhp,":o_ol_cnt",ADR(ordP->o_ol_cnt),
SIZ(ordP->o_ol_cnt),SQLT_INT);
OCIBNDRAA(octx->curo0, octx->ol_s_w_id_bp, errhp,":ol_s_w_id",
ordP->ol_supply_w_id,SIZ(int),SQLT_INT,
octx->ol_supply_w_id_ind,octx->ol_supply_w_id_len,
octx->ol_supply_w_id_rcode,NITEMS,
ADR(octx->ol_supply_w_id_csize));
OCIBNDRAA(octx->curo0, octx->ol_i_id_bp, errhp,":ol_i_id",ordP->ol_i_id,
SIZ(int),SQLT_INT,
octx->ol_i_id_ind,octx->ol_i_id_len,octx->ol_i_id_rcode,NITEMS,
ADR(octx->ol_i_id_csize));
OCIBNDRAA(octx->curo0, octx->ol_quantity_bp, errhp,":ol_quantity",
ordP->ol_quantity,SIZ(int),SQLT_INT,
octx->ol_quantity_ind,octx->ol_quantity_len,
octx->ol_quantity_rcode, NITEMS,ADR(octx->ol_quantity_csize));
OCIBNDRAA(octx->curo0, octx->ol_amount_bp, errhp,":ol_amount",ordP->ol_amount,
SIZ(float),SQLT_FLT,
octx->ol_amount_ind,octx->ol_amount_len,octx->ol_amount_rcode,
NITEMS,ADR(octx->ol_amount_csize));
OCIBNDRAA(octx->curo0, octx->ol_d_d_bp, errhp,":ol_d_d",ordP->ol_delivery_d,
SIZ(ordP->ol_delivery_d[0]),SQLT_STR,
octx->ol_delivery_d_ind,octx->ol_delivery_d_len,
octx->ol_delivery_d_rcode,NITEMS,
ADR(octx->ol_delivery_d_csize));
#else
/* c_id (customer id) is not known */
OCIBND(octx->curo0,octx->w_id_bp0,errhp,":w_id",ADR(ordP->w_id),SIZ(int),SQLT_INT);
OCIBND(octx->curo0,octx->d_id_bp0,errhp,":d_id",ADR(ordP->d_id),SIZ(int),SQLT_INT);
OCIBND(octx->curo0,octx->c_last_bp,errhp,":c_last",ordP->c_last,SIZ(ordP->c_last),
SQLT_STR);
OCIDFNRA(octx->curo0,octx->c_rowid_dp,errhp,1,octx->c_rowid_ptr,
sizeof(octx->c_rowid_ptr[0]),SQLT_RDD,octx->c_rowid_ind,
octx->c_rowid_len,octx->c_rowid_rcode);
OCIBND(octx->curo1,octx->c_rowid_bp,errhp,":cust_rowid",
&octx->c_rowid_ptr[octx->cust_idx],
sizeof( octx->c_rowid_ptr[0]),SQLT_RDD);
OCIDEF(octx->curo1,octx->c_id_dp,errhp,1,ADR(ordP->c_id),SIZ(int),SQLT_INT);
OCIDEF(octx->curo1,octx->c_balance_dp1,errhp,2,ADR(ordP->c_balance),
SIZ(double),SQLT_FLT);
OCIDEF(octx->curo1,octx->c_first_dp1,errhp,3,ordP->c_first,SIZ(ordP->c_first),
SQLT_STR);
OCIDEF(octx->curo1,octx->c_middle_dp1,errhp,4,ordP->c_middle,
SIZ(ordP->c_middle),SQLT_STR);
OCIDEF(octx->curo1,octx->o_id_dp1,errhp,5,ADR(ordP->o_id),SIZ(int),SQLT_INT);
OCIDEF(octx->curo1,octx->o_entry_d_dp1,errhp,6,
ordP-> o_entry_d_base,SIZ(ordP->o_entry_d_base),SQLT_DAT);
OCIDEF(octx->curo1,octx->o_cr_id_dp1,errhp,7,ADR(ordP->o_carrier_id),
SIZ(int),SQLT_INT);
OCIDEF(octx->curo1,octx->o_ol_cnt_dp1,errhp,8,ADR(ordP->o_ol_cnt),
SIZ(int),SQLT_INT);
OCIDEF(octx->curo1,octx->c_last_dp1,errhp,9,ordP->c_last,SIZ(ordP->c_last),
SQLT_STR);
/* Bind for third cursor , no-zero customer id */
OCIBND(octx->curo2,octx->w_id_bp2,errhp,":w_id",ADR(ordP->w_id),SIZ(int),SQLT_INT);
OCIBND(octx->curo2,octx->d_id_bp2,errhp,":d_id",ADR(ordP->d_id),SIZ(int),SQLT_INT);
OCIBND(octx->curo2,octx->c_id_bp,errhp,":c_id",ADR(ordP->c_id),SIZ(int),SQLT_INT);
OCIDEF(octx->curo2,octx->c_balance_dp2,errhp,1,ADR(ordP->c_balance),
SIZ(double),SQLT_FLT);
OCIDEF(octx->curo2,octx->c_first_dp2,errhp,2,ordP->c_first,SIZ(ordP->c_first),
SQLT_STR);
OCIDEF(octx->curo2,octx->c_middle_dp2,errhp,3,ordP->c_middle,
SIZ(ordP->c_middle),SQLT_STR);
OCIDEF(octx->curo2,octx->c_last_dp,errhp,4,ordP->c_last,SIZ(ordP->c_last),
SQLT_STR);
OCIDEF(octx->curo2,octx->o_id_dp2,errhp,5,ADR(ordP->o_id),SIZ(int),SQLT_INT);
OCIDEF(octx->curo2,octx->o_entry_d_dp2,errhp,6,
ordP->o_entry_d_base,SIZ(ordP->o_entry_d_base),SQLT_DAT);
OCIDEF(octx->curo2, octx->o_cr_id_dp2,errhp,7,ADR(ordP->o_carrier_id),
SIZ(int), SQLT_INT);
OCIDEF(octx->curo2,octx->o_ol_cnt_dp2,errhp,8,ADR(ordP->o_ol_cnt),
SIZ(int),SQLT_INT);
/* Bind for last cursor */
OCIBND(octx->curo3,octx->w_id_bp3,errhp,":w_id",ADR(ordP->w_id),SIZ(int),SQLT_INT);
OCIBND(octx->curo3,octx->d_id_bp3,errhp,":d_id",ADR(ordP->d_id),SIZ(int),SQLT_INT);
OCIBND(octx->curo3,octx->o_id_bp,errhp,":o_id",ADR(ordP->o_id),SIZ(int),SQLT_INT);
OCIDFNRA(octx->curo3, octx->ol_i_id_dp, errhp, 1, ordP->ol_i_id,SIZ(int),SQLT_INT,
octx->ol_i_id_ind,octx->ol_i_id_len, octx->ol_i_id_rcode);
OCIDFNRA(octx->curo3,octx->ol_supply_w_id_dp,errhp,2, ordP->ol_supply_w_id,
SIZ(int),SQLT_INT, octx->ol_supply_w_id_ind,
octx->ol_supply_w_id_len, octx->ol_supply_w_id_rcode);
OCIDFNRA(octx->curo3, octx->ol_quantity_dp,errhp,3, ordP->ol_quantity,SIZ(int),
SQLT_INT, octx->ol_quantity_ind,octx->ol_quantity_len,
octx->ol_quantity_rcode);
OCIDFNRA(octx->curo3,octx->ol_amount_dp,errhp,4,ordP->ol_amount, SIZ(int),
SQLT_INT,octx->ol_amount_ind, octx->ol_amount_len,
octx->ol_amount_rcode);
OCIDFNRA(octx->curo3,octx->ol_d_base_dp,errhp,5,ordP->ol_d_base,7, SQLT_DAT,
octx->ol_delivery_d_ind,octx->ol_delivery_d_len,
octx->ol_delivery_d_rcode);
#endif /* ISO8 */
return (0);
}
plord (ora_cn_data_t *ora_SlotDataP)
{
int i;
int rcount;
ordctx *octx = ora_SlotDataP->octx;
global_order_t *ordP = ora_SlotDataP->ordP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
retry:
for (i = 0; i < NITEMS; i++) {
octx->ol_supply_w_id_ind[i] = TRUE;
octx->ol_i_id_ind[i] = TRUE;
octx->ol_quantity_ind[i] = TRUE;
octx->ol_amount_ind[i] = TRUE;
octx->ol_delivery_d_ind[i] = TRUE;
octx->ol_supply_w_id_len[i] = sizeof(int);
octx->ol_i_id_len[i] = sizeof(int);
octx->ol_quantity_len[i] = sizeof(int);
octx->ol_amount_len[i] = sizeof(int);
octx->ol_delivery_d_len[i] = sizeof(ordP->ol_d_base[0]);
}
octx->ol_supply_w_id_csize = NITEMS;
octx->ol_i_id_csize = NITEMS;
octx->ol_quantity_csize = NITEMS;
octx->ol_amount_csize = NITEMS;
octx->ol_delivery_d_csize = NITEMS;
#ifdef ISO8
OCIERROR(errhp,
OCIStmtExecute(tpcsvc,octx->curo0,errhp,1,0,0,0,OCI_DEFAULT));
#else
if(ordP->bylastname) {
ordP->execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,3000,0,0,0,OCI_DEFAULT);
if (ordP->execstatus != OCI_NO_DATA) /* will get OCI_NO_DATA if <3000 found */
{
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
ordP->errcode = OCIERROR(errhp,ordP->execstatus);
if(ordP->errcode == NOT_SERIALIZABLE) {
ordP->retries++;
goto retry;
} else if (ordP->errcode == RECOVERR) {
ordP->retries++;
goto retry;
} else {
return -1;
}
}
/* get rowcount, find middle one */
OCIAttrGet(octx->curo0,OCI_HTYPE_STM,&rcount,NULL,OCI_ATTR_ROWCNT,errhp);
octx->cust_idx=(rcount+1)/2 ;
ordP->execstatus = OCIStmtExecute(tpcsvc,octx->curo1,errhp,1,0,0,0,OCI_DEFAULT);
if (ordP->execstatus != OCI_SUCCESS)
{
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
ordP->errcode = OCIERROR(errhp,ordP->execstatus);
if(ordP->errcode == NOT_SERIALIZABLE) {
ordP->retries++;
goto retry;
} else if (ordP->errcode == RECOVERR) {
ordP->retries++;
goto retry;
} else {
return -1;
}
}
} else {
ordP->execstatus = OCIStmtExecute(tpcsvc,octx->curo2,errhp,1,0,0,0,OCI_DEFAULT);
if (ordP->execstatus != OCI_SUCCESS)
{
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
ordP->errcode = OCIERROR(errhp,ordP->execstatus);
if(ordP->errcode == NOT_SERIALIZABLE) {
ordP->retries++;
```

```
goto retry;
} else if (ordP->errcode == RECOVERR) {
ordP->retries++;
goto retry;
} else {
return -1;
}
}
}
octx->ol_w_id_ind = TRUE;
octx->ol_d_id_ind = TRUE;
octx->ol_o_id_ind = TRUE;
octx->ol_w_id_len = sizeof(int);
octx->ol_d_id_len = sizeof(int);
octx->ol_o_id_len = sizeof(int);
ordP->execstatus = OCIStmtExecute(tpcsvc,octx->curo3,errhp,ordP->o_ol_cnt,0,0,0,
OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
if (ordP->execstatus != OCI_SUCCESS)
{
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
ordP->errcode = OCIERROR(errhp,ordP->execstatus);
if(ordP->errcode == NOT_SERIALIZABLE) {
ordP->retries++;
goto retry;
} else if (ordP->errcode == RECOVERR) {
ordP->retries++;
goto retry;
} else {
return -1;
}
}
#ifdef NOTMORE
OCIERROR(errhp,
OCITransCommit(tpcsvc,errhp,OCI_DEFAULT));
#endif
/* clean up and convert the delivery dates */
for (i = 0; i < ordP->o_ol_cnt; i++) {
if (octx->ol_delivery_d_ind[i] == -1) /* null date in field */
strncpy(ordP->ol_delivery_d[i],"1-1-1811",10);
else
cvtdmy(ordP->ol_d_base[i],ordP->ol_delivery_d[i]);
}
#endif
return (0);
}
void plorddone (ora_cn_data_t *ora_SlotDataP)
{
/* TODO: Should we free the cursor handles?? */
if (ora_SlotDataP->octx) {
free (ora_SlotDataP->octx);
ora_SlotDataP->octx = NULL;
}
if (ora_SlotDataP->ordP) {
free(ora_SlotDataP->ordP);
ora_SlotDataP->ordP = NULL;
}
}
```

plord.h

```
#ifndef TPCC_PLORD_H
#define TPCC_PLORD_H
#include <oci.h>
struct ordctx {
sb2 c_rowid_ind[3000];
sb2 ol_supply_w_id_ind[NITEMS];
sb2 ol_i_id_ind[NITEMS];
sb2 ol_quantity_ind[NITEMS];
sb2 ol_amount_ind[NITEMS];
sb2 ol_delivery_d_ind[NITEMS];
sb2 ol_w_id_ind;
sb2 ol_d_id_ind;
sb2 ol_o_id_ind;
sb2 c_id_ind;
sb2 c_first_ind;
sb2 c_middle_ind;
sb2 c_balance_ind;
sb2 c_last_ind;
sb2 o_id_ind;
sb2 o_entry_d_ind;
sb2 o_carrier_id_ind;
sb2 o_ol_cnt_ind;
ub2 c_rowid_len[3000];
ub2 ol_supply_w_id_len[NITEMS];
ub2 ol_i_id_len[NITEMS];
ub2 ol_quantity_len[NITEMS];
ub2 ol_amount_len[NITEMS];
ub2 ol_delivery_d_len[NITEMS];
ub2 ol_w_id_len;
ub2 ol_d_id_len;
ub2 ol_o_id_len;
ub2 c_rowid_rcode[3000];
ub2 ol_supply_w_id_rcode[NITEMS];
ub2 ol_i_id_rcode[NITEMS];
ub2 ol_quantity_rcode[NITEMS];
ub2 ol_amount_rcode[NITEMS];
ub2 ol_delivery_d_rcode[NITEMS];
ub2 ol_w_id_rcode;
ub2 ol_d_id_rcode;
ub2 ol_o_id_rcode;
ub4 ol_supply_w_id_csize;
ub4 ol_i_id_csize;
ub4 ol_quantity_csize;
ub4 ol_amount_csize;
ub4 ol_delivery_d_csize;
ub4 ol_w_id_csize;
ub4 ol_d_id_csize;
ub4 ol_o_id_csize;
OCIStmt *curo0;
OCIStmt *curo1;
OCIStmt *curo2;
OCIStmt *curo3;
OCIBind *w_id_bp0;
OCIBind *w_id_bp2;
OCIBind *w_id_bp3;
OCIBind *d_id_bp0;
OCIBind *d_id_bp2;
OCIBind *d_id_bp3;
OCIBind *c_id_bp;
OCIBind *byln_bp;
OCIBind *c_last_bp;
OCIBind *c_first_bp;
OCIBind *c_middle_bp;
OCIBind *c_balance_bp;
OCIBind *o_id_bp;
OCIBind *o_entry_d_bp;
OCIBind *o_cr_id_bp;
OCIBind *o_ol_cnt_bp;
OCIBind *ol_s_w_id_bp;
OCIBind *ol_i_id_bp;
OCIBind *ol_quantity_bp;
OCIBind *ol_amount_bp;
OCIBind *ol_d_d_bp;
OCIBind *c_rowid_bp;
OCIDefine *c_rowid_dp;
OCIDefine *c_last_dp;
OCIDefine *c_last_dp1;
OCIDefine *c_id_dp;
OCIDefine *c_first_dp1;
OCIDefine *c_first_dp2;
OCIDefine *c_middle_dp1;
OCIDefine *c_middle_dp2;
OCIDefine *c_balance_dp1;
OCIDefine *c_balance_dp2;
OCIDefine *o_id_dp1;
OCIDefine *o_id_dp2;
OCIDefine *o_entry_d_dp1;
OCIDefine *o_entry_d_dp2;
OCIDefine *o_cr_id_dp1;
OCIDefine *o_cr_id_dp2;
OCIDefine *o_ol_cnt_dp1;
OCIDefine *o_ol_cnt_dp2;
OCIDefine *ol_d_d_dp;
OCIDefine *ol_i_id_dp;
OCIDefine *ol_supply_w_id_dp;
OCIDefine *ol_quantity_dp;
OCIDefine *ol_amount_dp;
OCIDefine *ol_d_base_dp;
OCIRowid *c_rowid_ptr[3000];
int cs;
int cust_idx;
int norow;
};
typedef struct ordctx ordctx;
struct global_order_t {
unsigned char ol_d_base[15][7];
int w_id;
int d_id;
int c_id;
char c_last[17];
char c_first[17];
char c_middle[3];
double c_balance;
int o_id;
int o_carrier_id;
int o_ol_cnt;
int ol_supply_w_id[15];
int ol_i_id[15];
unsigned char o_entry_d_base[7];
int ol_quantity[15];
char ol_delivery_d[15][11];
int ol_amount[15];
int errcode;
int execstatus;
int retries;
int bylastname;
char o_entry_d[20];
};
typedef struct global_order_t global_order_t;
#endif /* TPCC_PLORD_H */
```

plpay.c

```
#ifdef RCSID
static char *RCSid =
"$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/plpay.c,v 1.2 1998/01/23 15:08:14 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */
/*+=================================================================+
| Copyright (c) 1995 Oracle Corp, Redwood Shores, CA |
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=================================================================+
| FILENAME
| plpay.c
| DESCRIPTION
| OCI version (using PL/SQL stored procedure) of
| PAYMENT transaction in TPC-C benchmark.
+=================================================================*/
#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"
#define SQLTXT_ZERO "BEGIN apayment.adopayment(:w_id, :d_id, :c_w_id, :c_d_id, \
:c_id, 0, \
:h_amount, :c_last, :w_street_1, :w_street_2, :w_city, :w_state, \
```

```
:w_zip, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip, :c_first, \
:c_middle, :c_street_1, :c_street_2, :c_city, :c_state, :c_zip, :c_phone, \
:c_since, :c_credit, :c_credit_lim, :c_discount, :c_balance, :c_data, \
:h_date, :retry, :cr_date); END;"
#define SQLTXT_NONZERO "BEGIN apayment.adopayment(:w_id, :d_id, :c_w_id, :c_d_id,
\
:c_id, 1, \
:h_amount, :c_last, :w_street_1, :w_street_2, :w_city, :w_state, \
:w_zip, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip, :c_first, \
:c_middle, :c_street_1, :c_street_2, :c_city, :c_state, :c_zip, :c_phone, \
:c_since, :c_credit, :c_credit_lim, :c_discount, :c_balance, :c_data, \
:h_date, :retry, :cr_date); END;"
#define SQLTXT_INIT "BEGIN pay.pay_init; END;"
plpayinit (ora_cn_data_t *ora_SlotDataP)
{
char *ora_home = getenv("ORACLE_HOME");
char sql_file_name[256];
payctx *pctx;
global_payment_t *payP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
text stmbuf[SQL_BUF_SIZE];
if (!ora_home) {
err_printf("Cannot find env variable ORACLE_HOME\n");
exit(13);
}
pctx = (payctx *)malloc(sizeof(payctx));
memset(pctx,(char)0,sizeof(payctx));
ora_SlotDataP->pctx = pctx;
ora_SlotDataP->payP = (global_payment_t *)malloc(sizeof(global_payment_t));
memset(ora_SlotDataP->payP,(char)0,sizeof(global_payment_t));
payP = ora_SlotDataP->payP;
/* cursor for init */
OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curpi)),
OCI_HTYPE_STMT,0,(dvoid**)0));
OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curp0)),
OCI_HTYPE_STMT,0,(dvoid**)0));
OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curp1)),
OCI_HTYPE_STMT,0,(dvoid**)0));
/* build the init statement and execute it */
sprintf ((char*)stmbuf, SQLTXT_INIT);
OCIERROR(errhp,OCIStmtPrepare(pctx->curpi, errhp, stmbuf,
strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
OCIERROR(errhp,
OCIStmtExecute(tpcsvc,pctx->curpi,errhp,1,0,0,0,OCI_DEFAULT));
/* customer id != 0, go by last name */
#ifdef ATOMA
sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/paynz_abort.sql", ora_home);
sqlfile(sql_file_name,stmbuf);
/* sqlfile("../../blocks/paynz_abort.sql",stmbuf); */
/* sprintf((char *)stmbuf, SQLTXT_NONZERO); */
#else
sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/paynz.sql", ora_home);
sqlfile(sql_file_name,stmbuf);
/* sqlfile("../../blocks/paynz.sql",stmbuf); */
#endif
OCIERROR(errhp,OCIStmtPrepare(pctx->curp0, errhp, stmbuf,
strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
/* customer id == 0, go by last name */
#ifdef ATOMA
sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/payz_abort.sql", ora_home);
sqlfile(sql_file_name,stmbuf);
/* sqlfile("../../blocks/payz_abort.sql",stmbuf); */
/* sprintf((char *)stmbuf, SQLTXT_ZERO); */
#else
sprintf(sql_file_name, "%s/bench/tpc/tpcc/blocks/payz.sql", ora_home);
sqlfile(sql_file_name,stmbuf);
/* sqlfile("../../blocks/payz.sql",stmbuf); */
#endif
OCIERROR(errhp,OCIStmtPrepare(pctx->curp1, errhp, stmbuf,
strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
pctx->w_id_ind = TRUE;
pctx->w_id_len = SIZ(payP->w_id);
pctx->d_id_ind = TRUE;
pctx->d_id_len = SIZ(payP->d_id);
pctx->c_w_id_ind = TRUE;
pctx->c_w_id_len = SIZ(payP->c_w_id);
pctx->c_d_id_ind = TRUE;
pctx->c_d_id_len = SIZ(payP->c_d_id);
pctx->c_id_ind = TRUE;
pctx->c_id_len = 0;
pctx->h_amount_len = SIZ(payP->h_amount);
pctx->h_amount_ind = TRUE;
pctx->c_last_ind = TRUE;
pctx->c_last_len = 0;
pctx->w_street_1_ind = TRUE;
pctx->w_street_1_len = 0;
pctx->w_street_2_ind = TRUE;
pctx->w_street_2_len = 0;
pctx->w_city_ind = TRUE;
pctx->w_city_len = 0;
pctx->w_state_ind = TRUE;
pctx->w_state_len = 0;
pctx->w_zip_ind = TRUE;
pctx->w_zip_len = 0;
pctx->d_street_1_ind = TRUE;
pctx->d_street_1_len = 0;
pctx->d_street_2_ind = TRUE;
pctx->d_street_2_len = 0;
pctx->d_city_ind = TRUE;
pctx->d_city_len = 0;
pctx->d_state_ind = TRUE;
pctx->d_state_len = 0;
pctx->d_zip_ind = TRUE;
pctx->d_zip_len = 0;
pctx->c_first_ind = TRUE;
pctx->c_first_len = 0;
pctx->c_middle_ind = TRUE;
pctx->c_middle_len = 0;
pctx->c_street_1_ind = TRUE;
pctx->c_street_1_len = 0;
pctx->c_street_2_ind = TRUE;
pctx->c_street_2_len = 0;
pctx->c_city_ind = TRUE;
pctx->c_city_len = 0;
pctx->c_state_ind = TRUE;
pctx->c_state_len = 0;
pctx->c_zip_ind = TRUE;
pctx->c_zip_len = 0;
pctx->c_phone_ind = TRUE;
pctx->c_phone_len = 0;
pctx->c_since_ind = TRUE;
pctx->c_since_len = 0;
pctx->c_credit_ind = TRUE;
pctx->c_credit_len = 0;
pctx->c_credit_lim_ind = TRUE;
pctx->c_credit_lim_len = 0;
pctx->c_discount_ind = TRUE;
pctx->c_discount_len = 0;
pctx->c_balance_ind = TRUE;
pctx->c_balance_len = sizeof(double);
pctx->c_data_ind = TRUE;
pctx->c_data_len = 0;
pctx->h_date_ind = TRUE;
pctx->h_date_len = 0;
pctx->retries_ind = TRUE;
pctx->retries_len = 0;
pctx->cr_date_ind = TRUE;
pctx->cr_date_len = 7;
/* bind variables */
OCIBNDR(pctx->curp0, pctx->w_id_bp, errhp,":w_id",ADR(payP->w_id),SIZ(int),
SQLT_INT, &pctx->w_id_ind, NULL, NULL);
OCIBNDR(pctx->curp0, pctx->d_id_bp, errhp,":d_id",ADR(payP->d_id),SIZ(int),
SQLT_INT, &pctx->d_id_ind, NULL, NULL);
OCIBND(pctx->curp0, pctx->c_w_id_bp, errhp,":c_w_id",ADR(payP->c_w_id),SIZ(int),
SQLT_INT);
OCIBND(pctx->curp0, pctx->c_d_id_bp, errhp,":c_d_id",ADR(payP->c_d_id),SIZ(int),
SQLT_INT);
OCIBND(pctx->curp0, pctx->c_id_bp, errhp,":c_id",ADR(payP->c_id),SIZ(int),
SQLT_INT);
OCIBNDR(pctx->curp0, pctx->h_amount_bp, errhp,":h_amount",ADR(payP->h_amount),
SIZ(int),SQLT_INT, &pctx->h_amount_ind, &pctx->h_amount_len,
&pctx->h_amount_rc);
OCIBNDR(pctx->curp0, pctx->c_last_bp, errhp,":c_last",payP->c_last,SIZ(payP->c_last),
SQLT_STR, &pctx->c_last_ind, &pctx->c_last_len, &pctx->c_last_rc);
OCIBNDR(pctx->curp0, pctx->w_street_1_bp, errhp,":w_street_1",payP->w_street_1,
SIZ(payP->w_street_1),SQLT_STR, &pctx->w_street_1_ind,
&pctx->w_street_1_len, &pctx->w_street_1_rc);
OCIBNDR(pctx->curp0, pctx->w_street_2_bp, errhp,":w_street_2",payP->w_street_2,
SIZ(payP->w_street_2),SQLT_STR, &pctx->w_street_2_ind,
&pctx->w_street_2_len, &pctx->w_street_2_rc);
OCIBNDR(pctx->curp0, pctx->w_city_bp,
errhp,":w_city",payP->w_city,SIZ(payP->w_city),
SQLT_STR, &pctx->w_city_ind, &pctx->w_city_len, &pctx->w_city_rc);
OCIBNDR(pctx->curp0, pctx->w_state_bp,
errhp,":w_state",payP->w_state,SIZ(payP->w_state),
SQLT_STR, &pctx->w_state_ind, &pctx->w_state_len, &pctx->w_state_rc);
OCIBNDR(pctx->curp0, pctx->w_zip_bp, errhp,":w_zip",payP->w_zip,SIZ(payP->w_zip),
SQLT_STR, &pctx->w_zip_ind, &pctx->w_zip_len, &pctx->w_zip_rc);
OCIBNDR(pctx->curp0, pctx->d_street_1_bp, errhp,":d_street_1",payP->d_street_1,
SIZ(payP->d_street_1),SQLT_STR, &pctx->d_street_1_ind,
&pctx->d_street_1_len, &pctx->d_street_1_rc);
OCIBNDR(pctx->curp0, pctx->d_street_2_bp, errhp,":d_street_2",payP->d_street_2,
SIZ(payP->d_street_2),SQLT_STR, &pctx->d_street_2_ind,
&pctx->d_street_2_len, &pctx->d_street_2_rc);
OCIBNDR(pctx->curp0, pctx->d_city_bp, errhp,":d_city",payP->d_city,SIZ(payP->d_city),
SQLT_STR, &pctx->d_city_ind, &pctx->d_city_len, &pctx->d_city_rc);
OCIBNDR(pctx->curp0, pctx->d_state_bp,
errhp,":d_state",payP->d_state,SIZ(payP->d_state),
SQLT_STR, &pctx->d_state_ind, &pctx->d_state_len, &pctx->d_state_rc);
OCIBNDR(pctx->curp0, pctx->d_zip_bp, errhp,":d_zip",payP->d_zip,SIZ(payP->d_zip),
SQLT_STR, &pctx->d_zip_ind, &pctx->d_zip_len, &pctx->d_zip_rc);
OCIBNDR(pctx->curp0, pctx->c_first_bp, errhp,":c_first",payP->c_first,SIZ(payP->c_first),
SQLT_STR, &pctx->c_first_ind, &pctx->c_first_len, &pctx->c_first_rc);
OCIBNDR(pctx->curp0, pctx->c_middle_bp, errhp,":c_middle",payP->c_middle,2,
SQLT_AFC, &pctx->c_middle_ind, &pctx->c_middle_len,
&pctx->c_middle_rc);
OCIBNDR(pctx->curp0, pctx->c_street_1_bp, errhp,":c_street_1",payP->c_street_1,
SIZ(payP->c_street_1),SQLT_STR, &pctx->c_street_1_ind,
&pctx->c_street_1_len, &pctx->c_street_1_rc);
OCIBNDR(pctx->curp0, pctx->c_street_2_bp, errhp,":c_street_2",payP->c_street_2,
SIZ(payP->c_street_2),SQLT_STR, &pctx->c_street_2_ind,
&pctx->c_street_2_len, &pctx->c_street_2_rc);
OCIBNDR(pctx->curp0, pctx->c_city_bp, errhp,":c_city",payP->c_city,SIZ(payP->c_city),
SQLT_STR, &pctx->c_city_ind, &pctx->c_city_len, &pctx->c_city_rc);
OCIBNDR(pctx->curp0, pctx->c_state_bp,
errhp,":c_state",payP->c_state,SIZ(payP->c_state),
SQLT_STR, &pctx->c_state_ind, &pctx->c_state_len, &pctx->c_state_rc);
OCIBNDR(pctx->curp0, pctx->c_zip_bp, errhp,":c_zip",payP->c_zip,SIZ(payP->c_zip),
SQLT_STR, &pctx->c_zip_ind, &pctx->c_zip_len, &pctx->c_zip_rc);
OCIBNDR(pctx->curp0, pctx->c_phone_bp,
errhp,":c_phone",payP->c_phone,SIZ(payP->c_phone), SQLT_STR,
&pctx->c_phone_ind, &pctx->c_phone_len, &pctx->c_phone_rc);
OCIBNDR(pctx->curp0, pctx->c_since_bp, errhp,":c_since",payP->c_since,7,
SQLT_DAT, &pctx->c_since_ind, &pctx->c_since_len, &pctx->c_since_rc);
OCIBNDR(pctx->curp0, pctx->c_credit_bp, errhp,":c_credit",payP->c_credit,
SIZ(payP->c_credit),SQLT_CHR, &pctx->c_credit_ind, &pctx->c_credit_len,
&pctx->c_credit_rc);
OCIBNDR(pctx->curp0, pctx->c_credit_lim_bp, errhp,":c_credit_lim",
ADR(payP->c_credit_lim),SIZ(int), SQLT_INT, &pctx->c_credit_lim_ind,
&pctx->c_credit_lim_len, &pctx->c_credit_lim_rc);
OCIBNDR(pctx->curp0, pctx->c_discount_bp, errhp,":c_discount",
ADR(payP->c_discount),SIZ(int), SQLT_INT, &pctx->c_discount_ind,
&pctx->c_discount_len, &pctx->c_discount_rc);
OCIBNDR(pctx->curp0, pctx->c_balance_bp, errhp,":c_balance",ADR(payP->c_balance),
SIZ(double),SQLT_FLT, &pctx->c_balance_ind, &pctx->c_balance_len,
```

```
&pctx->c_balance_rc);
OCIBNDR(pctx->curp0, pctx->c_data_bp,
errhp,":c_data",payP->c_data,SIZ(payP->c_data),
SQLT_STR, &pctx->c_data_ind, &pctx->c_data_len, &pctx->c_data_rc);
OCIBNDR(pctx->curp0, pctx->h_date_bp,
errhp,":h_date",payP->h_date,SIZ(payP->h_date),
SQLT_STR, &pctx->h_date_ind, &pctx->h_date_len, &pctx->h_date_rc);
OCIBNDR(pctx->curp0, pctx->retries_bp, errhp,":retry",ADR(payP->retries),SIZ(int),
SQLT_INT, &pctx->retries_ind, &pctx->retries_len, &pctx->retries_rc);
OCIBNDR(pctx->curp0, pctx->cr_date_bp, errhp,":cr_date",ADR(payP->cr_date),
SIZ(payP->cr_date),SQLT_DAT, &pctx->cr_date_ind, &pctx->cr_date_len,
&pctx->cr_date_rc);
/* ---- Binds for the second cursor */
OCIBNDR(pctx->curp1, pctx->w_id_bp1, errhp,":w_id",ADR(payP->w_id),SIZ(int),
SQLT_INT, &pctx->w_id_ind, &pctx->w_id_len, &pctx->w_id_rc);
OCIBNDR(pctx->curp1, pctx->d_id_bp1, errhp,":d_id",ADR(payP->d_id),SIZ(int),
SQLT_INT, &pctx->d_id_ind, &pctx->d_id_len, &pctx->d_id_rc);
OCIBND(pctx->curp1, pctx->c_w_id_bp1, errhp,":c_w_id",ADR(payP->c_w_id),SIZ(int),
SQLT_INT);
OCIBND(pctx->curp1, pctx->c_d_id_bp1, errhp,":c_d_id",ADR(payP->c_d_id),SIZ(int),
SQLT_INT);
OCIBNDR(pctx->curp1, pctx->c_id_bp1, errhp,":c_id",ADR(payP->c_id),SIZ(int),
SQLT_INT, &pctx->c_id_ind, &pctx->c_id_len, &pctx->c_id_rc);
OCIBNDR(pctx->curp1, pctx->h_amount_bp1, errhp,":h_amount",ADR(payP->h_amount),
SIZ(int),SQLT_INT, &pctx->h_amount_ind, &pctx->h_amount_len,
&pctx->h_amount_rc);
OCIBND(pctx->curp1, pctx->c_last_bp1, errhp,":c_last",payP->c_last,SIZ(payP->c_last),
SQLT_STR);
OCIBNDR(pctx->curp1, pctx->w_street_1_bp1, errhp,":w_street_1",payP->w_street_1,
SIZ(payP->w_street_1),SQLT_STR, &pctx->w_street_1_ind,
&pctx->w_street_1_len, &pctx->w_street_1_rc);
OCIBNDR(pctx->curp1, pctx->w_street_2_bp1, errhp,":w_street_2",payP->w_street_2,
SIZ(payP->w_street_2),SQLT_STR, &pctx->w_street_2_ind,
&pctx->w_street_2_len, &pctx->w_street_2_rc);
OCIBNDR(pctx->curp1, pctx->w_city_bp1,
errhp,":w_city",payP->w_city,SIZ(payP->w_city),
SQLT_STR, &pctx->w_city_ind, &pctx->w_city_len, &pctx->w_city_rc);
OCIBNDR(pctx->curp1, pctx->w_state_bp1,
errhp,":w_state",payP->w_state,SIZ(payP->w_state),
SQLT_STR, &pctx->w_state_ind, &pctx->w_state_len, &pctx->w_state_rc);
OCIBNDR(pctx->curp1, pctx->w_zip_bp1, errhp,":w_zip",payP->w_zip,SIZ(payP->w_zip),
SQLT_STR, &pctx->w_zip_ind, &pctx->w_zip_len, &pctx->w_zip_rc);
OCIBNDR(pctx->curp1, pctx->d_street_1_bp1, errhp,":d_street_1",payP->d_street_1,
SIZ(payP->d_street_1),SQLT_STR, &pctx->d_street_1_ind,
&pctx->d_street_1_len, &pctx->d_street_1_rc);
OCIBNDR(pctx->curp1, pctx->d_street_2_bp1, errhp,":d_street_2",payP->d_street_2,
SIZ(payP->d_street_2),SQLT_STR, &pctx->d_street_2_ind,
&pctx->d_street_2_len, &pctx->d_street_2_rc);
OCIBNDR(pctx->curp1, pctx->d_city_bp1,
errhp,":d_city",payP->d_city,SIZ(payP->d_city),
SQLT_STR, &pctx->d_city_ind, &pctx->d_city_len, &pctx->d_city_rc);
OCIBNDR(pctx->curp1, pctx->d_state_bp1, errhp,":d_state",payP->d_state,
SIZ(payP->d_state), SQLT_STR, &pctx->d_state_ind, &pctx->d_state_len,
&pctx->d_state_rc);
OCIBNDR(pctx->curp1, pctx->d_zip_bp1, errhp,":d_zip",payP->d_zip,SIZ(payP->d_zip),
SQLT_STR, &pctx->d_zip_ind, &pctx->d_zip_len, &pctx->d_zip_rc);
OCIBNDR(pctx->curp1, pctx->c_first_bp1, errhp,":c_first",payP->c_first,
SIZ(payP->c_first), SQLT_STR, &pctx->c_first_ind, &pctx->c_first_len,
&pctx->c_first_rc);
OCIBNDR(pctx->curp1, pctx->c_middle_bp1, errhp,":c_middle",payP->c_middle,2,
SQLT_AFC, &pctx->c_middle_ind, &pctx->c_middle_len,
&pctx->c_middle_rc);
OCIBNDR(pctx->curp1, pctx->c_street_1_bp1, errhp,":c_street_1",payP->c_street_1,
SIZ(payP->c_street_1),SQLT_STR, &pctx->c_street_1_ind,
&pctx->c_street_1_len, &pctx->c_street_1_rc);
OCIBNDR(pctx->curp1, pctx->c_street_2_bp1, errhp,":c_street_2",payP->c_street_2,
SIZ(payP->c_street_2),SQLT_STR, &pctx->c_street_2_ind,
&pctx->c_street_2_len, &pctx->c_street_2_rc);
OCIBNDR(pctx->curp1, pctx->c_city_bp1, errhp,":c_city",payP->c_city,
SIZ(payP->c_city),SQLT_STR,
&pctx->c_city_ind, &pctx->c_city_len, &pctx->c_city_rc);
OCIBNDR(pctx->curp1, pctx->c_state_bp1, errhp,":c_state",payP->c_state,
SIZ(payP->c_state),SQLT_STR, &pctx->c_state_ind, &pctx->c_state_len,
&pctx->c_state_rc);
OCIBNDR(pctx->curp1, pctx->c_zip_bp1, errhp,":c_zip",payP->c_zip,SIZ(payP->c_zip),
SQLT_STR, &pctx->c_zip_ind, &pctx->c_zip_len, &pctx->c_zip_rc);
OCIBNDR(pctx->curp1, pctx->c_phone_bp1, errhp,":c_phone",payP->c_phone,
SIZ(payP->c_phone), SQLT_STR, &pctx->c_phone_ind, &pctx->c_phone_len,
&pctx->c_phone_rc);
OCIBNDR(pctx->curp1, pctx->c_since_bp1, errhp,":c_since",payP->c_since,7,
SQLT_DAT, &pctx->c_since_ind, &pctx->c_since_len, &pctx->c_since_rc);
OCIBNDR(pctx->curp1, pctx->c_credit_bp1, errhp,":c_credit",payP->c_credit,
SIZ(payP->c_credit),SQLT_CHR, &pctx->c_credit_ind, &pctx->c_credit_len,
&pctx->c_credit_rc);
OCIBNDR(pctx->curp1, pctx->c_credit_lim_bp1, errhp,":c_credit_lim",
ADR(payP->c_credit_lim),SIZ(int), SQLT_INT, &pctx->c_credit_lim_ind,
&pctx->c_credit_lim_len, &pctx->c_credit_lim_rc);
OCIBNDR(pctx->curp1, pctx->c_discount_bp1, errhp,":c_discount",
ADR(payP->c_discount),SIZ(int), SQLT_INT, &pctx->c_discount_ind,
&pctx->c_discount_len, &pctx->c_discount_rc);
OCIBNDR(pctx->curp1, pctx->c_balance_bp1,
errhp,":c_balance",ADR(payP->c_balance),
SIZ(double),SQLT_FLT, &pctx->c_balance_ind, &pctx->c_balance_len,
&pctx->c_balance_rc);
OCIBNDR(pctx->curp1, pctx->c_data_bp1,
errhp,":c_data",payP->c_data,SIZ(payP->c_data),
SQLT_STR, &pctx->c_data_ind, &pctx->c_data_len, &pctx->c_data_rc);
OCIBNDR(pctx->curp1, pctx->h_date_bp1,
errhp,":h_date",payP->h_date,SIZ(payP->h_date),
SQLT_STR, &pctx->h_date_ind, &pctx->h_date_len, &pctx->h_date_rc);
OCIBNDR(pctx->curp1, pctx->retries_bp1, errhp,":retry",ADR(payP->retries),SIZ(int),
SQLT_INT, &pctx->retries_ind, &pctx->retries_len, &pctx->retries_rc);
OCIBNDR(pctx->curp1, pctx->cr_date_bp1, errhp,":cr_date",ADR(payP->cr_date),
SIZ(payP->cr_date),SQLT_DAT, &pctx->cr_date_ind, &pctx->cr_date_len,
&pctx->cr_date_rc);
return (0);
}
plpay (ora_cn_data_t *ora_SlotDataP)
{
payctx *pctx = ora_SlotDataP->pctx;
global_payment_t *payP = ora_SlotDataP->payP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
retry:
pctx->w_id_ind = TRUE;
pctx->w_id_len = SIZ(payP->w_id);
pctx->d_id_ind = TRUE;
pctx->d_id_len = SIZ(payP->d_id);
pctx->c_w_id_ind = TRUE;
pctx->c_w_id_len = 0;
pctx->c_d_id_ind = TRUE;
pctx->c_d_id_len = 0;
pctx->c_id_ind = TRUE;
pctx->c_id_len = 0;
pctx->h_amount_len = SIZ(payP->h_amount);
pctx->h_amount_ind = TRUE;
pctx->c_last_ind = TRUE;
pctx->c_last_len = SIZ(payP->c_last);
pctx->w_street_1_ind = TRUE;
pctx->w_street_1_len = 0;
pctx->w_street_2_ind = TRUE;
pctx->w_street_2_len = 0;
pctx->w_city_ind = TRUE;
pctx->w_city_len = 0;
pctx->w_state_ind = TRUE;
pctx->w_state_len = 0;
pctx->w_zip_ind = TRUE;
pctx->w_zip_len = 0;
pctx->d_street_1_ind = TRUE;
pctx->d_street_1_len = 0;
pctx->d_street_2_ind = TRUE;
pctx->d_street_2_len = 0;
pctx->d_city_ind = TRUE;
pctx->d_city_len = 0;
pctx->d_state_ind = TRUE;
pctx->d_state_len = 0;
pctx->d_zip_ind = TRUE;
pctx->d_zip_len = 0;
pctx->c_first_ind = TRUE;
pctx->c_first_len = 0;
pctx->c_middle_ind = TRUE;
pctx->c_middle_len = 0;
pctx->c_street_1_ind = TRUE;
pctx->c_street_1_len = 0;
pctx->c_street_2_ind = TRUE;
pctx->c_street_2_len = 0;
pctx->c_city_ind = TRUE;
pctx->c_city_len = 0;
pctx->c_state_ind = TRUE;
pctx->c_state_len = 0;
pctx->c_zip_ind = TRUE;
pctx->c_zip_len = 0;
pctx->c_phone_ind = TRUE;
pctx->c_phone_len = 0;
pctx->c_since_ind = TRUE;
pctx->c_since_len = 0;
pctx->c_credit_ind = TRUE;
pctx->c_credit_len = 0;
pctx->c_credit_lim_ind = TRUE;
pctx->c_credit_lim_len = 0;
pctx->c_discount_ind = TRUE;
pctx->c_discount_len = 0;
pctx->c_balance_ind = TRUE;
pctx->c_balance_len = sizeof(double);
pctx->c_data_ind = TRUE;
pctx->c_data_len = 0;
pctx->h_date_ind = TRUE;
pctx->h_date_len = 0;
pctx->retries_ind = TRUE;
pctx->retries_len = 0;
pctx->cr_date_ind = TRUE;
pctx->cr_date_len = 7;
if(payP->bylastname) {
payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp1,errhp,1,0,0,0,OCI_DEFAULT);
} else {
payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp0,errhp,1,0,0,0,OCI_DEFAULT);
}
if(payP->execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
payP->errcode = OCIERROR(errhp,payP->execstatus);
if(payP->errcode == NOT_SERIALIZABLE) {
payP->retries++;
goto retry;
} else if (payP->errcode == RECOVERR) {
payP->retries++;
goto retry;
} else {
return -1;
}
}
return (0);
}
void plpaydone(ora_cn_data_t *ora_SlotDataP)
{
/* TODO: Should we free the cursor handles?? */
if(ora_SlotDataP->pctx) {
free(ora_SlotDataP->pctx);
ora_SlotDataP->pctx = NULL;
}
if (ora_SlotDataP->payP) {
free(ora_SlotDataP->payP);
ora_SlotDataP->payP = NULL;
}
}
```

plpay.h

```
#ifndef TPCC_PLPAY_H
#define TPCC_PLPAY_H
#include <oci.h>
struct payctx {
OCIStmt *curpi;
OCIStmt *curp0;
OCIStmt *curp1;
OCIBind *w_id_bp;
OCIBind *w_id_bp1;
sb2 w_id_ind;
ub2 w_id_len;
ub2 w_id_rc;
OCIBind *d_id_bp;
OCIBind *d_id_bp1;
sb2 d_id_ind;
ub2 d_id_len;
ub2 d_id_rc;
OCIBind *c_w_id_bp;
OCIBind *c_w_id_bp1;
sb2 c_w_id_ind;
ub2 c_w_id_len;
ub2 c_w_id_rc;
OCIBind *c_d_id_bp;
OCIBind *c_d_id_bp1;
sb2 c_d_id_ind;
ub2 c_d_id_len;
ub2 c_d_id_rc;
OCIBind *c_id_bp;
OCIBind *c_id_bp1;
sb2 c_id_ind;
ub2 c_id_len;
ub2 c_id_rc;
OCIBind *h_amount_bp;
OCIBind *h_amount_bp1;
sb2 h_amount_ind;
ub2 h_amount_len;
ub2 h_amount_rc;
OCIBind *c_last_bp;
OCIBind *c_last_bp1;
sb2 c_last_ind;
ub2 c_last_len;
ub2 c_last_rc;
OCIBind *w_street_1_bp;
OCIBind *w_street_1_bp1;
sb2 w_street_1_ind;
ub2 w_street_1_len;
ub2 w_street_1_rc;
OCIBind *w_street_2_bp;
OCIBind *w_street_2_bp1;
sb2 w_street_2_ind;
ub2 w_street_2_len;
ub2 w_street_2_rc;
OCIBind *w_city_bp;
OCIBind *w_city_bp1;
sb2 w_city_ind;
ub2 w_city_len;
ub2 w_city_rc;
OCIBind *w_state_bp;
OCIBind *w_state_bp1;
sb2 w_state_ind;
ub2 w_state_len;
ub2 w_state_rc;
OCIBind *w_zip_bp;
OCIBind *w_zip_bp1;
sb2 w_zip_ind;
ub2 w_zip_len;
ub2 w_zip_rc;
OCIBind *d_street_1_bp;
OCIBind *d_street_1_bp1;
sb2 d_street_1_ind;
ub2 d_street_1_len;
ub2 d_street_1_rc;
OCIBind *d_street_2_bp;
OCIBind *d_street_2_bp1;
sb2 d_street_2_ind;
ub2 d_street_2_len;
ub2 d_street_2_rc;
OCIBind *d_city_bp;
OCIBind *d_city_bp1;
sb2 d_city_ind;
ub2 d_city_len;
ub2 d_city_rc;
OCIBind *d_state_bp;
OCIBind *d_state_bp1;
sb2 d_state_ind;
ub2 d_state_len;
ub2 d_state_rc;
OCIBind *d_zip_bp;
OCIBind *d_zip_bp1;
sb2 d_zip_ind;
ub2 d_zip_len;
ub2 d_zip_rc;
OCIBind *c_first_bp;
OCIBind *c_first_bp1;
sb2 c_first_ind;
ub2 c_first_len;
ub2 c_first_rc;
OCIBind *c_middle_bp;
OCIBind *c_middle_bp1;
sb2 c_middle_ind;
ub2 c_middle_len;
ub2 c_middle_rc;
OCIBind *c_street_1_bp;
OCIBind *c_street_1_bp1;
sb2 c_street_1_ind;
ub2 c_street_1_len;
ub2 c_street_1_rc;
OCIBind *c_street_2_bp;
OCIBind *c_street_2_bp1;
sb2 c_street_2_ind;
ub2 c_street_2_len;
ub2 c_street_2_rc;
OCIBind *c_city_bp;
OCIBind *c_city_bp1;
sb2 c_city_ind;
ub2 c_city_len;
ub2 c_city_rc;
OCIBind *c_state_bp;
OCIBind *c_state_bp1;
sb2 c_state_ind;
ub2 c_state_len;
ub2 c_state_rc;
OCIBind *c_zip_bp;
OCIBind *c_zip_bp1;
sb2 c_zip_ind;
ub2 c_zip_len;
ub2 c_zip_rc;
OCIBind *c_phone_bp;
OCIBind *c_phone_bp1;
sb2 c_phone_ind;
ub2 c_phone_len;
ub2 c_phone_rc;
OCIBind *c_since_bp;
OCIBind *c_since_bp1;
sb2 c_since_ind;
ub2 c_since_len;
ub2 c_since_rc;
OCIBind *c_credit_bp;
OCIBind *c_credit_bp1;
sb2 c_credit_ind;
ub2 c_credit_len;
ub2 c_credit_rc;
OCIBind *c_credit_lim_bp;
OCIBind *c_credit_lim_bp1;
sb2 c_credit_lim_ind;
ub2 c_credit_lim_len;
ub2 c_credit_lim_rc;
OCIBind *c_discount_bp;
OCIBind *c_discount_bp1;
sb2 c_discount_ind;
ub2 c_discount_len;
ub2 c_discount_rc;
OCIBind *c_balance_bp;
OCIBind *c_balance_bp1;
sb2 c_balance_ind;
ub2 c_balance_len;
ub2 c_balance_rc;
OCIBind *c_data_bp;
OCIBind *c_data_bp1;
sb2 c_data_ind;
ub2 c_data_len;
ub2 c_data_rc;
OCIBind *h_date_bp;
OCIBind *h_date_bp1;
sb2 h_date_ind;
ub2 h_date_len;
ub2 h_date_rc;
OCIBind *retries_bp;
OCIBind *retries_bp1;
sb2 retries_ind;
ub2 retries_len;
ub2 retries_rc;
OCIBind *cr_date_bp;
OCIBind *cr_date_bp1;
sb2 cr_date_ind;
ub2 cr_date_len;
ub2 cr_date_rc;
OCIBind *byln_bp;
sb2 byln_ind;
ub2 byln_len;
ub2 byln_rc;
};
typedef struct payctx payctx;
struct global_payment_t {
int w_id;
int d_id;
int c_id;
char c_last[17];
char c_first[17];
char c_middle[3];
double c_balance;
int retries;
int bylastname;
unsigned char c_since[7];
int execstatus;
int errcode;
int c_w_id;
int c_d_id;
int h_amount;
char w_street_1[21];
char w_street_2[21];
char w_city[21];
char w_state[3];
char w_zip[10];
char d_street_1[21];
char d_street_2[21];
char d_city[21];
char d_state[3];
char d_zip[10];
char c_street_1[21];
char c_street_2[21];
char c_city[21];
char c_state[3];
char c_zip[10];
char c_phone[17];
char c_since_d[11];
```

```
int c_discount;
char c_credit[3];
int c_credit_lim;
char c_data[201];
char h_date[20];
unsigned char cr_date[7];
};
typedef struct global_payment_t global_payment_t;
#endif /* TPCC_PLPAY_H */
```

plsto.c

```
#ifdef RCSID
static char *RCSid =
"$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/plsto.c,v 1.2 1998/01/23 15:08:16 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */
/*+=================================================================+
| Copyright (c) 1994 Oracle Corp, Redwood Shores, CA |
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=================================================================+
| FILENAME
| plsto.c
| DESCRIPTION
| OCI version of STOCK LEVEL transaction in TPC-C benchmark.
+=================================================================*/
#include "tpcc.h"
#include "tpccpl.h"
#include "plora.h"
#define SQLTXT "SELECT count (DISTINCT s_i_id) \
FROM order_line, stock, district \
WHERE d_id = :d_id AND d_w_id = :w_id AND \
d_id = ol_d_id AND d_w_id = ol_w_id AND \
ol_i_id = s_i_id AND ol_w_id = s_w_id AND \
s_quantity < :threshold AND \
ol_o_id BETWEEN (d_next_o_id - 20) AND (d_next_o_id - 1)"
#define SQLTXTTEST "BEGIN stocklevel.getstocklevel (:w_id, :d_id, \
:threshold); END;"
plstoinit (ora_cn_data_t *ora_SlotDataP)
{
stoctx *sctx;
global_stock_t *stoP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
text stmbuf[SQL_BUF_SIZE];
sctx = (stoctx *)malloc(sizeof(stoctx));
memset(sctx,(char)0,sizeof(stoctx));
ora_SlotDataP->sctx = sctx;
ora_SlotDataP->stoP = (global_stock_t *)malloc(sizeof(global_stock_t));
memset(ora_SlotDataP->stoP,(char)0,sizeof(global_stock_t));
stoP = ora_SlotDataP->stoP;
sctx->norow=0;
OCIERROR(errhp,
OCIHandleAlloc(tpcenv,(dvoid**)&sctx->curs,OCI_HTYPE_STMT,0,(dvoid**)0));
sprintf ((char *) stmbuf, SQLTXT);
OCIERROR(errhp,OCIStmtPrepare(sctx->curs,errhp,stmbuf,strlen((char *)stmbuf),
OCI_NTV_SYNTAX,OCI_DEFAULT));
OCIERROR(errhp,
OCIAttrSet(sctx->curs,OCI_HTYPE_STM,(dvoid*)&sctx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
/* bind variables */
OCIBND(sctx->curs,sctx->w_id_bp,errhp, ":w_id", ADR(stoP->w_id),sizeof(int),
SQLT_INT);
OCIBND(sctx->curs,sctx->d_id_bp,errhp, ":d_id", ADR(stoP->d_id),sizeof(int),
SQLT_INT);
OCIBND(sctx->curs,sctx->threshold_bp,errhp, ":threshold", ADR(stoP->threshold),
sizeof(int),SQLT_INT);
OCIDEFINE(sctx->curs,sctx->low_stock_bp,errhp, 1, ADR(stoP->low_stock),
sizeof(int), SQLT_INT);
return (0);
}
plsto (ora_cn_data_t *ora_SlotDataP)
{
stoctx *sctx = ora_SlotDataP->sctx;
global_stock_t *stoP = ora_SlotDataP->stoP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;
retry:
stoP->execstatus=
OCIStmtExecute(tpcsvc,sctx->curs,errhp,1,0,0,0,
OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
if(stoP->execstatus != OCI_SUCCESS) {
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
stoP->errcode = OCIERROR(errhp,stoP->execstatus);
if(stoP->errcode == NOT_SERIALIZABLE) {
stoP->retries++;
goto retry;
} else if (stoP->errcode == RECOVERR) {
stoP->retries++;
goto retry;
} else {
return -1;
}
}
return (0);
}
void plstodone (ora_cn_data_t *ora_SlotDataP)
{
stoctx *sctx = ora_SlotDataP->sctx;
if (sctx) {
free(sctx);
ora_SlotDataP->sctx = NULL;
}
if (ora_SlotDataP->stoP) {
free(ora_SlotDataP->stoP);
ora_SlotDataP->stoP = NULL;
}
}
```

plsto.h

```
#ifndef TPCC_PLSTO_H
#define TPCC_PLSTO_H
struct stoctx {
OCIStmt *curs;
OCIBind *w_id_bp;
OCIBind *d_id_bp;
OCIBind *threshold_bp;
OCIDefine *low_stock_bp;
int norow;
};
typedef struct stoctx stoctx;
struct global_stock_t {
int w_id;
int d_id;
int threshold;
int retries;
int low_stock;
int errcode;
int execstatus;
};
typedef struct global_stock_t global_stock_t;
#endif
```

tpcc.h

```
/*
* $Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/tpcc.h,v 1.2 1998/01/23 15:08:17 oz Exp $ Copyr (c) 1993 Oracle
*/
/*+=================================================================+
| Copyright (c) 1995 Oracle Corp, Redwood Shores, CA |
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=================================================================+
| FILENAME
| tpcc.h
| DESCRIPTION
| Include file for TPC-C benchmark programs.
+=================================================================*/
#ifndef TPCC_H
#define TPCC_H
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>
/*
#ifdef __STDC__
#include "ociapr.h"
#else
#include "ocikpr.h"
#endif
*/
typedef struct cda_def csrdef;
typedef struct cda_def ldadef;
/* TPC-C transaction functions */
extern int TPCinit ();
extern int TPCnew ();
extern int TPCpay ();
extern int TPCord ();
extern int TPCdel ();
extern int TPCsto ();
extern int TPCexit ();
extern int TPCdumpinit ();
extern int TPCdumpnew ();
extern int TPCdumppay ();
extern int TPCdumpord ();
extern int TPCdumpdel ();
extern int TPCdumpsto ();
extern int TPCdumpexit ();
/* Error codes */
#define RECOVERR -10
#define IRRECERR -20
#define NOERR 111
#define DEL_ERROR -666
#define DEL_DATE_LEN 7
#define NDISTS 10
#define NITEMS 15
#define SQL_BUF_SIZE 8192
#endif
```

tpcc_info.h

```
/*
* $Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/tpcc_info.h,v 1.2 1998/01/23 15:08:18 oz Exp $ Copyr (c) 1995 Oracle
*/
/*+=================================================================+
| Copyright (c) 1995 Oracle Corp, Redwood Shores, CA |
```

```
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=================================================================+
| FILENAME
| tpcc_info.h
| DESCRIPTION
| Include file for TPC-C benchmark programs.
+=================================================================*/
#ifndef TPCC_INFO_H
#define TPCC_INFO_H
/* New order */
struct newinstruct {
int w_id;
int d_id;
int c_id;
int ol_i_id[15];
int ol_supply_w_id[15];
int ol_quantity[15];
};
struct newoutstruct {
int terror;
int o_id;
int o_ol_cnt;
char c_last[17];
char c_credit[3];
float c_discount;
float w_tax;
float d_tax;
char o_entry_d[20];
float total_amount;
char i_name[15][25];
int s_quantity[15];
char brand_generic[15];
float i_price[15];
float ol_amount[15];
char status[26];
int retry;
};
struct newstruct {
struct newinstruct newin;
struct newoutstruct newout;
};
/* Payment */
struct payinstruct {
int w_id;
int d_id;
int c_w_id;
int c_d_id;
int c_id;
int bylastname;
int h_amount;
char c_last[17];
};
struct payoutstruct {
int terror;
char w_street_1[21];
char w_street_2[21];
char w_city[21];
char w_state[3];
char w_zip[10];
char d_street_1[21];
char d_street_2[21];
char d_city[21];
char d_state[3];
char d_zip[10];
int c_id;
char c_first[17];
char c_middle[3];
char c_last[17];
char c_street_1[21];
char c_street_2[21];
char c_city[21];
char c_state[3];
char c_zip[10];
char c_phone[17];
char c_since[11];
char c_credit[3];
double c_credit_lim;
float c_discount;
double c_balance;
char c_data[201];
char h_date[20];
int retry;
};
struct paystruct {
struct payinstruct payin;
struct payoutstruct payout;
};
/* Order status */
struct ordinstruct {
int w_id;
int d_id;
int c_id;
int bylastname;
char c_last[17];
};
struct ordoutstruct {
int terror;
int c_id;
char c_last[17];
char c_first[17];
char c_middle[3];
double c_balance;
int o_id;
char o_entry_d[20];
int o_carrier_id;
int o_ol_cnt;
int ol_supply_w_id[15];
int ol_i_id[15];
int ol_quantity[15];
float ol_amount[15];
char ol_delivery_d[15][11];
int retry;
};
struct ordstruct {
struct ordinstruct ordin;
struct ordoutstruct ordout;
};
/* Delivery */
struct delinstruct {
int w_id;
int o_carrier_id;
double qtime;
int in_timing_int;
};
struct deloutstruct {
int terror;
int retry;
};
struct delstruct {
struct delinstruct delin;
struct deloutstruct delout;
};
/* Stock level */
struct stoinstruct {
int w_id;
int d_id;
int threshold;
};
struct stooutstruct {
int terror;
int low_stock;
int retry;
};
struct stostruct {
struct stoinstruct stoin;
struct stooutstruct stoout;
};
#endif
```

tpccpl.c

```
#ifdef RCSID
static char *RCSid =
"$Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/tpccpl.c,v 1.3 1998/01/24 14:17:05 oz Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */
/* For now: Preallocate all the connections that we will need */
#if 1
#define PREALLOC_CN
#endif
/*=================================================================+
| Copyright (c) 1994 Oracle Corp, Redwood Shores, CA |
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=================================================================+
| FILENAME
| tpccpl.c
| DESCRIPTION
| TPC-C transactions in PL/SQL.
+=================================================================*/
#ifdef ENCINA
#include <dce/pthread.h>
#define TRACE_WITHOUT_TPP 1
#include <utils/trace.h>
#endif
#include <stdio.h>
#include <time.h>
#include "tpcc.h"
#include "tpcc_info.h"
#include "tpccpl.h"
#include "plora.h"
#define SQLTXT "alter session set isolation_level = serializable"
#ifdef SQL_TRACE
#define SQLTXT1 "alter session set sql_trace = true"
#endif
void print_rt_avg(total_tran_count_t *curP, total_tran_count_t *prevP);
static void check_threads(total_tran_count_t *tran_ctP);
#define PRINT_AV(total, num, str) \
{ \
if ((num) > 0) { \
fprintf(stderr, " %s %.3f,", str, (double)(total)/(num)); \
} \
}
/* ORA_RECONNECT_THRESHHOLD:
* Try reconnecting after this many consecutive errors.
* MIN_TIME_BETWEEN_RECONNECTS
* Min time in seconds to elapse between the last connection time before
* we are allowed to try and reconnect again.
*/
#define ORA_RECONNECT_THRESHHOLD 20
#define MIN_TIME_BETWEEN_RECONNECTS 20
#ifdef ENCINA
/* thread slot data that contains OCI handles and thread specific vars */
pthread_key_t thread_key;
int key_init = 0;
pthread_mutex_t key_lock;
pthread_mutex_t init_lock;
pthread_mutex_t dvry_log_lock;
#else
ora_cn_data_t *connectionP = NULL;
#endif
static char delivery_file_name[80];
FILE *lfp;
FILE *fopen ();
double gettime ();
int proc_no = 0;
```

```
char *TPC_uid, *TPC_pwd;
static void init_global_data(void);
#ifdef ENCINA
static ora_cn_data_t *get_cn(int tran, void *dataP);
static void done_with_cn(ora_cn_data_t *cnP, int tran, void *dataP);
#ifdef PREALLOC_CN
ora_cn_data_t *cn_array = NULL;
int num_connections = 0;
int cn_id = 0;
#endif
#else
#define get_cn(a,b) connectionP
#define done_with_cn(c,a,b) connectionP
#endif
static void init_cn_data(ora_cn_data_t *dataP);
static void clean_cn(void *ptr);
errrpt (lda, cur)
ldadef *lda;
csrdef *cur;
{
text msg[2048];
if (cur->rc) {
oerhms (lda, cur->rc, msg, 2048);
fprintf (stderr, "Error in TPC-C server %d: %s\n", proc_no, msg);
}
if ((cur->rc == DEADLOCK) || (cur->rc == SNAPSHOT_TOO_OLD))
return (RECOVERR);
else
return (IRRECERR);
}
/* vmm313 void ocierror(fname, lineno, errhp, status) */
int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
text errbuf[512];
ub4 buflen;
sb4 errcode;
switch (status) {
case OCI_SUCCESS:
break;
case OCI_SUCCESS_WITH_INFO:
(void) err_printf("Module %s Line %d\n", fname, lineno);
(void) err_printf("Error - OCI_SUCCESS_WITH_INFO\n");
break;
case OCI_NEED_DATA:
(void) err_printf("Module %s Line %d\n", fname, lineno);
(void) err_printf("Error - OCI_NEED_DATA\n");
break;
case OCI_NO_DATA:
(void) err_printf("Module %s Line %d\n", fname, lineno);
(void) err_printf("Error - OCI_NO_DATA\n");
return IRRECERR;
break;
case OCI_ERROR:
(void) OCIErrorGet (errhp, (ub4) 1,
(text *) NULL, &errcode, errbuf,
(ub4) sizeof(errbuf), OCI_HTYPE_ERR);
if (errcode != 8177)
{
(void) err_printf("Module %s Line %d\n", fname, lineno);
(void) err_printf("Warning - %s\n", errbuf);
}
if ( errcode == SNAPSHOT_TOO_OLD ) {
return RECOVERR;
}
if ( errcode == DEADLOCK ) {
return RECOVERR;
}
return errcode;
/* vmm313 TPCexit(1); */
/* vmm313 exit(1); */
break;
case OCI_INVALID_HANDLE:
(void) err_printf("Module %s Line %d\n", fname, lineno);
(void) err_printf("Error - OCI_INVALID_HANDLE\n");
TPCexit(1);
exit(-1);
break;
case OCI_STILL_EXECUTING:
(void) err_printf("Module %s Line %d\n", fname, lineno);
(void) err_printf("Error - OCI_STILL_EXECUTE\n");
break;
case OCI_CONTINUE:
(void) err_printf("Module %s Line %d\n", fname, lineno);
(void) err_printf("Error - OCI_CONTINUE\n");
break;
default:
break;
}
return RECOVERR;
}
FILE *vopen(fnam,mode)
char *fnam;
char *mode;
{
FILE *fd;
#ifdef DEBUG
fprintf(stderr, "tkvuopen() fnam: %s, mode: %s\n", fnam, mode);
#endif
fd = fopen((char *)fnam,(char *)mode);
if (!fd){
fprintf(stderr," fopen on %s failed %d\n",fnam,fd);
exit(-1);
}
return(fd);
}
int sqlfile(fnam,linebuf)
char *fnam;
text *linebuf;
{
FILE *fd;
int nulpt = 0;
#ifdef DEBUG
fprintf(stderr, "sqlfile() fnam: %s, linebuf: %#x\n", fnam, linebuf);
#endif
fd = vopen(fnam,"r");
while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
{
nulpt = strlen((char *)linebuf);
}
return(nulpt);
}
/*
void
vgetdate(unsigned char *buf) {
time_t tloc;
char temp[5];
int cen;
if(time(&tloc) == (time_t)-1) {
err_printf("Error getting date\n");
exit(1);
}
cftime(temp,"%d",&tloc);
buf[3] = (unsigned char)atoi(temp);
cftime(temp,"%m",&tloc);
buf[2] = (unsigned char)atoi(temp);
cftime(temp,"%Y",&tloc);
cen = atoi(temp);
buf[0] = (unsigned char)((cen/100)+100);
buf[1] = (unsigned char)((cen%100)+100);
cftime(temp,"%H",&tloc);
buf[4] = (unsigned char)(atoi(temp) + 1);
cftime(temp,"%M",&tloc);
buf[5] = (unsigned char)(atoi(temp) + 1);
cftime(temp,"%S",&tloc);
buf[6] = (unsigned char)(atoi(temp) + 1);
}
*/
void vgetdate (unsigned char *oradt)
{
struct tm timebuf;
struct tm *loctime = &timebuf;
time_t int_time;
struct ORADATE {
unsigned char century;
unsigned char year;
unsigned char month;
unsigned char day;
unsigned char hour;
unsigned char minute;
unsigned char second;
} Date;
int century;
int cnvrtOK;
/* assume convert is successful */
cnvrtOK = 1;
/* get the current date and time as an integer */
time( &int_time);
/* Convert the current date and time into local time */
localtime_r(&int_time, &timebuf);
century = (1900+loctime->tm_year) / 100;
Date.century = (unsigned char)(century + 100);
if (Date.century < 119 || Date.century > 120) cnvrtOK = 0;
Date.year = (unsigned char)(loctime->tm_year+100);
if (Date.year < 100 || Date.year > 199) cnvrtOK = 0;
Date.month = (unsigned char)(loctime->tm_mon + 1);
if (Date.month < 1 || Date.month > 12) cnvrtOK = 0;
Date.day = (unsigned char)loctime->tm_mday;
if (Date.day < 1 || Date.day > 31) cnvrtOK = 0;
Date.hour = (unsigned char)(loctime->tm_hour + 1);
if (Date.hour < 1 || Date.hour > 24) cnvrtOK = 0;
Date.minute= (unsigned char)(loctime->tm_min + 1);
if (Date.minute < 1 || Date.minute > 60) cnvrtOK = 0;
Date.second= (unsigned char)(loctime->tm_sec + 1);
if (Date.second < 1 || Date.second > 60) cnvrtOK = 0;
if (cnvrtOK)
memcpy(oradt,&Date,7);
else
*oradt = '\0';
return;
}
void cvtdmy (unsigned char *oradt, char *outdate)
{
struct ORADATE {
unsigned char century;
unsigned char year;
unsigned char month;
unsigned char day;
unsigned char hour;
unsigned char minute;
unsigned char second;
} Date;
int day,month,year;
memcpy(&Date,oradt,7);
year = (Date.century-100)*100 + Date.year-100;
month = Date.month;
day = Date.day;
/* sprintf(outdate,"%02d-%02d-%4d\0",day,month,year); */
sprintf(outdate,"%02d-%02d-%4d",day,month,year);
return;
}
void cvtdmyhms (unsigned char *oradt, char *outdate)
{
struct ORADATE {
unsigned char century;
unsigned char year;
unsigned char month;
```

```
unsigned char day;
unsigned char hour;
unsigned char minute;
unsigned char second;
} Date;
int day,month,year;
int hour,min,sec;
memcpy(&Date,oradt,7);
year = (Date.century-100)*100 + Date.year-100;
month = Date.month;
day = Date.day;
hour = Date.hour - 1;
min = Date.minute - 1;
sec = Date.second - 1;
/*sprintf(outdate,"%02d-%02d-%4d %02d:%02d:%02d\0", */
sprintf(outdate,"%02d-%02d-%4d %02d:%02d:%02d",
day,month,year,hour,min,sec);
return;
}
/* Each server may have multiple connections to the DB.
* The OCI handles, which used to be global, are now grouped together
* In a data structure. There is one such structure per DB connection.
*
* There are two routines to deal with them:
* initOCIhandles: Initializes all the handles a connection needs
* freeOCIhandles: Frees all those handles.
* When the program is initialized it initializes an array of connections.
* Each thread is then assigned one of these connections and keeps reusing
* that connection. The same connection may be shared by multiple threads.
*/
static void initOCIhandles(ora_cn_data_t *cn_dataP)
{
text stmbuf[SQL_BUF_SIZE];
OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
OCIStmt *curi;
dvoid *xmem;
/* Initialize OCI handles in thread slot data.
* This is called once per thread.
* Specify that OCI library should not handle mutexing
*/
OCIEnvInit(&tpcenv, OCI_ENV_NO_MUTEX, 0, (dvoid **)0);
OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid **)0);
OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid **)0);
OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid **)0);
OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SRVRCTXT, errhp);
OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)TPC_uid,
(ub4)strlen(TPC_uid),OCI_ATTR_USERNAME, errhp);
OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)TPC_pwd,
(ub4)strlen(TPC_pwd), OCI_ATTR_PASSWORD, errhp);
OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));
OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_USERCTXT, errhp);
/* run all transaction in serializable mode */
OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT);
OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT);
OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
OCIHandleFree(curi, OCI_HTYPE_STMT);
#ifdef SQL_TRACE
/* Turn on the SQL_TRACE */
OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, &xmem);
sprintf ((char *) stmbuf, SQLTXT1);
OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT);
OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
#endif /* End SQL_TRACE */
/* Store the handles just initialized in the thread slot
*/
cn_dataP->tpcenv = tpcenv;
cn_dataP->tpcsrv = tpcsrv;
cn_dataP->errhp = errhp;
cn_dataP->tpcsvc = tpcsvc;
cn_dataP->tpcusr = tpcusr;
cn_dataP->curi = curi;
cn_dataP->xmem = xmem;
}
static void freeOCIhandles(ora_cn_data_t *cn_dataP)
{
OCIServer *tpcsrv;
OCISession *tpcusr;
OCIEnv *tpcenv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
if (tpcusr = cn_dataP->tpcusr) {
err_printf("free_handles> OCIHandleFree tpcusr\n");
OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
}
if (tpcsvc = cn_dataP->tpcsvc) {
err_printf("free_handles> OCIHandleFree tpcsvc\n");
OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
}
if (errhp = cn_dataP->errhp) {
err_printf("free_handles> OCIHandleFree errhp\n");
OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
}
if (tpcsrv = cn_dataP->tpcsrv) {
err_printf("free_handles> OCIHandleFree tpcsrv\n");
OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
}
if (tpcenv = cn_dataP->tpcenv) {
err_printf("free_handles> OCIHandleFree tpcenv\n");
OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);
}
}
TPCexit ()
{
if (lfp) {
fclose (lfp);
lfp = NULL;
}
#ifndef ENCINA
clean_cn((void *)connectionP);
connectionP = NULL;
#endif
}
/* clean_cn
*
* Called to clean a connection.
* When using pthread this is registered during pthread_create
* and called automatically by pthread when the thread exits.
*/
static void clean_cn(void *ptr)
{
/* free trans specific cursor handles first and later the ora handles */
ora_cn_data_t *cn_dataP = (ora_cn_data_t *)ptr;
if (cn_dataP != NULL) {
err_printf("clean_cn, Freeing OCI handles\n");
plnewdone(cn_dataP);
plpaydone(cn_dataP);
plorddone(cn_dataP);
pldeldone(cn_dataP);
plstodone(cn_dataP);
freeOCIhandles(cn_dataP);
err_printf("free_handles> free cn_dataP\n");
}
}
static char *thread_state_to_str(int state)
{
char *retval;
switch(state) {
case SVR_STATE_NONE: retval = "None"; break;
case SVR_STATE_SENT: retval = "Sent"; break;
case SVR_STATE_REPLIED: retval = "Replied"; break;
case SVR_STATE_ERR: retval = "Err"; break;
default: retval = "unknown"; break;
}
return retval;
}
void print_rt_avg(total_tran_count_t *curP, total_tran_count_t *prevP)
{
int i;
double avg_queued_time;
static char *names[] = {"id", "no", "pa", "os", "dl", "sl", "dn", "dp"};
err_printf("bg_thread RT avg: ");
for (i=0; i<=MAX_TRAN_TYPE; i++) {
int num_trans = curP->tran[i].num - prevP->tran[i].num;
double rt_diff = curP->tran[i].RT - prevP->tran[i].RT;
PRINT_AV(rt_diff, num_trans, names[i]);
if (i == DELIVERY_TRANS && num_trans > 0) {
rt_diff = curP->dvry_queue_time - prevP->dvry_queue_time;
PRINT_AV(rt_diff, num_trans, "qd");
}
}
fprintf(stderr, "\n");
}
void print_tran_info(tran_spec_info_t *tran_copy)
{
int i;
struct payinstruct *paP;
struct newinstruct *noP;
switch (tran_copy->tran_type) {
case DIST_NEWO_TRANS:
case NEWO_TRANS:
noP = &(tran_copy->tran_info.no.newin);
err_printf("bg_thread: TPCnew : w_id %d, d_id %d, c_id %d\n",
noP->w_id,
noP->d_id,
noP->c_id);
for (i=0; i<15; i++) {
if (noP->ol_i_id[i] == 0) {
break;
}
err_printf("bg_thread TPCnew : (%d) i_id %d sup_w_id %d Qty %d\n",
i,
noP->ol_i_id[i],
noP->ol_supply_w_id[i],
noP->ol_quantity[i]);
}
break;
case DIST_PAY_TRANS:
case PAYMENT_TRANS:
paP = &(tran_copy->tran_info.pa.payin);
err_printf("bg_thread: TPCpay: w_id %d, D_id %d, C_w_id %d, c_id %d,
bylastname %d, amount %.2f, c_last %s \n",
paP->w_id,
paP->d_id,
paP->c_w_id,
paP->c_id,
paP->bylastname,
paP->h_amount,
paP->c_last);
break;
case ORDER_STAT_TRANS:
err_printf("bg_thread: TPCord: w_id %d, d_id %d, c_id %d, bylastname %d, c_last
%s \n",
tran_copy->tran_info.os.ordin.w_id,
tran_copy->tran_info.os.ordin.d_id,
tran_copy->tran_info.os.ordin.c_id,
tran_copy->tran_info.os.ordin.bylastname,
tran_copy->tran_info.os.ordin.c_last);
```

```
break;
case DELIVERY_TRANS:
err_printf("bg_thread: TPCdel: w_id %d, o_carrier_id %d, %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;
}
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.
*/
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);
}
```

tpccpl.h

```
/*
* $Header:
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/sp-tpcc/ora8MT_encMT/RC
S/tpccpl.h,v 1.2 1998/01/23 15:08:22 oz Exp $ Copyr (c) 1994 Oracle
*/
/*+=================================================================+
| Copyright (c) 1994 Oracle Corp, Redwood Shores, CA |
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=================================================================+
| FILENAME
| tpccpl.h
| DESCRIPTION
| Header file for TPC-C transactions in PL/SQL.
+=================================================================*/
#ifndef TPCCPL_H
#define TPCCPL_H
#include <stdio.h>
#include "tpcc.h"
#define DELRT 80.0
extern errrpt ();
extern int ocierror(char *fname, int lineno,OCIError *errhp, sword status);
extern int sqlfile(char *fname, text *linebuf);
extern FILE *lfp;
extern FILE *fopen ();
extern int proc_no;
#ifndef DISCARD
# define DISCARD (void)
#endif
#ifndef sword
# define sword int
#endif
#define VER7 2
#define NA -1 /* ANSI SQL NULL */
#define NLT 1 /* length for string null terminator */
#define DEADLOCK 60 /* ORA-00060: deadlock */
#define NO_DATA_FOUND 1403 /* ORA-01403: no data found */
#define NOT_SERIALIZABLE 8177 /* ORA-08177: transaction not serializable */
#define SNAPSHOT_TOO_OLD 1555 /* ORA-01555: snapshot too old */
#ifndef NULLP
# define NULLP (void *)NULL
#endif /* NULLP */
#define ADR(object) ((ub1 *)&(object))
#define SIZ(object) ((sword)sizeof(object))
typedef char date[24+NLT];
typedef char varchar2;
#define OCIERROR(errp,function)\
ocierror(__FILE__,__LINE__,(errp),(function));
#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
ocierror(__FILE__,__LINE__,(errp), \
OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
ocierror(__FILE__,__LINE__, (errp), \
OCIBindByName((stmp), &(bndp), (errp), \
(text *)(sqlvar), strlen((sqlvar)),\
(progv), (progvl), (ftype),0,0,0,0,0,OCI_DEFAULT));
#define OCIBNDRA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
ocierror(__FILE__,__LINE__,(errp), \
OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
ocierror(__FILE__,__LINE__,(errp), \
OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
(progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));
#define OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,cbf_nodata,cbf_data) \
ocierror(__FILE__,__LINE__,(errp), \
OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
ocierror(__FILE__,__LINE__,(errp), \
OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar), \
strlen((sqlvar)),0,(progvl),(ftype), \
indp,0,0,0,0,OCI_DATA_AT_EXEC)); \
ocierror(__FILE__,__LINE__,(errp), \
OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)));
#define OCIBNDR(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
ocierror(__FILE__,__LINE__,(errp), \
OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
ocierror(__FILE__,__LINE__,(errp), \
OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
(progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));
#define OCIBNDRAA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode,ms,cs) \
```

```
ocierror(__FILE__,__LINE__, (errp), \
OCIHandleAlloc((stmp),(dvoid **)(&(bndp)),OCI_HTYPE_BIND,0,(dvoid**)0)); \
ocierror(__FILE__,__LINE__,(errp),\
OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
(progv),(progvl),(ftype),(indp),(alen),(arcode),(ms),(ub4 *)(cs),OCI_DEFAULT));
#define OCIDEFINE(stmp,dfnp,errp,pos,progv,progvl,ftype)\
OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),(ftype),\
0,0,0,OCI_DEFAULT);
#define OCIDEF(stmp,dfnp,errp,pos,progv,progvl,ftype) \
OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
(dvoid**)0);\
OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),\
(ftype),NULL,NULL,NULL,OCI_DEFAULT); \
#define OCIDFNRA(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,alen,arcode) \
OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
(dvoid**)0);\
OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
(progvl),(ftype),(indp),(alen),\
(arcode),OCI_DEFAULT);\
/*
OCIDefineArrayOfStruct((dfnp),(errp),(progvl),\
sizeof((indp)[0]),\
sizeof((alen)[0]),\
sizeof((arcode)[0]));
*/
/*
#define OCIDFNRA(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,alen,arcode) \
ocierror(__FILE__,__LINE__,(errp), \
OCIHandleAlloc(tpcenv,&(dfnp),OCI_HTYPE_DEFINE,0,\
(dvoid**)0));\
ocierror(__FILE__,__LINE__,(errp), \
OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
(progvl),(ftype),(indp),(alen),\
(arcode),OCI_DEFAULT));\
ocierror(__FILE__,__LINE__,(errp),\
OCIDefineArrayOfStruct((dfnp),(errp),(progvl),\
sizeof((indp)[0]),\
sizeof((alen)[0]),\
sizeof((arcode)[0])));
*/
#define OBNDRV(lda,cursor,sqlvar,progv,progvl,ftype)\
if (obndrv((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(progvl),(ftype),NA,\
(sb2 *)0, (text *)0, NA, NA))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0
#define OBNDRA(lda,cursor,sqlvar,progv,progvl,ftype,indp,alen,arcode)\
if (obndra((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(progvl),(ftype),NA,\
(indp),(alen),(arcode),(ub4)0,(ub4*)0,(text*)0,NA,NA))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0
#define OBNDRAA(lda,cursor,sqlvar,progv,progvl,ftype,indp,alen,arcode,ms,cs)\
if (obndra((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(progvl),(ftype),NA,\
(indp),(alen),(arcode),(ub4)(ms),(ub4*)(cs),(text*)0,NA,NA))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0
#define ODEFIN(lda,cursor,pos,buf,bufl,ftype,scale,indp,fmt,fmtl,fmtt,rlen,rcode)\
if (odefin((cursor),(pos),(ub1*)(buf),(bufl),(ftype),(scale),(indp),\
(text*)(fmt),(fmtl),(fmtt),(rlen),(rcode)))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0
#define OEXFET(lda,cursor,nrows,cancel,exact)\
if (oexfet((cursor),(nrows),(cancel),(exact)))\
{if ((cursor)->rc == 1403) DISCARD 0; \
else if (errrpt(lda,cursor)==RECOVERR) \
{orol(lda);return(RECOVERR);} \
else{orol(lda);return(-1);}}\
else\
DISCARD 0
#define OOPEN(lda,cursor)\
if (oopen((cursor),(lda),(text*)0,NA,NA,(text*)0,NA))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0
#define OPARSE(lda,cursor,sqlstm,sqll,defflg,lngflg)\
if (oparse((cursor),(sqlstm),(sb4)(sqll),(defflg),(ub4)(lngflg)))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0
#define OFEN(lda,cursor,nrows)\
if (ofen((cursor),(nrows)))\
{if (errrpt(lda,cursor)==RECOVERR) \
{orol(lda);return(RECOVERR);} \
else{orol(lda);return(-1);}}\
else\
DISCARD 0
#define OEXEC(lda,cursor)\
if (oexec((cursor)))\
{if (errrpt(lda,cursor)==RECOVERR) \
{orol(lda);return(RECOVERR);} \
else{orol(lda);return(-1);}}\
else\
DISCARD 0
#define OCOM(lda,cursor)\
if (ocom((lda))) \
{errrpt(lda,cursor);orol(lda);return(-1);}\
else\
DISCARD 0
#define OEXN(lda,cursor,iters,rowoff)\
if (oexn((cursor),(iters),(rowoff))) \
{if (errrpt(lda,cursor)==RECOVERR) \
{orol(lda);return(RECOVERR);} \
else{orol(lda);return(-1);}}\
else\
DISCARD 0
#endif
/*---------------------------------------------------------------+
```

new.sql

```
DECLARE /* new order */
not_serializable EXCEPTION;
PRAGMA EXCEPTION_INIT(not_serializable,-8177);
deadlock EXCEPTION;
PRAGMA EXCEPTION_INIT(deadlock,-60);
snapshot_too_old EXCEPTION;
PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
BEGIN
LOOP BEGIN
SELECT c_discount, c_last, c_credit
INTO :c_discount, :c_last, :c_credit
FROM customer
WHERE c_id = :c_id
AND c_d_id = :d_id
AND c_w_id = :w_id;
UPDATE wh_dist SET d_next_o_id = d_next_o_id + 1, d_tax=d_tax+0
WHERE d_id = :d_id
AND w_id = :w_id
RETURNING d_tax, d_next_o_id-1, w_tax
INTO :d_tax, :o_id, :w_tax;
INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
VALUES (:o_id , :d_id, :w_id);
INSERT INTO orders (o_id, o_w_id, o_d_id, o_c_id, o_carrier_id,
o_ol_cnt, o_all_local,o_entry_d)
VALUES (:o_id, :w_id, :d_id, :c_id, 11,
:o_ol_cnt, :o_all_local, :cr_date);
EXIT;
EXCEPTION
WHEN not_serializable OR deadlock OR snapshot_too_old THEN
ROLLBACK;
:retry := :retry + 1;
END;
END LOOP;
END;
```

pay.sql

```
CREATE OR REPLACE PACKAGE pay
AS
TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
row_id rowidarray;
cust_rowid ROWID;
dist_name VARCHAR2(11);
ware_name VARCHAR2(11);
c_num BINARY_INTEGER;
PROCEDURE pay_init;
END pay;
/
CREATE OR REPLACE PACKAGE BODY pay AS
PROCEDURE pay_init IS
BEGIN
NULL;
END pay_init;
END pay;
/
exit;
```

paynz.sql

```
DECLARE /* paynz */
-- cust_rowid ROWID;
-- dist_name VARCHAR2(11);
-- ware_name VARCHAR2(11);
not_serializable EXCEPTION;
PRAGMA EXCEPTION_INIT(not_serializable,-8177);
deadlock EXCEPTION;
PRAGMA EXCEPTION_INIT(deadlock,-60);
snapshot_too_old EXCEPTION;
PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
BEGIN
LOOP BEGIN
UPDATE customer
SET c_balance = c_balance - :h_amount,
c_ytd_payment = c_ytd_payment + :h_amount,
c_payment_cnt = c_payment_cnt+1
WHERE c_id = :c_id AND c_d_id = :c_d_id AND
c_w_id = :c_w_id
RETURNING rowid, c_first, c_middle, c_last, c_street_1,
c_street_2, c_city, c_state, c_zip, c_phone,
c_since, c_credit, c_credit_lim,
c_discount, c_balance
INTO pay.cust_rowid,:c_first, :c_middle, :c_last,
:c_street_1,
:c_street_2, :c_city, :c_state, :c_zip, :c_phone,
:c_since, :c_credit, :c_credit_lim,
:c_discount, :c_balance;
IF SQL%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 = pay.cust_rowid
RETURNING substr(c_data,1, 200)
INTO :c_data;
END IF;
UPDATE district
SET d_ytd = d_ytd + :h_amount
WHERE d_id = :d_id
```

```
AND d_w_id = :w_id
RETURNING d_name, d_street_1, d_street_2, d_city,d_state, d_zip
INTO pay.dist_name,:d_street_1,:d_street_2,:d_city,:d_state,
:d_zip;
IF SQL%NOTFOUND THEN
raise NO_DATA_FOUND;
END IF;
UPDATE warehouse
SET w_ytd = w_ytd + :h_amount
WHERE w_id = :w_id
RETURNING w_name, w_street_1, w_street_2, w_city, w_state, w_zip
INTO pay.ware_name, :w_street_1, :w_street_2, :w_city,
:w_state,
:w_zip;
INSERT INTO history (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
h_amount, h_date, h_data)
VALUES
(:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
:cr_date, pay.ware_name || ' ' || pay.dist_name);
COMMIT;
:h_date := to_char (:cr_date, 'DD-MM-YYYY.HH24:MI:SS');
EXIT;
EXCEPTION
WHEN not_serializable OR deadlock OR snapshot_too_old THEN
ROLLBACK;
:retry := :retry + 1;
END;
END LOOP;
END;
```

payz.sql

```
DECLARE /* payz */
-- TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
-- cust_rowid ROWID;
-- dist_name VARCHAR2(11);
-- ware_name VARCHAR2(11);
-- c_num BINARY_INTEGER;
-- row_id rowidarray;
not_serializable EXCEPTION;
PRAGMA EXCEPTION_INIT(not_serializable,-8177);
deadlock EXCEPTION;
PRAGMA EXCEPTION_INIT(deadlock,-60);
snapshot_too_old EXCEPTION;
PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
CURSOR c_cur IS
SELECT rowid
FROM customer
WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last = :c_last
ORDER BY c_w_id, c_d_id, c_last, c_first;
BEGIN
LOOP BEGIN
pay.c_num := 0;
FOR c_id_rec IN c_cur LOOP
pay.c_num := pay.c_num + 1;
pay.row_id(pay.c_num) := c_id_rec.rowid;
END LOOP;
pay.cust_rowid := pay.row_id ((pay.c_num + 1) / 2);
UPDATE customer
SET c_balance = c_balance - :h_amount,
c_ytd_payment = c_ytd_payment+ :h_amount,
c_payment_cnt = c_payment_cnt+1
WHERE rowid = pay.cust_rowid
RETURNING
c_id, c_first, c_middle, c_last, c_street_1, c_street_2,
c_city, c_state, c_zip, c_phone,
c_since, c_credit, c_credit_lim,
c_discount, c_balance
INTO :c_id, :c_first, :c_middle, :c_last,
:c_street_1, :c_street_2, :c_city, :c_state,
:c_zip, :c_phone, :c_since, :c_credit,
:c_credit_lim, :c_discount, :c_balance;
:c_data := ' ';
IF :c_credit = 'BC' THEN
UPDATE customer
SET c_data = substr ((to_char (:c_id) || ' ' ||
to_char (:c_d_id) || ' ' ||
to_char (:c_w_id) || ' ' ||
to_char (:d_id) || ' ' ||
to_char (:w_id) || ' ' ||
to_char (:h_amount/100, '9999.99') ||
' |
')
|| c_data, 1, 500)
WHERE rowid = pay.cust_rowid
RETURNING substr(c_data,1, 200)
INTO :c_data;
END IF;
UPDATE district
SET d_ytd = d_ytd+:h_amount
WHERE d_id = :d_id
AND d_w_id = :w_id
RETURNING d_name, d_street_1, d_street_2, d_city,
d_state, d_zip
INTO pay.dist_name, :d_street_1, :d_street_2, :d_city,
:d_state, :d_zip;
IF SQL%NOTFOUND THEN
raise NO_DATA_FOUND;
END IF;
UPDATE warehouse
SET w_ytd = w_ytd+:h_amount
WHERE w_id = :w_id
RETURNING w_name,
w_street_1, w_street_2, w_city, w_state, w_zip
INTO pay.ware_name,
:w_street_1, :w_street_2, :w_city, :w_state, :w_zip;
INSERT INTO history (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
h_amount, h_date, h_data)
VALUES (:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
:cr_date, pay.ware_name || ' ' || pay.dist_name);
COMMIT;
:h_date := to_char (:cr_date, 'DD-MM-YYYY.HH24:MI:SS');
EXIT;
EXCEPTION
WHEN not_serializable OR deadlock OR snapshot_too_old THEN
ROLLBACK;
:retry := :retry + 1;
END;
END LOOP;
END;
```

views.sql

```
create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
from district d, warehouse w
where w.w_id = d.d_w_id
/
create or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
s_order_cnt, s_ytd, s_remote_cnt,
s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
select i.i_id, s_w_id, i.i_price, i.i_name, i.i_data, s_data, s_quantity,
s_order_cnt, s_ytd, s_remote_cnt,
s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
from stock s, item i
where i.i_id = s.s_i_id
/
exit;
```

APPENDIX B: Tunable Parameters

B.1 Database Parameters

```
# $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
db_writer_processes=2
recovery_parallelism = 20
compatible = 8.0.4.0.0
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 = 270
max_dump_file_size = 3000
open_cursors = 200
sessions = 700
transactions = 700
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,t100,t101,t102,t103,t104, \
t105,t106,t107,t108,t109,t110,t111,t112,t113,t114,t115,t116, \
t117,t118,t119,t120,t121,t122,t123,t124,t125,t126,t127,t128, \
t129,t130,t131,t132,t133,t134,t135,t136,t137,t138,t139,t140, \
t141,t142,t143,t144,t145,t146,t147,t148,t149,t150,t151,t152, \
t153,t154,t155,t156,t157,t158,t159,t160,t161,t162,t163,t164,t165,t166,\
t167,t168,t169,t170,t171,t172,t173,t174,t175,t176,t177,t178,t179,t180,\
t181,t182,t183,t184,t185,t186,t187,t188,t189,t190,t191,t192,t193,t194, \
t195,t196,t197,t198,t199,t200,t201,t202,t203,t204,t205,t206,t207,t208, \
t209,t210,t211,t212,t213,t214,t215,t216,t217,t218,t219,t220,t221,t222, \
t223,t224,t225,t226,t227,t228,t229,t230,t231,t232,t233,t234,t235,t236, \
t237,t238,t239,t240,t241,t242,t243,t244,t245,t246,t247,t248,t249,t250, \
t251,t252,t253,t254,t255,t256,t257,t258,t259,t260,t261,t262,t263,t264)
discrete_transactions_enabled = FALSE
cursor_space_for_time = TRUE
replication_dependency_tracking=FALSE
log_small_entry_max_size = 80
processes = 280
shared_pool_size = 30000000
db_block_lru_latches = 24
buffer_pool_recycle = (buffers:3000,lru_latches:4)
spin_count = 6000
log_buffer = 3145728
_db_block_write_batch = 4096
db_block_checkpoint_batch = 2048
db_block_buffers = 3438560
_db_block_hash_buckets = 1146186
```

B.2 Transaction Monitor Parameters

```
#
# ID: $Id: tpccCommon.tcl,v 1.14 1997/08/01 13:08:58 oz Exp $
#
# COMPONENT_NAME: Encina Administration Test
#
# ORIGINS: Transarc Corp.
#
# (C) COPYRIGHT Transarc Corp. 1995
# All Rights Reserved
# Licensed Materials - Property of Transarc
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with Transarc Corp
#
# HISTORY
# $TALog: tpccCommon.tcl,v $
#
#-----------------------------------------------------------------------
# This tcl script provides procedures for creating and starting cells,
# node managers, sfs, rqs and mas servers. It is a general purpose
# script which uses enccp commands to accomplish the abvove.
#
# This script determines the cell name by checking whether the environment
# variable ENCINA_TPM_CELL is set and if so uses that as the CELL name
# otherwise, it builds a cell name using the user name and machine name
#
#-----------------------------------------------------------------------
# getenv - import an environment variable. Error if it doesn't exist.
#-----------------------------------------------------------------------
proc getenv {variable} {
global $variable
global env
set $variable $env($variable)
}
## Default values used for the different scripts
set DEFAULT_TRACE "all=none"
set TPCC_BIN_DIR "/home/encina/bin"
getenv USER
set DCE_PWD e26spw
## The number PAs each server is configured to have
set MT_PAS 6
set MT_THREADS 5
set MT_DVRY_PAS 1
set DVRY_THREADS 3
set DB_NAME oratpcc
set NO_PAS 45
set PAY_PAS 15
set SL_PAS 5
set OS_PAS 3
set DVRY_PAS 5
set XA_PAS 0
set REMOTE_PAS 0
set DEFAULT_PAS 1
set QUEUE_PARAM "ENCINA_THREAD_POOL_QUEUE_LENGTH=1800
ENCINA_DEFAULT_THREAD_POOL_QUEUE_LENGTH=1800"
set UNAME [exec uname]
set ENCINA_BIN_PATH "/usr/lpp/encina/bin"
if { $UNAME == "AIX" } {
set HOSTNAME [exec hostname -s]
} else {
set HOSTNAME [exec hostname]
}
#set ADDITIONAL_ENV "RPC_SUPPORTED_PROTSEQS=ncadg_ip_udp
TRPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp"
#set ADDITIONAL_ENV "RPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp
TRPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp"
set ADDITIONAL_ENV ""
set ADDITIONAL_MAS_ENV "RPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp
TRPC_SUPPORTED_PROTSEQS=ncacn_ip_tcp"
#set ADDITIONAL_MAS_ENV "RPC_SUPPORTED_PROTSEQS=ncadg_ip_udp
TRPC_SUPPORTED_PROTSEQS=ncadg_ip_udp"
# set ADDITIONAL_MAS_ENV ""
set UNSUPPORTED_NETIFS $env(RPC_UNSUPPORTED_NETIFS)
if {"${HOSTNAME}" == "perf" || "${HOSTNAME}" == "hydra1" ||
"${HOSTNAME}" == "titan" } {
set TPCC_BIN_DIR "/afs/tr/usr/oz/kansas/test/build/benchmarks/tpcc/sp-tpcc"
set UNSUPPORTED_NETIFS ""
set ENCINA_BIN_PATH "/afs/tr/kansas/latest/internal/bin"
set DEFAULT_PAS 2
set NO_PAS 2
set PAY_PAS 2
set SL_PAS 2
set OS_PAS 2
set DVRY_PAS 2
set XA_PAS 2
set REMOTE_PAS 4
}
if {"${HOSTNAME}" == "frog"} {
set TPCC_BIN_DIR "/afs/tr/usr/oz/kansas/test/build/benchmarks/tpcc/sp-tpcc"
set UNSUPPORTED_NETIFS ""
set ENCINA_BIN_PATH "/afs/tr/kansas/latest/internal/bin"
set DEFAULT_PAS 2
set NO_PAS 2
set PAY_PAS 2
set SL_PAS 2
set OS_PAS 2
set DVRY_PAS 2
set XA_PAS 2
set REMOTE_PAS 4
}
proc showHelp {} {
global NO_PAS SL_PAS OS_PAS DVRY_PAS PAY_PAS XA_PAS REMOTE_PAS
puts "There are a number of scripts to simplify the TPCC"
puts "configuration. The main ones are:"
puts " showMas -- shows the current status of all MAS"
puts " stopAllMas - Stops all the MASs"
puts " startAllMas - Start all the MASs"
puts " deleteAllMas - Delete all the MASs"
puts " showHelp -- Show this help message"
puts " "
puts " doMas <op> <node> - perform the op on all the MAS of that node"
puts " e.g. doMas start k66n28"
puts " "
puts " Configuration"
puts " createCell"
puts " createNode"
puts " createMas <node> <db> - Create and start all the MASs needed"
puts " for the specified DB on the specified node"
puts " createMasDbg <node> <db> - Create and start all the "
puts " debug MASs needed"
puts " for the specified DB on the specified node"
puts " createTpcc <numNodes> - Create and start all the MASs needed"
puts " for the specified number of nodes. The nodes are:"
puts " 28->DB13, 27->DB05, 29->DB09, and 30->DB01"
puts " if <n> nodes are specified the first n nodes will be created"
puts " configMas <node> <DB> - Set the paCount on all the MASs."
puts " configMasDbg <node> <DB> - Set the paCount on all the "
puts " debug MASs."
puts " Defaults: NO $NO_PAS, PAY $PAY_PAS, OS $OS_PAS, SL $SL_PAS, DVRY
$DVRY_PAS, XA $XA_PAS, REMOTE $REMOTE_PAS"
puts " "
puts " createMas calls the following three functions:"
puts " dari_mas_all <node> <name> <db_spec> - Create all the local"
```

```
puts " servers for the specified node and db"
puts " e.g. dari_mas_all k66n28 05A db:tpcc05"
puts " dari_xa <node> <name> <db_spec> <if> - create an XA server"
puts " on the specified node. <if> should be -1"
puts " e.g. dari_xa k66n28 05A db:tpcc05 -1"
puts " dari_remote <node> <name> <db_spec> - create a REMOTE server"
puts " on the specified node. "
puts " e.g. dari_remote k66n28 05A db:tpcc05"
puts " "
puts " Conventions: "
puts " - The names of the servers start with the name of the node"
puts " - The names contain the DB it is connect to"
puts " - Remote servers have -remote- in their names and XA ones"
puts " have -xa- in their name."
puts " - Local servers end with the name of the transaction they process."
puts " - The name of the nodes is comprised of the short name of the"
puts " machine."
puts " Example names:"
puts " k66n28-xa-05A"
puts " k66n28-05A-dvry"
puts " "
}
proc listMas {} {
foreach mas [mas list] {
puts "$mas"
}
}
proc countThreads {mas} {
set vars [mas show $mas -environment]
foreach var $vars {
if [regexp {([^=]*)=(.*)$} $var A B C] {
echo "Var $B is $C"
}
}
}
proc showMas {} {
foreach mas [mas list] {
mas show $mas -setArray masInfo
set threads 0
foreach el $masInfo(environment) {
if [regexp {ENCINA_(APPL_)TPOOL_SIZE=([0-9]+)} $el a name num] {
set threads [expr $threads + $num]
}
}
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} ENCINA_TRACE=tpm=all ENCINA_TRACE_VERBOSE=1" \
-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 $protLevel
}
#--------------------------------------------------------------------
# createInterfaces :
# procedure for creating required interfaces
#--------------------------------------------------------------------
proc createInterfaces {} {
catch {interface create delivery}
catch {interface create neworder}
catch {interface create orderstatus}
catch {interface create stocklevel}
catch {interface create payment}
catch {interface create tpccTrans}
}
#--------------------------------------------------------------------
# create_and_start_mas :
# procedure for creating and starting a mas server which
# exports the interfaces tpmServer and tpmServer_admin
# Takes the name of the mas server to be created
#--------------------------------------------------------------------
proc mt_mas {node name db ifs threads} {
global TPCC_BIN_DIR env DVRY_THREADS
if {$db == ""} {
set cmdline "-no_db $ifs MON_CONCURRENT_SHARED 1 dvry=$DVRY_THREADS
db:null"
} else {
set cmdline "$ifs MON_CONCURRENT_SHARED 1 dvry=$DVRY_THREADS db:$db"
}
set db_env "TWO_TASK=$db ORACLE_SID=$db
ORACLE_HOME=$env(ORACLE_HOME)"
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=$a_tpool"
set local_env "${db_env} ${mas_env}"
set name [create_and_start_mas $node $name $cmdline $local_env]
puts "Created and started $name"
setTimeouts $name
return $name
}
proc create_and_start_mas {node name cmdline masenv} {
global SHORT_CELL_NAME env
global ENCINA_TPM_CELL QUEUE_PARAM
global HOSTNAME ADDITIONAL_MAS_ENV
global TPCC_BIN_DIR DEFAULT_PAS
global DEFAULT_TRACE UNSUPPORTED_NETIFS
set mas_name ${node}-${name}
puts "Creating MAS ${mas_name} on node $node"
puts " cmdline: '$cmdline'"
puts " Binary: ${TPCC_BIN_DIR}/serverMT_stats"
mas create ${mas_name} -node ${node} \
-executable ${TPCC_BIN_DIR}/serverMT_stats\
-commandLineArgs "$cmdline" \
-environment "TZ=CST6CDT ENCINA_TRACE=${DEFAULT_TRACE}
RPC_UNSUPPORTED_NETIFS=$UNSUPPORTED_NETIFS
ENCINA_TRPC_DS_PATH=${ENCINA_TPM_CELL}/trpc/ $QUEUE_PARAM ${masenv}
${ADDITIONAL_MAS_ENV}" \
-interfaces {delivery tpccTrans} \
-paCount $DEFAULT_PAS -groupName staff
# need this to do ema_attr stuff
set server_principal $SHORT_CELL_NAME/server/$mas_name
# puts "Adding principal $server_principal to encina_admin_group"
# group add encina_admin_group -member $server_principal
mas show $mas_name -setArray masInfo
puts "Starting MAS ${mas_name}"
mas start ${mas_name} -myPwd encina_admin
puts "Setting ACLs on all the interfaces"
catch {exec acl_edit ${ENCINA_TPM_CELL}/ecm/interface/tpccTrans -m
user:encina_admin:x}
catch {exec acl_edit ${ENCINA_TPM_CELL}/ecm/interface/delivery -m
user:encina_admin:x}
return ${mas_name}
}
proc doMas {op node} {
foreach mas [mas list] {
if [ string match ${node}* $mas ] {
echo "mas $op $mas"
catch "mas $op $mas"
} else {
# puts "Ignoring $mas"
}
}
return
}
proc setTimeouts {mas} {
mas modify $mas -authorizationEnabled 0
puts "Setting timeouts for longRes on $mas"
mas modify $mas -longTermResFirstUseTimeout 35:00:00
mas modify $mas -longTermResPingTimeout 35:00:00
mas modify $mas -longTermResPingInterval 30:00:00
puts "Setting timeouts for ping on $mas"
mas modify $mas -pingTimeout 25:00:00
mas modify $mas -pingInitialTimeout 25:00:00
mas modify $mas -pingInterval 20:00:00
puts "Timeouts set"
}
proc createMas {node DB} {
global XA_PAS
createInterfaces
dari_mas_all $node ${DB}-A $DB
if {$XA_PAS > 0} {
dari_xa $node ${DB}-A tpcc$DB 0xffff
dari_remote $node ${DB}-A tpcc$DB
}
configMas $node $DB
}
proc createMtMas {node {dataB "-check-"} } {
global MT_PAS MT_DVRY_PAS DB_NAME
global MT_THREADS DVRY_THREADS
createInterfaces
if { $dataB == "-check-" } { set dataB $DB_NAME }
if { $dataB == "NULL" } { set dataB "" }
set all_ifs 23
if {$MT_DVRY_PAS > 0} {
set name [mt_mas $node ${dataB}-dvr-MT $dataB 8 $MT_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 40000 Maximum number of PROCESSES allowed per user True
autorestart false Automatically REBOOT system after a crash True
iostat true Continuously maintain DISK I/O history True
realmem 16777216 Amount of usable physical memory in Kbytes False
conslogin enable System Console Login False
fwversion IBM,19971120 (B) Firmware version and revision levels False
maxpout 0 HIGH water mark for pending write I/Os per file True
minpout 0 LOW water mark for pending write I/Os per file True
fullcore false Enable full CORE dump True
pre430core false IBM PowerPC CHRP Computer True
rtasversion 1 Open Firmware RTAS version False
modelname IBM,7015-S70 Machine name False
systemid IBM,011090014 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
alter tablespace $1 add datafile ’$2’ size $3 reuse;
exit;
!
if [ -d ./outdir ]
then
echo ‘date‘ >> ./outdir/${FILE}.addf
fi
```

benchdb.sh

```
#!/bin/ksh
#
# $Header: benchdb.sh 7030100.1 96/05/02 19:05:22 plai Generic<base> $ Copyr (c) 1995
Oracle
#
#
#=============================================================+
# Copyright (c) 1996 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
# MODIFIED
# L.R 11/6/97 Modified for 1700 warehouses
#=============================================================
#
BENCH_HOME=$ORACLE_HOME/bench/tpc
TPCC_ADMIN=admin
while [ "$#" != "0" ]
do
case $1 in
-n) shift
NO_CREATE="y"
;;
-c) shift
NO_CAT="y"
;;
*) echo "Bag 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 430M reuse
logfile ’/dev/rlvlog1’ size 1599M reuse,
’/dev/rlvlog2’ size 1599M reuse;
exit
!
#
# Create more rollback segments
#
svrmgrl <<!
connect system/manager
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);
disconnect;
connect internal;
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 roll datafile
’/dev/rlvroll1’ size 103M reuse;
create tablespace hist datafile
’/dev/rlvhist0’ size 100K reuse;
create tablespace ware datafile
’/dev/rlvware’ size 70M reuse;
create tablespace cust datafile
’/dev/rlvcust0’ size 100K reuse;
create tablespace items datafile
’/dev/rlvitem’ size 30M reuse;
create tablespace ord datafile
’/dev/rlvord0’ size 100k reuse;
create tablespace nord datafile
’/dev/rlvnord0’ size 100k reuse;
create tablespace ordl datafile
’/dev/rlvordl0’ size 100k reuse;
create tablespace stocks datafile
’/dev/rlvstock0’ size 100K reuse;
# index tablespaces
create tablespace icust1 datafile
’/dev/rlvicust10’ size 100K reuse;
create tablespace icust2 datafile
’/dev/rlvicust20’ size 100K reuse;
create tablespace istk datafile
’/dev/rlvistk0’ size 100K reuse;
create tablespace iord1 datafile
’/dev/rlviord10’ size 100k reuse;
create tablespace iord2 datafile
’/dev/rlviord20’ size 100k reuse;
create tablespace inord datafile
’/dev/rlvinord0’ size 100k reuse;
create tablespace iordl datafile
’/dev/rlviordl0’ size 100k reuse;
create tablespace temp datafile
’/dev/rlvtemp0’ size 1999M reuse;
exit;
!
#
# Add datafiles to tablespaces in parallel
#
addfile.sh hist /dev/rlvhist1 1007M &
addfile.sh hist /dev/rlvhist2 1007M &
addfile.sh hist /dev/rlvhist3 1007M &
addfile.sh hist /dev/rlvhist4 1007M &
#
addfile.sh nord /dev/rlvnord1 415M &
#
addfile.sh ord /dev/rlvord1 1007M &
addfile.sh ord /dev/rlvord2 1007M &
addfile.sh ord /dev/rlvord3 1007M &
addfile.sh ord /dev/rlvord4 1007M &
#
addfile.sh ordl /dev/rlvordl1 1887M &
addfile.sh ordl /dev/rlvordl2 1887M &
addfile.sh ordl /dev/rlvordl3 1887M &
addfile.sh ordl /dev/rlvordl4 1887M &
addfile.sh ordl /dev/rlvordl5 1887M &
addfile.sh ordl /dev/rlvordl6 1887M &
addfile.sh ordl /dev/rlvordl7 1887M &
addfile.sh ordl /dev/rlvordl8 1887M &
addfile.sh ordl /dev/rlvordl9 1887M &
addfile.sh ordl /dev/rlvordl10 1887M &
wait
addfile.sh ordl /dev/rlvordl11 1887M &
addfile.sh ordl /dev/rlvordl12 1887M &
addfile.sh ordl /dev/rlvordl13 1887M &
addfile.sh ordl /dev/rlvordl14 1887M &
addfile.sh ordl /dev/rlvordl15 1887M &
addfile.sh ordl /dev/rlvordl16 1887M &
addfile.sh ordl /dev/rlvordl17 1887M &
addfile.sh ordl /dev/rlvordl18 1887M &
addfile.sh ordl /dev/rlvordl19 1887M &
addfile.sh ordl /dev/rlvordl20 1887M &
addfile.sh ordl /dev/rlvordl21 1887M &
addfile.sh ordl /dev/rlvordl22 1887M &
```

```
addfile.sh ordl /dev/rlvordl23 1887M &
addfile.sh ordl /dev/rlvordl24 1887M &
wait
addfile.sh ordl /dev/rlvordl25 1887M &
addfile.sh ordl /dev/rlvordl26 1887M &
addfile.sh ordl /dev/rlvordl27 1887M &
addfile.sh ordl /dev/rlvordl28 1887M &
addfile.sh ordl /dev/rlvordl29 1887M &
addfile.sh ordl /dev/rlvordl30 1887M &
addfile.sh ordl /dev/rlvordl31 1887M &
addfile.sh ordl /dev/rlvordl32 1887M &
wait
#
addfile.sh icust1 /dev/rlvicust1 1303M &
#
addfile.sh istk /dev/rlvistk1 1311M &
addfile.sh istk /dev/rlvistk2 1311M &
addfile.sh istk /dev/rlvistk3 1311M &
#
addfile.sh iord1 /dev/rlviord11 671M &
addfile.sh iord1 /dev/rlviord12 671M &
addfile.sh iord1 /dev/rlviord13 671M &
wait
addfile.sh iord2 /dev/rlviord21 590M &
addfile.sh iord2 /dev/rlviord22 590M &
addfile.sh iord2 /dev/rlviord23 590M &
addfile.sh iord2 /dev/rlviord24 590M &
addfile.sh iord2 /dev/rlviord25 590M &
addfile.sh iord2 /dev/rlviord26 590M &
wait
#index new-order
addfile.sh inord /dev/rlvinord1 239M &
addfile.sh inord /dev/rlvinord2 239M &
addfile.sh inord /dev/rlvinord3 239M &
#index iorder-line
addfile.sh iordl /dev/rlviordl1 1263M &
addfile.sh iordl /dev/rlviordl2 1263M &
addfile.sh iordl /dev/rlviordl3 1263M &
addfile.sh iordl /dev/rlviordl4 1263M &
addfile.sh iordl /dev/rlviordl5 1263M &
addfile.sh iordl /dev/rlviordl6 1263M &
addfile.sh iordl /dev/rlviordl7 1263M &
addfile.sh iordl /dev/rlviordl8 1263M &
addfile.sh iordl /dev/rlviordl9 1263M &
addfile.sh iordl /dev/rlviordl10 1263M &
addfile.sh iordl /dev/rlviordl11 1263M &
addfile.sh iordl /dev/rlviordl12 1263M &
addfile.sh iordl /dev/rlviordl13 1263M &
addfile.sh iordl /dev/rlviordl14 1263M &
addfile.sh iordl /dev/rlviordl15 1263M &
addfile.sh iordl /dev/rlviordl16 1263M &
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
#
addfile.sh icust2 /dev/rlvicust21 751M &
addfile.sh icust2 /dev/rlvicust22 751M &
addfile.sh icust2 /dev/rlvicust23 751M &
addfile.sh icust2 /dev/rlvicust24 751M &
#
#
addfile.sh cust /dev/rlvcust1 783M &
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 &
wait
addfile.sh cust /dev/rlvcust17 783M &
addfile.sh cust /dev/rlvcust18 783M &
addfile.sh cust /dev/rlvcust19 783M &
addfile.sh cust /dev/rlvcust20 783M &
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 &
addfile.sh cust /dev/rlvcust41 783M &
addfile.sh cust /dev/rlvcust42 783M &
addfile.sh cust /dev/rlvcust43 783M &
addfile.sh cust /dev/rlvcust44 783M &
wait
#
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 &
addfile.sh cust /dev/rlvcust59 783M &
addfile.sh cust /dev/rlvcust60 783M &
addfile.sh cust /dev/rlvcust61 783M &
addfile.sh cust /dev/rlvcust62 783M &
addfile.sh cust /dev/rlvcust63 783M &
addfile.sh cust /dev/rlvcust64 783M &
addfile.sh cust /dev/rlvcust65 783M &
addfile.sh cust /dev/rlvcust66 783M &
addfile.sh cust /dev/rlvcust67 783M &
addfile.sh cust /dev/rlvcust68 783M &
wait
#
addfile.sh stocks /dev/rlvstock1 1750M &
addfile.sh stocks /dev/rlvstock2 1750M &
addfile.sh stocks /dev/rlvstock3 1750M &
addfile.sh stocks /dev/rlvstock4 1750M &
addfile.sh stocks /dev/rlvstock5 1750M &
addfile.sh stocks /dev/rlvstock6 1750M &
addfile.sh stocks /dev/rlvstock7 1750M &
addfile.sh stocks /dev/rlvstock8 1750M &
addfile.sh stocks /dev/rlvstock9 1750M &
addfile.sh stocks /dev/rlvstock10 1750M &
addfile.sh stocks /dev/rlvstock11 1750M &
addfile.sh stocks /dev/rlvstock12 1750M &
addfile.sh stocks /dev/rlvstock13 1750M &
addfile.sh stocks /dev/rlvstock14 1750M &
addfile.sh stocks /dev/rlvstock15 1750M &
addfile.sh stocks /dev/rlvstock16 1750M &
addfile.sh stocks /dev/rlvstock17 1750M &
addfile.sh stocks /dev/rlvstock18 1750M &
addfile.sh stocks /dev/rlvstock19 1750M &
addfile.sh stocks /dev/rlvstock20 1750M &
wait
addfile.sh stocks /dev/rlvstock21 1750M &
addfile.sh stocks /dev/rlvstock22 1750M &
addfile.sh stocks /dev/rlvstock23 1750M &
addfile.sh stocks /dev/rlvstock24 1750M &
addfile.sh stocks /dev/rlvstock25 1750M &
addfile.sh stocks /dev/rlvstock26 1750M &
addfile.sh stocks /dev/rlvstock27 1750M &
addfile.sh stocks /dev/rlvstock28 1750M &
addfile.sh stocks /dev/rlvstock29 1750M &
addfile.sh stocks /dev/rlvstock30 1750M &
addfile.sh stocks /dev/rlvstock31 1750M &
addfile.sh stocks /dev/rlvstock32 1750M &
addfile.sh stocks /dev/rlvstock33 1750M &
addfile.sh stocks /dev/rlvstock34 1750M &
addfile.sh stocks /dev/rlvstock35 1750M &
addfile.sh stocks /dev/rlvstock36 1750M &
addfile.sh stocks /dev/rlvstock37 1750M &
addfile.sh stocks /dev/rlvstock38 1750M &
addfile.sh stocks /dev/rlvstock39 1750M &
addfile.sh stocks /dev/rlvstock40 1750M &
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

```
#
# $Header: benchsetup.sh 7030100.2 96/05/16 17:59:17 plai Generic<base> $ Copyr (c)
1995 Oracle
#
#
#+=================================================================+
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
```

```
#=====================================================================+
# FILENAME
# benchsetup.sh
# DESCRIPTION
# Usage: benchsetup.sh [options]
# -mu <multiplier> (# of warehouses)
# -nd do not run benchdb.sh
# -nt do not create tpcc tables
# -nx do not create index for tpcc tables
#=====================================================================
#
BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$TPCC_SCRIPTS/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
OUTDIR=outdir
MULT=1700
PATH=${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 "Bag arg: $1"
exit 1;
;;
esac
done
if [ "$MULT" = "" ]
then
echo $N "Database multiplier (# of warehouses)? [1]" $C
read MULT
if [ "$MULT" = "" ]
then
MULT=1
fi
fi
if [ ! -d $OUTDIR ]
then
mkdir $OUTDIR
fi
#
# Create database.
#
if [ "$NO_DB" = "" ]
then
benchdb.sh
fi
switchlog.sh
#
#
# Create tables.
#
if [ "$NO_TAB" = "" ]
then
sqlplus system/manager @$BUILD_SQL/tpcc_tab
sqlplus system/manager @$BUILD_SQL/tpcc_rol
fi
#
# Load history, new-order, order, order-line tables
#
pload.sh > ${OUTDIR}/pload.out 2>&1
switchlog.sh
#
# Create customer and stock tables.
#
if [ "$NO_TAB" = "" ]
then
sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_tab2 > ${OUTDIR}/tab2.out 2>&1 &
sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_tab3 > ${OUTDIR}/tab3.out 2>&1 &
fi
wait
switchlog.sh
#
# Load warehouse, district, item tables
#
tpccload -M $MULT -w
tpccload -M $MULT -d
tpccload -M $MULT -i
#
#
#
# Load customer table (in parallel with loading stock table)
#
I=1
SW=1
EW=50
INC=50
while [ $I -le 34 ]
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 stock table (in parallel with loading customer table)
#
I=1
SI=1
EI=2000
INC=2000
while [ $I -le 50 ]
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 + $INC`
done
wait
switchlog.sh
#
#
# Create indexes
#
if [ "$NO_IND" = "" ]
then
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;
!
sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_ix1
sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_ix2
svrmgrl <<!
connect internal
alter tablespace temp
default storage (initial 20K next 20K pctincrease 50);
exit;
!
fi
#
# Analyze tables and indexes
#
sqlplus tpcc/tpcc @TPCC_SQL/tpcc_ana
#
# Create table for processing benchmark results
#
sqlplus sys/change_on_install @$GEN_SQL/orst_cre
sqlplus sys/change_on_install @$TPCC_SQL/c_stat
sqlplus sys/change_on_install @$GEN_SQL/pst_c
#
sqlplus tpcc/tpcc @$ORACLE_HOME/bench/tpc/tpcc/blocks/pay
sqlplus tpcc/tpcc @$ORACLE_HOME/bench/tpc/tpcc/blocks/views
#
# Get some statistics
#
$TPCC_SCRIPTS/utils/ext_all.sh > ${OUTDIR}/ext_all.out 2>&1
$TPCC_SCRIPTS/utils/space_init.sh
$TPCC_SCRIPTS/utils/space_get.sh 1651 18686
$TPCC_SCRIPTS/utils/space_rpt.sh ${OUTDIR}/space.rpt
sqlplus system/manager <<!
alter user tpcc temporary tablespace system;
quit;
!
sqlplus sys/change_on_install <<!
grant execute on dbms_lock to public;
grant execute on dbms_pipe to public;
grant select on v_\$parameter to public;
quit;
!
sqlplus tpcc/tpcc @$AUDIT_SQL/plsql_mon
sqlplus tpcc/tpcc @$AUDIT_SQL/cre_tab
# Make blocks from customer table use recycle buffer pool
sqlplus tpcc/tpcc <<!
alter cluster ccluster storage ( buffer_pool recycle);
quit;
!
svrmgrl <<!
connect internal;
drop tablespace temp including contents;
exit;
!
sqlplus tpcc/tpcc <<!
alter table history storage (next 299M);
alter cluster ccluster storage (next 10M);
alter cluster scluster storage (next 4M);
alter table orders storage (next 242M);
alter table order_line storage (next 699M);
alter table new_order storage (next 149M);
alter index iorders storage (next 219M);
alter index iorders2 storage (next 214M);
alter index inew_order storage (next 98M);
alter index iorder_line storage (next 190M);
alter index istock storage (next 110M);
alter index icustomer storage (next 116M);
alter index icustomer2 storage (next 25M);
```

```
quit;
!
dml.sh
# add logs
log.sh
svrmgrl <<!
set echo off;
connect sys/change_on_install;
@?/rdbms/admin/catparr;
exit;
!
#
# Shutdown database
#
svrmgrl <<!
connect internal;
alter system switch logfile;
alter system switch logfile;
shutdown;
exit;
!
date
```

dml.sh

```
#
# $Header: dml.sh 7030100.1 96/05/02 10:22:52 plai Generic<base> $ Copyr (c) 1995 Oracle
#
#=================================================================+
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=================================================================+
# FILENAME
# dml.sh
# DESCRIPTION
# Disable table locks for TPC-C tables.
# USAGE
# dml.sh
#=================================================================*/
sqlplus tpcc/tpcc <<!
alter table warehouse disable table lock;
alter table district disable table lock;
alter table customer disable table lock;
alter table history disable table lock;
alter table item disable table lock;
alter table stock disable table lock;
alter table orders disable table lock;
alter table new_order disable table lock;
alter table order_line disable table lock;
quit;
!
```

pload.sh

```
#
# $Header: pload.sh 7030100.1 96/05/02 19:06:06 plai Generic<base> $ Copyr (c) 1995
Oracle
#
#
#=================================================================+
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=================================================================+
# FILENAME
# pload.sh
# DESCRIPTION
# Usage: pload.sh [options]
# -mu <multiplier> (# of warehouses)
# MODIFIED
#=================================================================
#
BENCH_HOME=$ORACLE_HOME/bench/tpc
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_LOADER=$BENCH_HOME/tpcc/loader
LDIR=data
OUTDIR=outdir
MULT=1700
PATH=${PATH}:$TPCC_SOURCE
export PATH
if echo "\c" | grep c >/dev/null 2>&1; then
N='-n'
else
C='\c'
fi
export N C
while [ "$#" != "0" ]
do
case $1 in
-mu) shift
if [ "$1" != "" ]
then
MULT=$1
shift
fi
;;
-nd) shift
NO_DB="y"
;;
-nt) shift
NO_TAB="y"
;;
-nx) shift
NO_IND="y"
;;
*) echo "Bag arg: $1"
exit 1;
;;
esac
done
if [ "$MULT" = "" ]
then
echo $N "Database multiplier (# of warehouses)? [1]" $C
read MULT
if [ "$MULT" = "" ]
then
MULT=1
fi
fi
if [ ! -d $LDIR ]
then
mkdir $LDIR
fi
if [ ! -d $OUTDIR ]
then
mkdir $OUTDIR
fi
#
# Load history table
#
I=1
while [ $I -le 4 ]
do
mknod ${LDIR}/hist${I}.dat p
I=`expr $I + 1`
done
I=1
SW=1
EW=425
INC=425
while [ $I -le 4 ]
do
tpccload -M $MULT -h -g -b $SW -e $EW > ${LDIR}/hist${I}.dat 2> \
${OUTDIR}/hist${I}.out &
I=`expr $I + 1`
SW=`expr $SW + $INC`
EW=`expr $EW + $INC`
done
sleep 30
I=1
while [ $I -le 4 ]
do
sqlldr tpcc/tpcc control=$TPCC_LOADER/hist.ctl \
log=${OUTDIR}/hist${I}.log \
bad=${OUTDIR}/hist${I}.bad data=${LDIR}/hist${I}.dat \
discard=${OUTDIR}/hist${I}.dsc \
file=/dev/rlvhist${I} &
I=`expr $I + 1`
done
wait
I=1
while [ $I -le 4 ]
do
rm -f ${LDIR}/hist${I}.dat
I=`expr $I + 1`
done
#
# Load new-order table
#
I=1
mknod ${LDIR}/neword${I}.dat p
SW=1
tpccload -M $MULT -n -g -b $SW -e $MULT > ${LDIR}/neword${I}.dat 2> \
${OUTDIR}/neword${I}.out &
sleep 30
sqlldr tpcc/tpcc control=$TPCC_LOADER/neword.ctl \
log=${OUTDIR}/neword${I}.log \
bad=${OUTDIR}/neword${I}.bad data=${LDIR}/neword${I}.dat \
discard=${OUTDIR}/neword${I}.dsc \
file=/dev/rlvnord${I} &
wait
rm -f ${LDIR}/neword${I}.dat
#
# Load order and order-line table
#
I=1
while [ $I -le 32 ]
do
mknod ${LDIR}/order${I}.dat p
mknod ${LDIR}/ordline${I}.dat p
I=`expr $I + 1`
done
I=1
SW=1
EW=54
INC=54
while [ $I -le 31 ]
do
tpccload -M $MULT -o ${LDIR}/ordline${I}.dat -g -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 -g -b $SW -e $MULT > \
${LDIR}/order${I}.dat 2> ${OUTDIR}/order${I}.out &
sleep 30
I=1
while [ $I -le 32 ]
do
J=`expr $I - 1`
J=`expr $J / 8`
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 \
```

```
discard=${OUTDIR}/order${I}.dsc \
file=/dev/rlvord${J} &
sqlldr tpcc/tpcc control=$TPCC_LOADER/ordline.ctl \
log=${OUTDIR}/ordline${I}.log \
bad=${OUTDIR}/ordline${I}.bad data=${LDIR}/ordline${I}.dat \
discard=${OUTDIR}/ordline${I}.dsc \
file=/dev/rlvordl${I} &
I=`expr $I + 1`
done
wait
I=1
while [ $I -le 32 ]
do
rm -f ${LDIR}/order${I}.dat
rm -f ${LDIR}/ordline${I}.dat
I=`expr $I + 1`
done
```

switchlog.sh

```
#!/bin/ksh
#
# $Header: switchlog.sh 7030100.1 96/05/02 10:20:11 plai Generic<base> $ Copyr (c) 1995
Oracle
#
#=============================================================+
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=============================================================+
# FILENAME
# switchlog.sh
# DESCRIPTION
# Switch to next log file twice.
# USAGE
# switchlog.sh
#=============================================================*/
svrmgrl <<!
connect internal;
alter system switch logfile;
alter system switch logfile;
exit;
!
```

undml.sh

```
#

# $Header: undml.sh 7030100.2 96/05/02 10:29:30 plai Generic<base> $ Copyr (c) 1995
Oracle
#
#=============================================================+
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=============================================================+
# FILENAME
# undml.sh
# DESCRIPTION
# Enable table locks for TPC-C tables.
# USAGE
# undml.sh
#=============================================================*/
sqlplus tpcc/tpcc <<!
alter table warehouse enable table lock;
alter table district enable table lock;
alter table customer enable table lock;
alter table history enable table lock;
alter table item enable table lock;
alter table stock enable table lock;
alter table orders enable table lock;
alter table new_order enable table lock;
alter table order_line enable table lock;
quit;
!
```

C.2 SQL Scripts

tpcc_ana.sql

```
rem
rem =============================================================+
rem Copyright (c) 1995 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =============================================================+
rem FILENAME
rem tpcc_ana.sql
rem DESCRIPTION
rem Analyze all tables and indexes of TPC-C database.
rem =============================================================
rem
set timing on;
analyze table warehouse compute statistics;
analyze table district compute statistics;
analyze table item estimate statistics;
analyze table history estimate statistics;
analyze table customer estimate statistics;
analyze table stock estimate statistics;
analyze table orders estimate statistics;
analyze table new_order estimate statistics;
analyze table order_line estimate statistics;
analyze cluster icluster estimate statistics;
analyze cluster scluster estimate statistics;
analyze cluster ccluster estimate statistics;
analyze index iwarehouse compute statistics;
analyze index idistrict compute statistics;
analyze index icustomer estimate statistics;
analyze index icustomer2 estimate statistics;
analyze index istock estimate statistics;
analyze index iitem estimate statistics;
analyze index iorders estimate statistics;
analyze index iorders2 estimate statistics;
analyze index inew_order estimate statistics;
analyze index iorder_line estimate statistics;
quit;
```

tpcc_ix1.sql

```
rem
rem =============================================================+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =============================================================+
rem FILENAME
rem tpcc_ix1.sql
rem DESCRIPTION
rem Create indexes for TPC-C database.
rem =============================================================
rem
drop index iwarehouse;
drop index idistrict;
drop index icustomer;
drop index icustomer2;
drop index istock;
drop index iitem;
set timing on
create unique index iwarehouse on warehouse(w_id)
tablespace ware
initrans 3
storage (initial 1400K next 1M pctincrease 0) pctfree 1;
create unique index idistrict on district(d_w_id, d_id)
tablespace ware
initrans 3
storage (initial 13M next 1M pctincrease 0) pctfree 1;
create unique index iitem on item(i_id)
tablespace items
storage (initial 8M next 100K pctincrease 0) pctfree 1;
create unique index icustomer on customer(c_w_id, c_d_id, c_id)
tablespace icust1
initrans 3
parallel 2
storage (initial 20M next 20M pctincrease 0) pctfree 1;
create unique index icustomer2 on customer(c_last, c_w_id, c_d_id, c_first, c_id)
tablespace icust2
initrans 3
parallel 8
storage (initial 25M next 25M pctincrease 0) pctfree 1;
create unique index istock on stock(s_i_id, s_w_id)
tablespace istk
parallel 3
storage (initial 50M next 50M pctincrease 0) pctfree 1;
exit;
```

tpcc_ix2.sql

```
rem
rem =============================================================+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =============================================================+
rem FILENAME
rem tpcc_ix2.sql
rem DESCRIPTION
rem Create indexes for TPC-C database.
rem =============================================================
rem
drop index iorders;
drop index iorders2;
drop index inew_order;
drop index iorder_line;
set timing on
create unique index iorders on orders(o_w_id, o_d_id, o_id)
tablespace iord1
initrans 3
parallel 6
pctfree 1
storage (initial 20M next 20M pctincrease 0
freelist groups 40 freelists 9);
create unique index iorders2 on orders(o_w_id, o_d_id, o_c_id, o_id)
tablespace iord2
initrans 3
parallel 11
pctfree 25
storage (initial 25M next 25M pctincrease 0
freelist groups 40 freelists 9);
create unique index inew_order on new_order(no_w_id, no_d_id, no_o_id)
tablespace inord
initrans 4
parallel 6
pctfree 5
storage (initial 10M next 10M pctincrease 0
freelist groups 40 freelists 9);
create unique index iorder_line on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
tablespace iordl
initrans 4
parallel 16
pctfree 1
storage (initial 140M next 140M pctincrease 0
freelist groups 40 freelists 9);
exit;
```

tpcc_rol.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_rol.sql
rem DESCRIPTION
rem Create rollback segments for TPCC database.
rem ================================================================
rem
set timing on;
host date;
CREATE ROLLBACK SEGMENT t1
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t2
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t3
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t4
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t5
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t6
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t7
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t8
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t9
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t10
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t11
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t12
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t13
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t14
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t15
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t16
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t17
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t18
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t19
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t20
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t21
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t22
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t23
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t24
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t25
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t26
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t27
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t28
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t29
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t30
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t31
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t32
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t33
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t34
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t35
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t36
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t37
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t38
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t39
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t40
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t41
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t42
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t43
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t44
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t45
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t46
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t47
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t48
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t49
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t50
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t51
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t52
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t53
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t54
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t55
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t56
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t57
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t58
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t59
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t60
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t61
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t62
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t63
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t64
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t65
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t66
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t67
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t68
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t69
TABLESPACE roll
```

```
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t70
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t71
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t72
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t73
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t74
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t75
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t76
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t77
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t78
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t79
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t80
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t81
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t82
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t83
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t84
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t85
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t86
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t87
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t88
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t89
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t90
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t91
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t92
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t93
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t94
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t95
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t96
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t97
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t98
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t99
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t100
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t101
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t102
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t103
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t104
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t105
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t106
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t107
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t108
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t109
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t110
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t111
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t112
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t113
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t114
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t115
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t116
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t117
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t118
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t119
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t120
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t121
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t122
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t123
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t124
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t125
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t126
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t127
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t128
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t129
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t130
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t131
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t132
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t133
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t134
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t135
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t136
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t137
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t138
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t139
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t140
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t141
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t142
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
```

```
CREATE ROLLBACK SEGMENT t143
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t144
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t145
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t146
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t147
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t148
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t149
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t150
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t151
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t152
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t153
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t154
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t155
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t156
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t157
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t158
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t159
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t160
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t161
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t162
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t163
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t164
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t165
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t166
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t167
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t168
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t169
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t170
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t171
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t172
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t173
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t174
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t175
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t176
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t177
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t178
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t179
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t180
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t181
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t182
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t183
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t184
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t185
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t186
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t187
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t188
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t189
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t190
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t191
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t192
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t193
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t194
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t195
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t196
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t197
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t198
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t199
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t200
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t201
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t202
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t203
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t204
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t205
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t206
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t207
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t208
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t209
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t210
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t211
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t212
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t213
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t214
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t215
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t216
```

```
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t217
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t218
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t219
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t220
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t221
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t222
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t223
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t224
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t225
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t226
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t227
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t228
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t229
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t230
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t231
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t232
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t233
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t234
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t235
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t236
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t237
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t238
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t239
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t240
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t241
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t242
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t243
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t244
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t245
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t246
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t247
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t248
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t249
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t250
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t251
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t252
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t253
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t254
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t255
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t256
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t257
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t258
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t259
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t260
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t261
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t262
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t263
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t264
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t265
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t266
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t267
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t268
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t269
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t270
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t271
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t272
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t273
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t274
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t275
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t276
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t277
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t278
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t279
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t280
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t281
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t282
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t283
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t284
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t285
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t286
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t287
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t288
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t289
TABLESPACE roll
```

```
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t290
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t291
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t292
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t293
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t294
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t295
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t296
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t297
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t298
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t299
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t300
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
host date;
exit;
```

tpcc_tab.sql

```
rem
rem ============================================================+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem ============================================================+
rem FILENAME
rem tpcc_tab.sql
rem DESCRIPTION
rem Create tables for TPC-C database.
rem L.R change extent size and freelist groups for 1700 warehouses
rem ============================================================
rem
rem
rem FIRST, create TPCC userid and connect to it.
rem
grant connect,resource,unlimited tablespace to tpcc identified by tpcc;
alter user tpcc temporary tablespace temp;
connect tpcc/tpcc
rem
rem NEXT, DROP all first
rem
drop cluster icluster including tables;
drop table warehouse;
drop table district;
drop table history;
drop table orders;
drop table new_order;
drop table order_line;
drop table item;
set timing on
rem
rem LAST, CREATE all tables
rem
rem
rem WAREHOUSE table
rem
create table warehouse (
w_id number,
w_ytd number(12),
w_tax number(4),
w_name varchar2(10),
w_street_1 varchar2(20),
w_street_2 varchar2(20),
w_city varchar2(20),
w_state char(2),
w_zip char(9)
)
tablespace ware
initrans 4
pctfree 97 pctused 2
storage (initial 4000K next 1M pctincrease 0);
rem
rem DISTRICT table
rem
create table district (
d_id number,
d_w_id number,
d_ytd number(12),
d_tax number(4),
d_next_o_id number,
d_name varchar2(10),
d_street_1 varchar2(20),
d_street_2 varchar2(20),
d_city varchar2(20),
d_state char(2),
d_zip char(9)
)
tablespace ware
initrans 4
pctfree 97 pctused 2
storage (initial 36000K next 1M pctincrease 0);
rem
rem HISTORY table
rem
create table history (
h_c_id number,
h_c_d_id number,
h_c_w_id number,
h_d_id number,
h_w_id number,
h_date date,
h_amount number(6),
h_data varchar2(24)
)
tablespace hist
initrans 3
pctfree 1
storage (initial 80K next 500M pctincrease 0
freelist groups 40 freelists 9);
rem
rem ORDER table
rem
create table orders (
o_id number,
o_d_id number,
o_w_id number,
o_c_id number,
o_entry_d date,
o_carrier_id number,
o_ol_cnt number,
o_all_local number
)
tablespace ord
initrans 3
pctfree 5
storage (initial 80K next 50M pctincrease 0
freelist groups 40 freelists 9);
rem
rem NEW_ORDER table
rem
create table new_order (
no_o_id number,
no_d_id number,
no_w_id number
)
tablespace nord
initrans 4
pctfree 5
storage (initial 80K next 413M pctincrease 0
freelist groups 40 freelists 9);
rem
rem ORDER_LINE table
rem
create table order_line (
ol_o_id number,
ol_d_id number,
ol_w_id number,
ol_number number,
ol_delivery_d date,
ol_i_id number,
ol_supply_w_id number,
ol_quantity number,
ol_amount number(6),
ol_dist_info char(24)
)
tablespace ordl
initrans 4
pctfree 5
storage (initial 80K next 1886M pctincrease 0
freelist groups 40 freelists 9);
rem
rem ITEM table
rem
create cluster icluster (
i_id number(6,0)
)
hashkeys 100000
hash is i_id
size 120
initrans 3
pctfree 0
tablespace items
storage (initial 14M next 1M pctincrease 0);
create table item (
i_id number(6,0),
i_im_id number,
i_name varchar2(24),
i_price number(5,0),
i_data varchar2(50)
)
cluster icluster(i_id);
rem
rem done
rem
exit;
```

tpcc_tab2.sql

```
rem
rem ============================================================+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem ============================================================+
rem FILENAME
rem tpcc_tab2.sql
```

```
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem ============================================================
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(4,0)
)
hashkeys 51000000
hash is (c_w_id * 30000 + c_d_id * 3000 + c_id)
size 850
initrans 3
pctfree 0
tablespace cust
storage (initial 772M next 772M pctincrease 0 minextents 68);
create table customer (
c_id number(5,0),
c_d_id number(2,0),
c_w_id number(4,0),
c_first varchar2(16),
c_middle char(2),
c_last varchar2(16),
c_street_1 varchar2(20),
c_street_2 varchar2(20),
c_city varchar2(20),
c_state char(2),
c_zip char(9),
c_phone char(16),
c_since date,
c_credit char(2),
c_credit_lim number(12),
c_discount number(4),
c_balance number(12),
c_ytd_payment number(12),
c_payment_cnt number(8),
c_delivery_cnt number(8),
c_data varchar2(500)
)
cluster ccluster (c_id, c_d_id, c_w_id);
rem
rem done
rem
exit;
```

tpcc_tab3.sql

```
rem
rem ================================================================+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem ================================================================+
rem FILENAME
rem tpcc_tab3.sql
rem DESCRIPTION
rem Create stock table for TPC-C database.
rem ============================================================
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(4,0)
)
hashkeys 170000000
hash is (s_i_id * 1700 + s_w_id)
size 350
initrans 3
pctfree 0
tablespace stocks
storage (initial 1746M next 1746M pctincrease 0 minextents 40);
create table stock (
s_i_id number(6,0),
s_w_id number(4,0),
s_quantity number(6,0),
s_dist_01 char(24),
s_dist_02 char(24),
s_dist_03 char(24),
s_dist_04 char(24),
s_dist_05 char(24),
s_dist_06 char(24),
s_dist_07 char(24),
s_dist_08 char(24),
s_dist_09 char(24),
s_dist_10 char(24),
s_ytd number(10,0),
s_order_cnt number(6,0),
s_remote_cnt number(6,0),
s_data varchar2(50)
)
cluster scluster (s_i_id, s_w_id);
rem
rem done
rem
exit;
```

C.3 Data Generation Code

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 VALUES (:w_id, 30000000, :w_tax, :w_name,
:w_street_1, \
:w_street_2, :w_city, :w_state, :w_zip)"
#define SQLTXTD "INSERT INTO district VALUES (:d_id, :d_w_id,3000000, :d_tax, \
3001, :d_name, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip)"
#define SQLTXTC "INSERT INTO customer VALUES (:c_id, :c_d_id, :c_w_id, \
:c_first, 'OE', :c_last, :c_street_1, :c_street_2, :c_city, :c_state, \
:c_zip, :c_phone, SYSDATE, :c_credit, 5000000, :c_discount, -1000, 1000, 1, \
0, :c_data)"
#define SQLTXTH "INSERT INTO history VALUES (:h_c_id, :h_c_d_id, :h_c_w_id, \
:h_d_id, :h_w_id, SYSDATE, 1000, :h_data)"
#define SQLTXTS "INSERT INTO stock VALUES (:s_i_id, :s_w_id, :s_quantity, \
:s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05, :s_dist_06, \
:s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10, 0, 0, 0, :s_data)" \
#define SQLTXTI "INSERT INTO item VALUES (:i_id, :i_im_id, :i_name, :i_price, \
:i_data)"
#define SQLTXTO1 "INSERT INTO orders VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
SYSDATE, :o_carrier_id, :o_ol_cnt, 1)"
#define SQLTXTO2 "INSERT INTO orders VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
SYSDATE, 11, :o_ol_cnt, 1)"
#define SQLTXTOL1 "INSERT INTO order_line VALUES (:ol_o_id, :ol_d_id, \
:ol_w_id, :ol_number, SYSDATE, :ol_i_id, :ol_supply_w_id, 5, 0, \
:ol_dist_info)"
#define SQLTXTOL2 "INSERT INTO order_line VALUES (:ol_o_id, :ol_d_id, \
:ol_w_id, :ol_number, to_date('01-Jan-1811'), :ol_i_id, :ol_supply_w_id, 5, :ol_amount, \
:ol_dist_info)"
#define SQLTXTNO "INSERT INTO new_order VALUES (:no_o_id, :no_d_id, :no_w_id)"
ldadef tpclda;
csrdef curw, curd, curc, curh, curs, curi, curo1, curo2, curol1, curol2, curno;
unsigned long tpchda[256];
static char *lastname[] = {
"BAR",
"OUGHT",
"ABLE",
"PRI",
"PRES",
"ESE",
"ANTI",
"CALLY",
"ATION",
```

```
"EING"
};
char num9[10];
char num16[17];
char str2[3];
char str24[15][25];
int randperm3000[3000];
myusage()
{
fprintf (stderr, "\n");
fprintf (stderr, "Usage:\ttpccload -M <multiplier> [options]\n");
fprintf (stderr, "options:\n");
fprintf (stderr, "\t-A :\tload all tables\n");
fprintf (stderr, "\t-w :\tload warehouse table\n");
fprintf (stderr, "\t-d :\tload district table\n");
fprintf (stderr, "\t-c :\tload customer table\n");
fprintf (stderr, "\t-i :\tload item table\n");
fprintf (stderr, "\t-s :\tload stock table (cluster around s_w_id)\n");
fprintf (stderr, "\t-S :\tload stock table (cluster around s_i_id)\n");
fprintf (stderr, "\t-h :\tload history table\n");
fprintf (stderr, "\t-n :\tload new-order table\n");
fprintf (stderr, "\t-o <oline file> :\tload order and order-line table\n");
fprintf (stderr, "\t-b <ware#> :\tbeginning warehouse number\n");
fprintf (stderr, "\t-e <ware#> :\tending warehouse number\n");
fprintf (stderr, "\t-j <item#> :\tbeginning item number (with -S)\n");
fprintf (stderr, "\t-k <item#> :\tending item number (with -S)\n");
fprintf (stderr, "\t-g :\tgenerate rows to standard output\n");
fprintf (stderr, "\n");
exit(1);
}
errrpt (lda, cur)
csrdef *lda;
csrdef *cur;
{
text msg[2048];
if (cur->rc) {
oerhms (lda, cur->rc, msg, 2048);
fprintf (stderr, "TPC-C load error: %s\n", msg);
}
}
quit ()
{
if (oclose (&curw))
errrpt (&tpclda, &curw);
if (oclose (&curd))
errrpt (&tpclda, &curd);
if (oclose (&curc))
errrpt (&tpclda, &curc);
if (oclose (&curh))
errrpt (&tpclda, &curh);
if (oclose (&curs))
errrpt (&tpclda, &curs);
if (oclose (&curi))
errrpt (&tpclda, &curi);
if (oclose (&curo1))
errrpt (&tpclda, &curo1);
if (oclose (&curo2))
errrpt (&tpclda, &curo2);
if (oclose (&curol1))
errrpt (&tpclda, &curol1);
if (oclose (&curol2))
errrpt (&tpclda, &curol2);
if (oclose (&curno))
errrpt (&tpclda, &curno);
if (ologof (&tpclda))
fprintf (stderr, "TPC-C load error: Error in logging off\n");
}
main (argc, argv)
int argc;
char *argv[];
{
char *uid="tpcc/tpcc";
text sqlbuf[1024];
int scale=0;
int i, j;
int loop;
int loopcount;
int cid;
int dwid;
int cdid;
int cwid;
int sid;
int swid;
int olcnt;
int nrows;
int row;
int w_id;
char w_name[11];
char w_street_1[21];
char w_street_2[21];
char w_city[21];
char w_state[2];
char w_zip[9];
int w_tax;
int d_id[10];
int d_w_id[10];
char d_name[10][11];
char d_street_1[10][21];
char d_street_2[10][21];
char d_city[10][21];
char d_state[10][2];
char d_zip[10][9];
int d_tax[10];
int c_id[100];
int c_d_id[100];
int c_w_id[100];
char c_first[100][17];
char c_last[100][17];
char c_street_1[100][21];
char c_street_2[100][21];
char c_city[100][21];
char c_state[100][2];
char c_zip[100][9];
char c_phone[100][16];
char c_credit[100][2];
int c_discount[100];
char c_data[100][501];
int i_id[100];
int i_im_id[100];
int i_price[100];
char i_name[100][25];
char i_data[100][51];
int s_i_id[100];
int s_w_id[100];
int s_quantity[100];
char s_dist_01[100][24];
char s_dist_02[100][24];
char s_dist_03[100][24];
char s_dist_04[100][24];
char s_dist_05[100][24];
char s_dist_06[100][24];
char s_dist_07[100][24];
char s_dist_08[100][24];
char s_dist_09[100][24];
char s_dist_10[100][24];
char s_data[100][51];
int h_w_id[100];
int h_d_id[100];
int h_c_id[100];
char h_data[100][25];
int o_id[100];
int o_d_id[100];
int o_w_id[100];
int o_c_id[100];
int o_carrier_id[100];
int o_ol_cnt[100];
int ol_o_id[15];
int ol_d_id[15];
int ol_w_id[15];
int ol_number[15];
int ol_i_id[15];
int ol_supply_w_id[15];
int ol_amount[15];
char ol_dist_info[15][24];
int no_o_id[100];
int no_d_id[100];
int no_w_id[100];
char sdate[30];
double begin_time, end_time;
double begin_cpu, end_cpu;
double gettime(), getcpu();
extern int getopt();
extern char *optarg;
extern int optind, opterr;
char *argstr="M:AwdcisShno:b:e:j:k:g";
int opt;
int do_A=0;
int do_w=0;
int do_d=0;
int do_i=0;
int do_c=0;
int do_s=0;
int do_S=0;
int do_h=0;
int do_o=0;
int do_n=0;
int gen=0;
int bware=1;
int eware=0;
int bitem=1;
int eitem=0;
FILE *olfp=NULL;
char olfname[100];
/*---------------------------------------------------------------+
| Parse command line -- look for scale factor. |
+---------------------------------------------------------------*/
if (argc == 1) {
myusage ();
}
while ((opt = getopt (argc, argv, argstr)) != -1) {
switch (opt) {
case '?': myusage ();
break;
case 'M': scale = atoi (optarg);
break;
case 'A': do_A = 1;
break;
case 'w': do_w = 1;
break;
case 'd': do_d = 1;
break;
case 'c': do_c = 1;
break;
case 'i': do_i = 1;
break;
case 's': do_s = 1;
break;
case 'S': do_S = 1;
break;
case 'h': do_h = 1;
break;
case 'n': do_n = 1;
break;
case 'o': do_o = 1;
strcpy (olfname, optarg);
break;
case 'b': bware = atoi (optarg);
break;
case 'e': eware = atoi (optarg);
break;
```

```
case 'j': bitem = atoi (optarg);
break;
case 'k': eitem = atoi (optarg);
break;
case 'g': gen = 1;
break;
default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
fprintf (stderr, "(reached default case in getopt ())\n");
myusage ();
}
}
/*---------------------------------------------------------------*|
| Rudimentary error checking |
|*---------------------------------------------------------------*/
if (scale < 1) {
fprintf (stderr, "Invalid scale factor: '%d'\n", scale);
myusage ();
}
if (!(do_A || do_w || do_d || do_c || do_i || do_s || do_S || do_h || do_o ||
do_n)) {
fprintf (stderr, "What should I load???\n");
myusage ();
}
if (gen && (do_A || (do_w + do_d + do_c + do_i + do_s + do_S + do_h + do_o +
do_n > 1))) {
fprintf (stderr, "Can only generate table one at a time\n");
myusage ();
}
if (do_S && (do_A || do_s)) {
fprintf (stderr, "Cluster stock table around s_w_id or s_i_id?\n");
myusage ();
}
if (eware <= 0)
eware = scale;
if (eitem <= 0)
eitem = STOCFAC;
if (do_S) {
if ((bitem < 1) || (bitem > STOCFAC)) {
fprintf (stderr, "Invalid beginning item number: '%d'\n", bitem);
myusage ();
}
if ((eitem < bitem) || (eitem > STOCFAC)) {
fprintf (stderr, "Invalid ending item number: '%d'\n", eitem);
myusage ();
}
}
if ((bware < 1) || (bware > scale)) {
fprintf (stderr, "Invalid beginning warehouse number: '%d'\n", bware);
myusage ();
}
if ((eware < bware) || (eware > scale)) {
fprintf (stderr, "Invalid ending warehouse number: '%d'\n", eware);
myusage ();
}
if (gen && do_o) {
if ((olfp = fopen (olfname, "w")) == NULL) {
fprintf (stderr, "Can't open '%s' for writing order lines\n", olfname);
myusage ();
}
}
/*---------------------------------------------------------------+
| Prepare to insert into database. |
+---------------------------------------------------------------*/
sysdate (sdate);
if (!gen) {
/* log on to Oracle */
if (orlon (&tpclda, (ub1 *) tpchda, (text *) uid, -1, (text *) 0, -1, 0)) {
fprintf (stderr, "TPC-C load error: Error in logging on\n");
errrpt (&tpclda, &tpclda);
exit (1);
}
fprintf (stderr, "\nConnected to Oracle userid '%s'.\n", uid);
/* turn off auto-commit */
if (ocof (&tpclda)) {
errrpt (&tpclda, &tpclda);
ologof (&tpclda);
exit (1);
}
/* open cursors */
if (oopen (&curw, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curw);
ologof (&tpclda);
exit (1);
}
if (oopen (&curd, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curd);
oclose (&curw);
ologof (&tpclda);
exit (1);
}
if (oopen (&curc, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curc);
oclose (&curw);
oclose (&curd);
ologof (&tpclda);
exit (1);
}
if (oopen (&curh, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curh);
oclose (&curw);
oclose (&curd);
oclose (&curc);
ologof (&tpclda);
exit (1);
}
if (oopen (&curs, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curs);
oclose (&curw);
oclose (&curd);
oclose (&curc);
oclose (&curh);
ologof (&tpclda);
exit (1);
}
if (oopen (&curi, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curi);
oclose (&curw);
oclose (&curd);
oclose (&curc);
oclose (&curh);
oclose (&curs);
ologof (&tpclda);
exit (1);
}
if (oopen (&curo1, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curo1);
oclose (&curw);
oclose (&curd);
oclose (&curc);
oclose (&curh);
oclose (&curs);
oclose (&curi);
ologof (&tpclda);
exit (1);
}
if (oopen (&curo2, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curo2);
oclose (&curw);
oclose (&curd);
oclose (&curc);
oclose (&curh);
oclose (&curs);
oclose (&curi);
oclose (&curo1);
ologof (&tpclda);
exit (1);
}
if (oopen (&curol1, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curol1);
oclose (&curw);
oclose (&curd);
oclose (&curc);
oclose (&curh);
oclose (&curs);
oclose (&curi);
oclose (&curo1);
oclose (&curo2);
ologof (&tpclda);
exit (1);
}
if (oopen (&curol2, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curol2);
oclose (&curw);
oclose (&curd);
oclose (&curc);
oclose (&curh);
oclose (&curs);
oclose (&curi);
oclose (&curo1);
oclose (&curo2);
oclose (&curol1);
ologof (&tpclda);
exit (1);
}
if (oopen (&curno, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
errrpt (&tpclda, &curno);
oclose (&curw);
oclose (&curd);
oclose (&curc);
oclose (&curh);
oclose (&curs);
oclose (&curi);
oclose (&curo1);
oclose (&curo2);
oclose (&curol1);
oclose (&curol2);
ologof (&tpclda);
exit (1);
}
/* parse statements */
sprintf ((char *) sqlbuf, SQLTXTW);
if (oparse (&curw, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curw);
quit ();
exit (1);
}
sprintf ((char *) sqlbuf, SQLTXTD);
if (oparse (&curd, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
sprintf ((char *) sqlbuf, SQLTXTC);
if (oparse (&curc, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
sprintf ((char *) sqlbuf, SQLTXTH);
if (oparse (&curh, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curh);
quit ();
exit (1);
}
sprintf ((char *) sqlbuf, SQLTXTS);
if (oparse (&curs, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
```

```
sprintf ((char *) sqlbuf, SQLTXTI);
if (oparse (&curi, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curi);
quit ();
exit (1);
}
sprintf ((char *) sqlbuf, SQLTXTO1);
if (oparse (&curo1, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curo1);
quit ();
exit (1);
}
sprintf ((char *) sqlbuf, SQLTXTO2);
if (oparse (&curo2, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curo2);
quit ();
exit (1);
}
sprintf ((char *) sqlbuf, SQLTXTOL1);
if (oparse (&curol1, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curol1);
quit ();
exit (1);
}
sprintf ((char *) sqlbuf, SQLTXTOL2);
if (oparse (&curol2, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curol2);
quit ();
exit (1);
}
sprintf ((char *) sqlbuf, SQLTXTNO);
if (oparse (&curno, sqlbuf, -1, 0, 1)) {
errrpt (&tpclda, &curno);
quit ();
exit (1);
}
/* bind variables */
/* warehouse */
if (obndrv (&curw, (text *) ":w_id", -1, (ub1 *) &w_id, sizeof (w_id),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curw);
quit ();
exit (1);
}
if (obndrv (&curw, (text *) ":w_name", -1, (ub1 *) w_name, 11,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curw);
quit ();
exit (1);
}
if (obndrv (&curw, (text *) ":w_street_1", -1, (ub1 *) w_street_1, 21,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curw);
quit ();
exit (1);
}
if (obndrv (&curw, (text *) ":w_street_2", -1, (ub1 *) w_street_2, 21,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curw);
quit ();
exit (1);
}
if (obndrv (&curw, (text *) ":w_city", -1, (ub1 *) w_city, 21,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curw);
quit ();
exit (1);
}
if (obndrv (&curw, (text *) ":w_state", -1, (ub1 *) w_state, 2,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curw);
quit ();
exit (1);
}
if (obndrv (&curw, (text *) ":w_zip", -1, (ub1 *) w_zip, 9,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curw);
quit ();
exit (1);
}
if (obndrv (&curw, (text *) ":w_tax", -1, (ub1 *) &w_tax, sizeof (w_tax),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curw);
quit ();
exit (1);
}
/* district */
if (obndrv (&curd, (text *) ":d_id", -1, (ub1 *) d_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
if (obndrv (&curd, (text *) ":d_w_id", -1, (ub1 *) d_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
if (obndrv (&curd, (text *) ":d_name", -1, (ub1 *) d_name, 11,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
if (obndrv (&curd, (text *) ":d_street_1", -1, (ub1 *) d_street_1, 21,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
if (obndrv (&curd, (text *) ":d_street_2", -1, (ub1 *) d_street_2, 21,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
if (obndrv (&curd, (text *) ":d_city", -1, (ub1 *) d_city, 21,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
if (obndrv (&curd, (text *) ":d_state", -1, (ub1 *) d_state, 2,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
if (obndrv (&curd, (text *) ":d_zip", -1, (ub1 *) d_zip, 9,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
if (obndrv (&curd, (text *) ":d_tax", -1, (ub1 *) d_tax, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curd);
quit ();
exit (1);
}
/* customer */
if (obndrv (&curc, (text *) ":c_id", -1, (ub1 *) c_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_d_id", -1, (ub1 *) c_d_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_w_id", -1, (ub1 *) c_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_first", -1, (ub1 *) c_first, 17,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_last", -1, (ub1 *) c_last, 17,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_street_1", -1, (ub1 *) c_street_1, 21,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_street_2", -1, (ub1 *) c_street_2, 21,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_city", -1, (ub1 *) c_city, 21,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_state", -1, (ub1 *) c_state, 2,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_zip", -1, (ub1 *) c_zip, 9,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_phone", -1, (ub1 *) c_phone, 16,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_credit", -1, (ub1 *) c_credit, 2,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
if (obndrv (&curc, (text *) ":c_discount", -1, (ub1 *) c_discount,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1,
-1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
```

```
}
if (obndrv (&curc, (text *) ":c_data", -1, (ub1 *) c_data, 501,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curc);
quit ();
exit (1);
}
/* item */
if (obndrv (&curi, (text *) ":i_id", -1, (ub1 *) i_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curi);
quit ();
exit (1);
}
if (obndrv (&curi, (text *) ":i_im_id", -1, (ub1 *) i_im_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curi);
quit ();
exit (1);
}
if (obndrv (&curi, (text *) ":i_name", -1, (ub1 *) i_name, 25,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curi);
quit ();
exit (1);
}
if (obndrv (&curi, (text *) ":i_price", -1, (ub1 *) i_price,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1,
-1)) {
errrpt (&tpclda, &curi);
quit ();
exit (1);
}
if (obndrv (&curi, (text *) ":i_data", -1, (ub1 *) i_data, 51,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curi);
quit ();
exit (1);
}
/* stock */
if (obndrv (&curs, (text *) ":s_i_id", -1, (ub1 *) s_i_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_w_id", -1, (ub1 *) s_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_quantity", -1, (ub1 *) s_quantity,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_01", -1, (ub1 *) s_dist_01, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_02", -1, (ub1 *) s_dist_02, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_03", -1, (ub1 *) s_dist_03, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_04", -1, (ub1 *) s_dist_04, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_05", -1, (ub1 *) s_dist_05, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_06", -1, (ub1 *) s_dist_06, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_07", -1, (ub1 *) s_dist_07, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_08", -1, (ub1 *) s_dist_08, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_09", -1, (ub1 *) s_dist_09, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_dist_10", -1, (ub1 *) s_dist_10, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
if (obndrv (&curs, (text *) ":s_data", -1, (ub1 *) s_data, 51,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curs);
quit ();
exit (1);
}
/* history */
if (obndrv (&curh, (text *) ":h_c_id", -1, (ub1 *) h_c_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curh);
quit ();
exit (1);
}
if (obndrv (&curh, (text *) ":h_c_d_id", -1, (ub1 *) h_d_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curh);
quit ();
exit (1);
}
if (obndrv (&curh, (text *) ":h_c_w_id", -1, (ub1 *) h_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curh);
quit ();
exit (1);
}
if (obndrv (&curh, (text *) ":h_d_id", -1, (ub1 *) h_d_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curh);
quit ();
exit (1);
}
if (obndrv (&curh, (text *) ":h_w_id", -1, (ub1 *) h_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curh);
quit ();
exit (1);
}
if (obndrv (&curh, (text *) ":h_data", -1, (ub1 *) h_data, 25,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curh);
quit ();
exit (1);
}
/* order_line (delivered) */
if (obndrv (&curol1, (text *) ":ol_o_id", -1, (ub1 *) ol_o_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol1);
quit ();
exit (1);
}
if (obndrv (&curol1, (text *) ":ol_d_id", -1, (ub1 *) ol_d_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol1);
quit ();
exit (1);
}
if (obndrv (&curol1, (text *) ":ol_w_id", -1, (ub1 *) ol_w_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol1);
quit ();
exit (1);
}
if (obndrv (&curol1, (text *) ":ol_number", -1, (ub1 *) ol_number,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol1);
quit ();
exit (1);
}
if (obndrv (&curol1, (text *) ":ol_i_id", -1, (ub1 *) ol_i_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol1);
quit ();
exit (1);
}
if (obndrv (&curol1, (text *) ":ol_supply_w_id", -1,
(ub1 *) ol_supply_w_id, sizeof (int), SQLT_INT, -1,
(sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol1);
quit ();
exit (1);
}
if (obndrv (&curol1, (text *) ":ol_dist_info", -1, (ub1 *) ol_dist_info,
24, SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol1);
quit ();
exit (1);
}
/* order_line (not delivered) */
if (obndrv (&curol2, (text *) ":ol_o_id", -1, (ub1 *) ol_o_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol2);
quit ();
exit (1);
}
if (obndrv (&curol2, (text *) ":ol_d_id", -1, (ub1 *) ol_d_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol2);
quit ();
exit (1);
}
if (obndrv (&curol2, (text *) ":ol_w_id", -1, (ub1 *) ol_w_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
```

```
errrpt (&tpclda, &curol2);
quit ();
exit (1);
}
if (obndrv (&curol2, (text *) ":ol_number", -1, (ub1 *) ol_number,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol2);
quit ();
exit (1);
}
if (obndrv (&curol2, (text *) ":ol_i_id", -1, (ub1 *) ol_i_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol2);
quit ();
exit (1);
}
if (obndrv (&curol2, (text *) ":ol_supply_w_id", -1,
(ub1 *) ol_supply_w_id, sizeof (int), SQLT_INT, -1,
(sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol2);
quit ();
exit (1);
}
if (obndrv (&curol2, (text *) ":ol_amount", -1, (ub1 *) ol_amount,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol2);
quit ();
exit (1);
}
if (obndrv (&curol2, (text *) ":ol_dist_info", -1, (ub1 *) ol_dist_info,
24, SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curol2);
quit ();
exit (1);
}
/* orders (delivered) */
if (obndrv (&curo1, (text *) ":o_id", -1, (ub1 *) o_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo1);
quit ();
exit (1);
}
if (obndrv (&curo1, (text *) ":o_d_id", -1, (ub1 *) o_d_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo1);
quit ();
exit (1);
}
if (obndrv (&curo1, (text *) ":o_w_id", -1, (ub1 *) o_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo1);
quit ();
exit (1);
}
if (obndrv (&curo1, (text *) ":o_c_id", -1, (ub1 *) o_c_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo1);
quit ();
exit (1);
}
if (obndrv (&curo1, (text *) ":o_carrier_id", -1, (ub1 *) o_carrier_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo1);
quit ();
exit (1);
}
if (obndrv (&curo1, (text *) ":o_ol_cnt", -1, (ub1 *) o_ol_cnt,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo1);
quit ();
exit (1);
}
/* orders (not delivered) */
if (obndrv (&curo2, (text *) ":o_id", -1, (ub1 *) o_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo2);
quit ();
exit (1);
}
if (obndrv (&curo2, (text *) ":o_d_id", -1, (ub1 *) o_d_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo2);
quit ();
exit (1);
}
if (obndrv (&curo2, (text *) ":o_w_id", -1, (ub1 *) o_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo2);
quit ();
exit (1);
}
if (obndrv (&curo2, (text *) ":o_c_id", -1, (ub1 *) o_c_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo2);
quit ();
exit (1);
}
if (obndrv (&curo2, (text *) ":o_ol_cnt", -1, (ub1 *) o_ol_cnt,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curo2);
quit ();
exit (1);
}
/* new order */
if (obndrv (&curno, (text *) ":no_o_id", -1, (ub1 *) no_o_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curno);
quit ();
exit (1);
}
if (obndrv (&curno, (text *) ":no_d_id", -1, (ub1 *) no_d_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curno);
quit ();
exit (1);
}
if (obndrv (&curno, (text *) ":no_w_id", -1, (ub1 *) no_w_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &curno);
quit ();
exit (1);
}
}
/*---------------------------------------------------------------------+
| Initialize random number generator |
+---------------------------------------------------------------------*/
srand (getpid ());
srand48 (getpid ());
initperm ();
/*---------------------------------------------------------------------+
| Load the WAREHOUSE table. |
+---------------------------------------------------------------------*/
if (do_A || do_w) {
nrows = eware - bware + 1;
fprintf (stderr, "Loading/generating warehouse: w%d - w%d (%d rows)\n",
bware, eware, nrows);
begin_time = gettime ();
begin_cpu = getcpu ();
for (loop = bware; loop <= eware; loop++) {
w_tax = (rand () % 2001);
randstr (w_name, 6, 10);
randstr (w_street_1, 10, 20);
randstr (w_street_2, 10, 20);
randstr (w_city, 10, 20);
randstr (str2, 2, 2);
randnum (num9, 9);
num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
if (gen) {
printf ("%d 30000000 %d %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 %d 30000.0 3001\n",
i + 1, dwid, d_name[i], d_street_1[i], d_street_2[i],
d_city[i], str2, num9, d_tax[i]); */
/* Reordered columns */
printf ("%d %d 3000000 %d 3001 %s %s %s %s %s %s\n",
i + 1, dwid, d_tax[i], d_name[i], d_street_1[i],
d_street_2[i], d_city[i], str2, num9 );
}
else {
d_id[i] = i + 1;
d_w_id[i] = dwid;
strncpy (d_state[i], str2, 2);
strncpy (d_zip[i], num9, 9);
}
}
if (gen) {
fflush (stdout);
}
else {
if (oexn (&curd, DISTARR, 0)) {
errrpt (&tpclda, &curd);
```

```
orol (&tpclda);
fprintf (stderr, "Aborted at warehouse %d, district 1\n", dwid);
quit ();
exit (1);
}
else if (ocom (&tpclda)) {
errrpt (&tpclda, &tpclda);
orol (&tpclda);
fprintf (stderr, "Aborted at warehouse %d, district 1\n", dwid);
quit ();
exit (1);
}
}
}
end_time = gettime ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
nrows, end_time - begin_time, end_cpu - begin_cpu);
}
/*---------------------------------------------------------------+
| Load the CUSTOMER table. |
+---------------------------------------------------------------*/
if (do_A || do_c) {
nrows = (eware - bware + 1) * CUSTFAC * DISTFAC;
fprintf (stderr, "Loading/generating customer: w%d - w%d (%d rows)\n ",
bware, eware, nrows);
begin_time = gettime ();
begin_cpu = getcpu ();
cid = 0;
cdid = 1;
cwid = bware;
loopcount = 0;
for (row = 0; row < nrows; ) {
for (i = 0; i < CUSTARR; i++, row++) {
cid++;
if (cid > CUSTFAC) { /* cycle cust id */
cid = 1; /* cheap mod */
cdid++; /* shift district cycle */
if (cdid > DISTFAC) {
cdid = 1;
cwid++; /* shift warehouse cycle */
}
}
c_id[i] = cid;
c_d_id[i] = cdid;
c_w_id[i] = cwid;
if (cid <= 1000)
randlastname (c_last[i], cid - 1);
else
randlastname (c_last[i], NURand (255, 0, 999, CNUM1));
c_credit[i][1] = 'C';
if (rand () % 10)
c_credit[i][0] = 'G';
else
c_credit[i][0] = 'B';
c_discount[i] = (rand () % 5001);
randstr (c_first[i], 8, 16);
randstr (c_street_1[i], 10, 20);
randstr (c_street_2[i], 10, 20);
randstr (c_city[i], 10, 20);
randstr (str2, 2, 2);
randnum (num9, 9);
num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
randnum (num16, 16);
randstr (c_data[i], 300, 500);
if (gen) {
printf ("%d %d %d %s OE %s %s %s %s %s %s %s %cC 5000000 %d -1000 1000
1 0 %s\n",
cid, cdid, cwid, c_first[i], c_last[i],
c_street_1[i], c_street_2[i], c_city[i], str2, num9,
num16, sdate, c_credit[i][0], c_discount[i], c_data[i]);
}
else {
strncpy (c_state[i], str2, 2);
strncpy (c_zip[i], num9, 9);
strncpy (c_phone[i], num16, 16);
}
}
if (gen) {
fflush (stdout);
}
else {
if (oexn (&curc, CUSTARR, 0)) {
errrpt (&tpclda, &curc);
orol (&tpclda);
fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
c_w_id[0], c_d_id[0], c_id[0]);
quit ();
exit (1);
}
else if (ocom (&tpclda)) {
errrpt (&tpclda, &tpclda);
orol (&tpclda);
fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
c_w_id[0], c_d_id[0], c_id[0]);
quit ();
exit (1);
}
}
if ((++loopcount) % 50)
fprintf (stderr, ".");
else
fprintf (stderr, " %d rows committed\n ", row);
}
end_time = gettime ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
nrows, end_time - begin_time, end_cpu - begin_cpu);
}
/*---------------------------------------------------------------+
| Load the ITEM table. |
+---------------------------------------------------------------*/
if (do_A || do_i) {
nrows = ITEMFAC;
fprintf (stderr, "Loading/generating item: (%d rows)\n ", nrows);
begin_time = gettime ();
begin_cpu = getcpu ();
loopcount = 0;
for (row = 0; row < nrows; ) {
for (i = 0; i < ITEMARR; i++, row++) {
i_im_id[i] = (rand () % 10000) + 1;
i_price[i] = ((rand () % 9901) + 100);
randstr (i_name[i], 14, 24);
randdatastr (i_data[i], 26, 50);
if (gen) {
printf ("%d %d %s %d %s\n", row + 1, i_im_id[i], i_name[i],
i_price[i], i_data[i]);
}
else {
i_id[i] = row + 1;
}
}
if (gen) {
fflush (stdout);
}
else {
if (oexn (&curi, ITEMARR, 0)) {
errrpt (&tpclda, &curi);
orol (&tpclda);
fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
quit ();
exit (1);
}
else if (ocom (&tpclda)) {
errrpt (&tpclda, &tpclda);
orol (&tpclda);
fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
quit ();
exit (1);
}
}
if ((++loopcount) % 50)
fprintf (stderr, ".");
else
fprintf (stderr, " %d rows committed\n ", row);
}
end_time = gettime ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
nrows, end_time - begin_time, end_cpu - begin_cpu);
}
/*---------------------------------------------------------------+
| Load the STOCK table. |
+---------------------------------------------------------------*/
if (do_A || do_s) {
nrows = (eware - bware + 1) * STOCFAC;
fprintf (stderr, "Loading/generating stock: w%d - w%d (%d rows)\n ",
bware, eware, nrows);
begin_time = gettime ();
begin_cpu = getcpu ();
sid = 0;
swid = bware;
loopcount = 0;
for (row = 0; row < nrows; ) {
for (i = 0; i < STOCARR; i++, row++) {
if (++sid > STOCFAC) { /* cheap mod */
sid = 1;
swid++;
}
s_quantity[i] = (rand () % 91) + 10;
randstr (str24[0], 24, 24);
randstr (str24[1], 24, 24);
randstr (str24[2], 24, 24);
randstr (str24[3], 24, 24);
randstr (str24[4], 24, 24);
randstr (str24[5], 24, 24);
randstr (str24[6], 24, 24);
randstr (str24[7], 24, 24);
randstr (str24[8], 24, 24);
randstr (str24[9], 24, 24);
randdatastr (s_data[i], 26, 50);
if (gen) {
printf ("%d %d %d %s %s %s %s %s %s %s %s 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 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 9/24 Saar */
printf ("%d %d %d %d %s 11 %d 1\n", cid, cdid, cwid,
randperm3000[cid - 1], sdate, o_ol_cnt[i]);
}
}
else {
o_id[i] = cid;
o_d_id[i] = cdid;
o_w_id[i] = cwid;
o_c_id[i] = randperm3000[cid - 1];
}
for (j = 0; j < o_ol_cnt[i]; j++) {
ol_i_id[j] = sid = lrand48 () % 100000 + 1;
if (cid < 2101)
ol_amount[j] = 0;
else
ol_amount[j] = (lrand48 () % 999999 + 1) ;
```

```
randstr (str24[j], 24, 24);
if (gen) {
if (cid < 2101) {
fprintf (olfp, "%d %d %d %d %s %d %d 5 %ld %s\n", cid,
cdid, cwid, j + 1, sdate, ol_i_id[j], cwid,
ol_amount[j], str24[j]);
}
else {
/* change from null date, reorg columns 9/24 Saar */
fprintf (olfp, "%d %d %d %d 01-Jan-1811 %d %d 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);
}
```

Appendix D: RTE Scripts

D.1 RTE Parameters

```
/* values of 65 and 119 and excluding the value of 96 and 112 */
LASTC=86
MEASUREMENT="1"
MASTER lmaster
SUT = "raven"
SLAVES driver1,driver2,driver3,driver4,driver5,driver6, driver7,
driver8,driver9,driver10,driver11, driver12,driver13, driver14,driver15,driver16
CLIENT_REAL = "client1 client2 client3 client4 client5 client6 client7 client8"
CLIENT client1x oracle oracle
CLIENT client1y oracle oracle
CLIENT client2x oracle oracle
CLIENT client2y oracle oracle
CLIENT client3x oracle oracle
CLIENT client3y oracle oracle
CLIENT client4x oracle oracle
CLIENT client4y oracle oracle
CLIENT client5x oracle oracle
CLIENT client5y oracle oracle
CLIENT client6x oracle oracle
CLIENT client6y oracle oracle
CLIENT client7x oracle oracle
CLIENT client7y oracle oracle
CLIENT client8x oracle oracle
CLIENT client8y oracle oracle
TELNET telnet 23
SOCKET socket 199703
SOCKET_NETWORK socket1 6700 driver3
SOCKET_NETWORK socket2 6701 driver4
SOCKET_NETWORK socket3 6702 driver3
SOCKET_NETWORK socket4 6703 driver4
SOCKET_NETWORK socket5 6704 driver3
SOCKET_NETWORK socket6 6705 driver4
SOCKET_NETWORK socket7 6706 driver3
SOCKET_NETWORK socket8 6707 driver4
SOCKET_NETWORK socket9 6708 driver3
SOCKET_NETWORK socket10 6709 driver4
SOCKET_NETWORK socket11 6700 driver5
SOCKET_NETWORK socket12 6701 driver6
SOCKET_NETWORK socket13 6702 driver5
SOCKET_NETWORK socket14 6703 driver6
SOCKET_NETWORK socket15 6704 driver5
SOCKET_NETWORK socket16 6705 driver6
SOCKET_NETWORK socket17 6706 driver5
SOCKET_NETWORK socket18 6707 driver6
SOCKET_NETWORK socket19 6708 driver5
SOCKET_NETWORK socket20 6709 driver6
SOCKET_NETWORK socket21 6700 driver7
SOCKET_NETWORK socket22 6701 driver8
SOCKET_NETWORK socket23 6702 driver7
SOCKET_NETWORK socket24 6703 driver8
SOCKET_NETWORK socket25 6704 driver7
SOCKET_NETWORK socket26 6705 driver8
SOCKET_NETWORK socket27 6706 driver7
SOCKET_NETWORK socket28 6707 driver8
SOCKET_NETWORK socket29 6708 driver7
SOCKET_NETWORK socket30 6709 driver8
SOCKET_NETWORK socket31 6700 driver9
SOCKET_NETWORK socket32 6701 driver10
SOCKET_NETWORK socket33 6702 driver9
SOCKET_NETWORK socket34 6703 driver10
SOCKET_NETWORK socket35 6704 driver9
SOCKET_NETWORK socket36 6705 driver10
SOCKET_NETWORK socket37 6706 driver9
SOCKET_NETWORK socket38 6707 driver10
SOCKET_NETWORK socket39 6708 driver9
SOCKET_NETWORK socket40 6709 driver10
SOCKET_NETWORK socket41 6700 driver11
SOCKET_NETWORK socket42 6701 driver12
SOCKET_NETWORK socket43 6702 driver11
SOCKET_NETWORK socket44 6703 driver12
SOCKET_NETWORK socket45 6704 driver11
SOCKET_NETWORK socket46 6705 driver12
SOCKET_NETWORK socket47 6706 driver11
SOCKET_NETWORK socket48 6707 driver12
SOCKET_NETWORK socket49 6708 driver11
SOCKET_NETWORK socket50 6709 driver12
SOCKET_NETWORK socket51 6700 driver13
SOCKET_NETWORK socket52 6701 driver14
SOCKET_NETWORK socket53 6702 driver13
SOCKET_NETWORK socket54 6703 driver14
SOCKET_NETWORK socket55 6704 driver13
SOCKET_NETWORK socket56 6705 driver14
SOCKET_NETWORK socket57 6706 driver13
SOCKET_NETWORK socket58 6707 driver14
SOCKET_NETWORK socket59 6708 driver13
SOCKET_NETWORK socket60 6709 driver14
SOCKET_NETWORK socket61 6700 driver15
SOCKET_NETWORK socket62 6701 driver16
SOCKET_NETWORK socket63 6702 driver15
SOCKET_NETWORK socket64 6703 driver16
SOCKET_NETWORK socket65 6704 driver15
SOCKET_NETWORK socket66 6705 driver16
SOCKET_NETWORK socket67 6706 driver15
SOCKET_NETWORK socket68 6707 driver16
SOCKET_NETWORK socket69 6708 driver15
SOCKET_NETWORK socket70 6709 driver16
SOCKET_NETWORK socket71 6700 driver1
SOCKET_NETWORK socket72 6701 driver2
SOCKET_NETWORK socket73 6702 driver1
SOCKET_NETWORK socket74 6703 driver2
SOCKET_NETWORK socket75 6704 driver1
SOCKET_NETWORK socket76 6705 driver2
SOCKET_NETWORK socket77 6706 driver1
SOCKET_NETWORK socket78 6707 driver2
SOCKET_NETWORK socket79 6708 driver1
SOCKET_NETWORK socket80 6709 driver2
OUTPUTNAME="../runs/raven"
CPU=12
BEGIN_WAIT=10:00
RAMPUP=43:00
RUNTIME=30:00
RAMPDOWN=10:00
RAMPDOWN_WAIT=5:00
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 = "12.1, 0, 0"
PAYMENT = "12.1, 0, 0, 43.15, 43.15, 43.15 "
ORDSTAT = "10.1, 0, 0, 4.05, 4.05, 4.05 "
DELIVERY = "05.1, 0, 0, 4.05, 4.05, 4.05 "
STOCKLEV = "05.1, 0, 0, 4.05, 4.05, 4.05 "
START client1x socket71 195
START client1y socket72 195
START client1x socket73 195
START client1y socket74 195
START client1x socket75 195
START client1y socket76 195
#if 1
START client1x socket77 195
START client1y socket78 195
START client1x socket79 195
START client1y socket80 196
#endif
START client2x socket1 195
START client2y socket2 195
START client2x socket3 195
START client2y socket4 195
START client2x socket5 195
START client2y socket6 195
#if 1
START client2x socket7 195
START client2y socket8 195
START client2x socket9 195
START client2y socket10 196
#endif
START client3x socket11 195
START client3y socket12 195
START client3x socket13 195
START client3y socket14 195
START client3x socket15 195
START client3y socket16 195
#if 1
START client3x socket17 195
START client3y socket18 195
START client3x socket19 195
START client3y socket20 196
#endif
START client4x socket21 195
START client4y socket22 195
START client4x socket23 195
START client4y socket24 195
START client4x socket25 195
START client4y socket26 195
#if 1
START client4x socket27 195
START client4y socket28 195
START client4x socket29 195
START client4y socket30 196
#endif
START client5x socket31 195
START client5y socket32 195
START client5x socket33 195
START client5y socket34 195
START client5x socket35 195
START client5y socket36 195
#if 1
START client5x socket37 195
START client5y socket38 195
START client5x socket39 195
START client5y socket40 196
#endif
START client6x socket41 195
START client6y socket42 195
START client6x socket43 195
START client6y socket44 195
START client6x socket45 195
START client6y socket46 195
#if 1
START client6x socket47 195
START client6y socket48 195
START client6x socket49 195
START client6y socket50 196
#endif
START client7x socket51 195
START client7y socket52 195
START client7x socket53 195
START client7y socket54 195
START client7x socket55 195
START client7y socket56 195
#if 1
START client7x socket57 195
START client7y socket58 195
START client7x socket59 195
START client7y socket60 197
```

```
#endif
START client8x socket61 195
START client8y socket62 195
START client8x socket63 195
START client8y socket64 195
START client8x socket65 195
START client8y socket66 195
#if 1
START client8x socket67 195
START client8y socket68 195
START client8x socket69 195
START client8y socket70 197
#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: Quotations

ORACLE CORPORATION

March 23, 1998

To whom it may concern:

The following describes the Oracle software pricing and support for the TPC-C Benchmarks on a Bull Escala RL 470 Server.

Server	Product	Unit Price	Extended Price	5 year support	Total
Escala RL 470	Oracle 8.0 and SQL*NET	222721	222721	222721	445442

SQL*Net is bundled with Oracle 8.0 and not priced separately.

The Oracle software pricing is based on the enterprise server pricing for the Oracle product(s) on the platform as configured.

Support Pricing

The support pricing is based on the Oracle Bronze Customer Care. Oracle Bronze Customer Care includes the following:

Real Time Telephone Assistance between 5:AM and 6:00 PM PST.
Product Updates.
Online Access to the Real Time Support System.
Online Access to Oracle Worldwide Support's Mail Server.
Subscription to Oracle Worldwide Support's newsletter.
Access to Oracle Worldwide Support's private forum.

The prices quoted here are good for 60 days.

Youlun Pang
200 Oracle Parkway
Box 659204
Redwood Shores, CA 94065

CDW COMPUTER DISCOUNT WAREHOUSE
The right price. The right advice.
800-800-4239 www.cdw.com
200 N. Milwaukee Ave. Vernon Hills, IL 60061
phone 847.465.6000 fax 847.465.6800

SALES QUOTATION
3-04-98 5:44 PM

ORDER NO.
7599936

BILL TO: 3003989

BULL FA
1 RUE DE PROVENCE

ESHIROLLES ,FR 38130
PHONE# 33476297638

SHIP TO:

BULL FA
1 RUE DE PROVENCE

ESHIROLLES ,FR 38130
CUST PO# MONTH QUOTE

QUOTE DATE	SHIPPED VIA	TERMS	SLS	RESALE NO.
3-04-98	IE	Master Card / VIS	KEEGAN MCG	

EDC CD	QTY ORD	ITEM NUMBER/ DESCRIPTION	UNIT	UNIT PRICE	EXTENDED PRICE
075530	863	LNK-EW20HUB LINKSYS 20PORT 10BT 1BNC ENET HUB	EA	119.00	102,697.00

SUB-TOTAL	102,697.00
SHIPPING	[illegible]
SALES TAX	[illegible]
TOTAL DUE IN U.S DOLLARS	[illegible]2,697.[illegible]