ORACLE®

TPC Benchmark™ C
Full Disclosure Report

Sun Enterprise™ 6500 Cluster
Using Oracle8i™ RDBMS

Submitted for Review

September 24, 1999
Compliant with Revision 3.4 of the TPC-C specification

TPC Benchmark C Full Disclosure Report

First Printing

901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A.

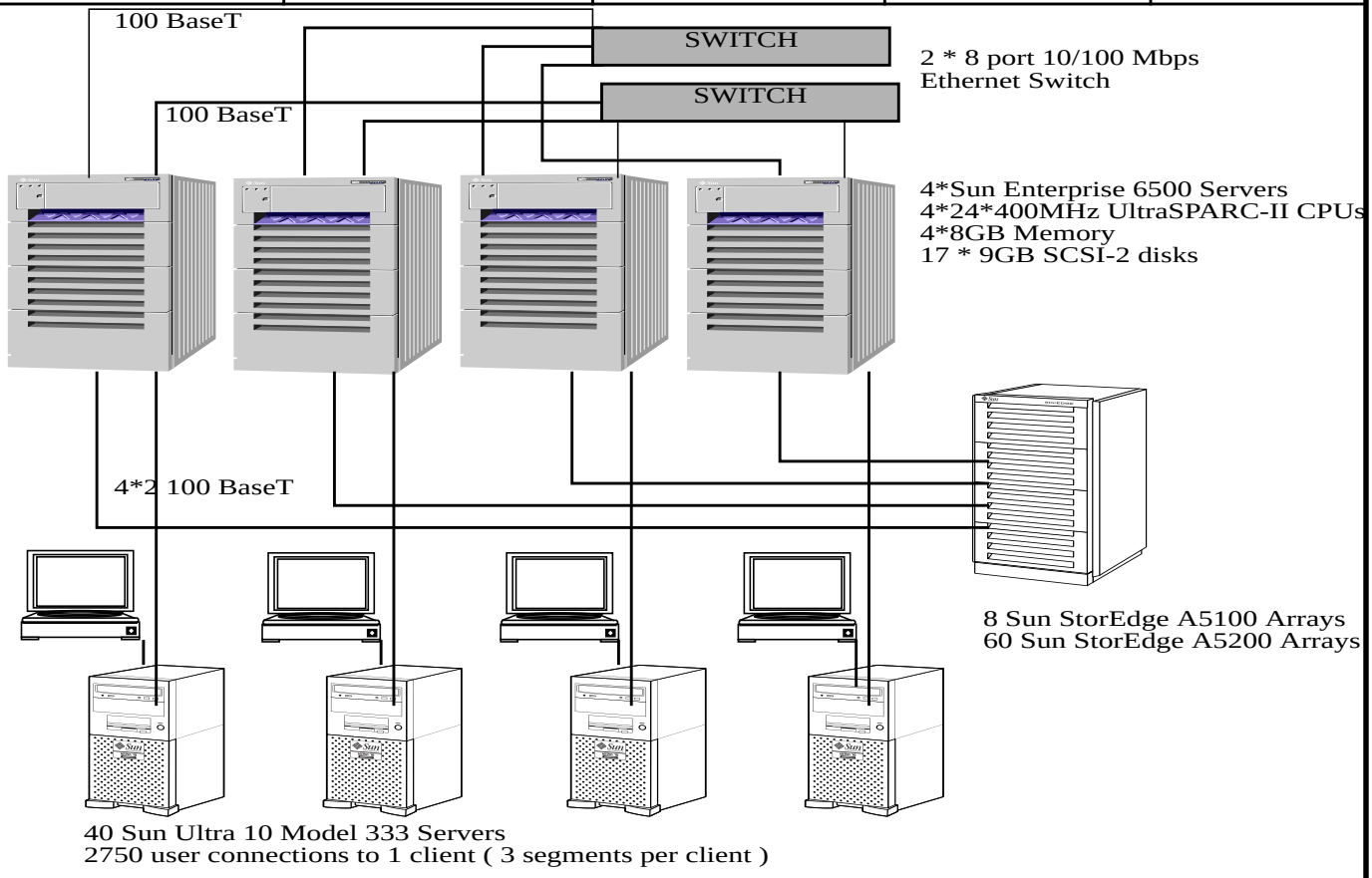
TRADEMARKS

Sun Enterprise 6500 Cluster using Oracle8i

TPC-C 3.4

Report Date:
September 24, 1999

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date	
$13,153,324	**135,461.40 tpmC**	**$97.10 per tpmC**	**January 31, 2000**	
Processors	Database Manager	Operating System	Other Software	Number of Users
4*24 * 400MHz UltraSPARC II	**Oracle8i v 8.1.6**	**Solaris 2.6**	**Sun Cluster 2.2 BEA Tuxedo 6.3**	**110,000**



Configuration

	Server System	Front End Systems
Database Nodes:	4 Sun Enterprise 6500 Servers	40 * Ultra 10 Model 333
Processors	4*24* 400 MHz UltraSPARC II	1 * 333 MHz UltraSPARC II each
Cache memory	32KB (D+I), 8MB external	32KB (D+I), 512KB external, each
Main memory	4*8 GB	1 GB each
Disk controllers	4 * 16 dualport FC-AL, 4 *4 Fast/Wide SCSI-2	1
Disk Drives	1320 * 9GB FC-AL, 112 * 18GB FC-AL, 17 * 9GB SCSI-2	1 * 9 GB disk
Total Disk Storage	14,050 GB	9 GB each
10 BaseT Hub	None	13760 * 9-Port Hubs
100 Base T Hubs	8 x 8-Port Hubs	None
100 BaseT Switches	2 * 8-Port	None

Sun

ORACLE®

Sun Enterprise 6500 Cluster using Oracle8i

TPC-C 3.4

Report Date: September 24, 1999

Pricing Summary

Description	Part Number	Source	Unit Price	Qty	Ext. Price	5 Yr. Maint.
Server Hardware						
Enterprise 6500 Server Base	E6501	2	86,400	4	345,600	540,518
CPU/Memory board	2602A	2	4,320	48	207,360	487,526
400MHz/8MB UltraSPARC II	2580A	2	10,800	96	1,036,800	
1GB memory (8 * 128MB)	7023A	2	6,840	32	218,880	
PS/300W for Ex000	954A	2	1,296	24	31,104	
SBus I/O board	2612A	2	4,680	16	74,880	
FCAL 100MB/S SBus Host Adapter	6730A	2	1,944	48	93,312	
FCAL GBIC Module	6731A	2	432	188	81,216	
200GB StorEdge A5200 Array	SG-ARY522A-200GR4	2	68,040	16	1,088,640	225,792
200GB StorEdge A5200 Array	SG-ARY520A-200G	2	68,040	44	2,993,760	620,928
252GB StorEdge A5100 Array	SG-ARY530A-254G	2	45,288	8	362,304	112,896
FC-AL cables	978A	2	978	72	70,416	
9GB disk Unipack	SG-XDSK010A-9G	2	1,053	17	17,901	
Cluster Terminal Concentrator	X1312A	2	1,800	1	1,800	
12-24GB 4mm DDS-3 Tape Drive	6283A	2	972	4	3,888	
8 port 10/100 Mbps Ethernet Switch	NX-SW8	5	219	4	876	
Server Hardware Subtotal					**6,628,737**	**1,987,661**
Server Software						
Solaris Server Software	SOLMS-26ZW9999	1	100	1	100	
SPARC Compiler C/C++ 5.0	WCCIS-500-T999	1	995	1	995	1,080
Sun Cluster 2.2 license for E6500	CL7IS-229-9999	1	24,000	4	96,000	139,680
Sun Cluster 2.2 OPS Agent for E6500	CLCIS-22P-9999	1	4,000	4	16,000	23,280
Oracle OPS8i		3	1,909,080	1	1,909,080	1,909,080
Server Software Subtotal					**2,022,175**	**2,073,120**
Client Hardware						
Ultra 10 Server Model 333	A22UHC1Z9S-B512CP	2	5,433	40	217,320	172,224
512MB Memory for Ultra 10	7039A	2	1,562	40	62,480	
PCI QEF Card	1034A	2	1,292	40	51,680	
Color Monitor	x7126	2	418	40	16,720	
Client Hardware Subtotal					**348,200**	**172,224**
Client Software						
BEA Tuxedo CFS 6.3		4	3,000	40	120,000	96,000
Client Software Subtotal					**120,000**	**96,000**
User Connectivity						
8-port 10/100Mbps Fast Ethernet Hub	NX-SOHODH8	5	129	10	1,290	
9-port 10Mbps Ethernet Hub	NX-H9EZ	5	30	15136	454,080	
User Connectivity Subtotal					**455,370**	
Sun Enterprise Services discounts						**(750,163)**
			Total		**9,574,482**	**3,578,842**
			5Yr. cost		**13,153,324**	
			tpmC Rating		**135,461**	
			$/tpmC		**$97.10**	

Service for all Sun products is from Sun Microsystems, Inc.

Notes:

1. Sun Microsystems Inc. 2. CAT Technology Inc 3. Oracle Corp 4. BEA Systems, Inc. 5. NETLUX

Audited by: Lorna Livingtree, Performance Metrics Inc.

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 purchase are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.

Sun ORACLE®	Sun Enterprise 6500 Cluster using Oracle8i	TPC-C 3.4 **Report Date: September 24, 1999**

Numerical Quantity Summary

MQTH, Computed Maximum Qualified Throughput **= 135,461.40 tpmC**

% throughput difference, reported & reproducibility runs = < 0.5%

Response Times (in secs)	90th Percentile	Average	Maximum
Menu	0.50	0.25	1.09
New-Order	1.00	0.47	13.53
Payment	0.60	0.34	17.43
Order-Status	0.40	0.34	5.18
Delivery(interactive)	0.74	0.30	1.50
Delivery(deferred)	2.00	0.28	8.00
Stock-level	5.00	3.24	12.24

Transaction Mix, in percent of total transactions

New-Order	44.75%
Payment	43.11%
Order-Status	4.04%
Delivery	4.05%
Stock-level	4.04%

Keying/Think Times (in secs)	Average.	Min.	Maximum
New-Order	18.02 / 12.09	18.01 / 0	18.06 / 120.80
Payment	3.02 / 12.08	3.01 / 0	3.04 / 120.80
Order-Status	2.02 / 10.08	2.01 / 0	2.04 / 100.80
Delivery	2.02 / 5.08	2.01 / 0	2.06 / 50.80
Stock-level	2.02 / 5.08	2.01 / 0	2.05 / 50.80

Test Duration	
Ramp-up time	29 minutes
Measurement Interval	30 minutes
Number of checkpoints	1
Checkpoint Interval	30 minutes
Number of transactions (all types) completed in measurement interval	9081376

Preface

This report documents the compliance of the Sun Microsystems TPC Benchmark ™C testing on the Sun Enterprise 6500 Cluster running Solaris 2.6, Oracle8i v 8.1.6 RDBMS and Tuxedo 6.3 with the *TPC Benchmark ™C Standard Revision 3.4*

Document Structure

The *TPC Benchmark ™C Full Disclosure Report* is organized as follows:

- The main body of the document lists each item in Clause 8 of the TPC Benchmark ™C Standard and explains how each specification is satisfied.
- Appendix A contains the application source code that implements the transactions and forms modules.
- Appendix B contains the database layout.
- Appendix C contains the code used to create and load the database.
- Appendix D contains the configuration information for the operating system, the RDBMS and Tuxedo.
- Appendix E contains the 180-day space calculations.
- Appendix F contains the code used to generate transactions and measure response times.
- Appendix G contains the screen layouts of all the forms.
- Appendix H contains the quotes.

Additional Copies

Additional copies of this report may be ordered from the administrator of the TPC:

Shanley P.R.
777 N First Street, Suite 600
San Jose, CA 95112-6311
(408) 295-8894
FAX (408) 295-2613

TPC Benchmark C Overview

The *TPC Benchmark ™C Standard Specification* requires test sponsors to publish, and make available to the public, a full disclosure report for the results to be considered compliant with the Standard. The required contents of the full disclosure report are specified in Clause 8.

This report is intended to satisfy the Standard's requirement for full disclosure. It documents the compliance of the benchmark tests reported in the *TPC Benchmark ™C* results for the Sun Enterprise 6500 Cluster running the Oracle8i v 8.1.6 RDBMS.

In the *Standard Specification*, the main headings in Clause 8 are keyed to the other clauses. The headings in this report use the same sequence, so that they correspond to the titles or subjects referred to in Clause 8.

Each section in this report begins with the text of the corresponding item from Clause 8 of the *Standard Specification*, printed in italic type. The plain type text that follows explains how the tests comply with the TPC Benchmark ™C requirement. In sections where Clause 8 requires extensive listings, the section refers to the appropriate appendix at the end of this report.

1. General Items

1.1 Application Code and Definition Statements

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 application source code that implements the transactions and forms modules.

1.2 Benchmark Sponsor

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

Sun Microsystems and Oracle Corporation are the sponsors of this TPC-C benchmark.

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:

- *Database Tuning Options*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and application configuration parameters*
- *Compilation and linkage options and run-time optimizations used to create/install applications, OS, and/or databases.*

This requirement can be satisfied by providing a full list of all parameters and options.

Appendix D contains the Solaris 2.6 and Oracle8i parameters used in this benchmark.

1.4 Configuration Diagrams

Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences.

Figure 1 on page 10 is a diagram of the benchmark configuration. Figure 2 on page 11 is a configuration diagram of the priced system.

1.4.1 Configuration Items for the Sun Enterprise 6500 Cluster

For both the benchmarked and priced configuration, the server machines were four Sun Enterprise 6500. The priced configuration consisted of the following:

- 24UltraSPARC II 400 MHz processors per node.
- 8GB of main memory per node.
- 16 dual-port FC-AL host adapters per node.

- 4 SCSI-2 host adapters per node.
- 8 Sun StorEdge Arrays Model A5100 (14 x 18GB FC-AL disks in one array).
- 60 Sun StorEdge Arrays Model A5200 (22 x 9GB FC-AL disks in one array).
- 17 * 9GB SCSI-2 disks (5 on 1 node and 4 on each other node.).
- 12GB Backup Tape Device per node.
- 2 100 BaseT networks for connecting to the client systems per node.
- 2 8-port 10/100 Mbps Ethernet Switch for private interconnects.

The benchmark configuration consisted of the following:

- 24UltraSPARC II 400 MHz processors per node.
- 8GB of main memory per node.
- 16 dual-port FC-AL host adapters per node.
- 4 SCSI-2 host adapters per node.
- 8 Sun StorEdge Arrays Model A5100 (14 x 18GB FC-AL disks in one array).
- 50 Sun StorEdge Arrays Model A5000 (14 x 9GB FC-AL disks in one array).
- 27 Sun StorEdgeArrays Model A5200 (22 x 9GB FC-AL disks in one array).
- 12GB Backup Tape Device per node.
- 2 100 BaseT networks for connecting to the client systems per node.
- 16 * 4.2GB SCSI-2 disks (4 on each node), 1 * 2.1GB SCSI-2 disk (1 on 1 node).
- 1 8-port 10/100 Mbps Ethernet Switch and 1 100BaseT hub for private interconnects.

For the priced configuration, the client machines were 40 Sun Ultra 10 Model 333's each containing:

- One UltraSPARC II 333 MHz processor.
- 32KB (D+I), 512KB external cache.
- 1GB of main memory.
- One controller.
- One internal 9GB disk.

In the benchmark configuration, 4 of the clients were configured with 4GB disk instead of the 9GB disk.

The benchmark configuration used a Remote Terminal Emulator (RTE) to emulate TPC-C user sessions. The driver systems were directly connected through ethernet to the Ultra 10 Model 333's which execute the database client sessions

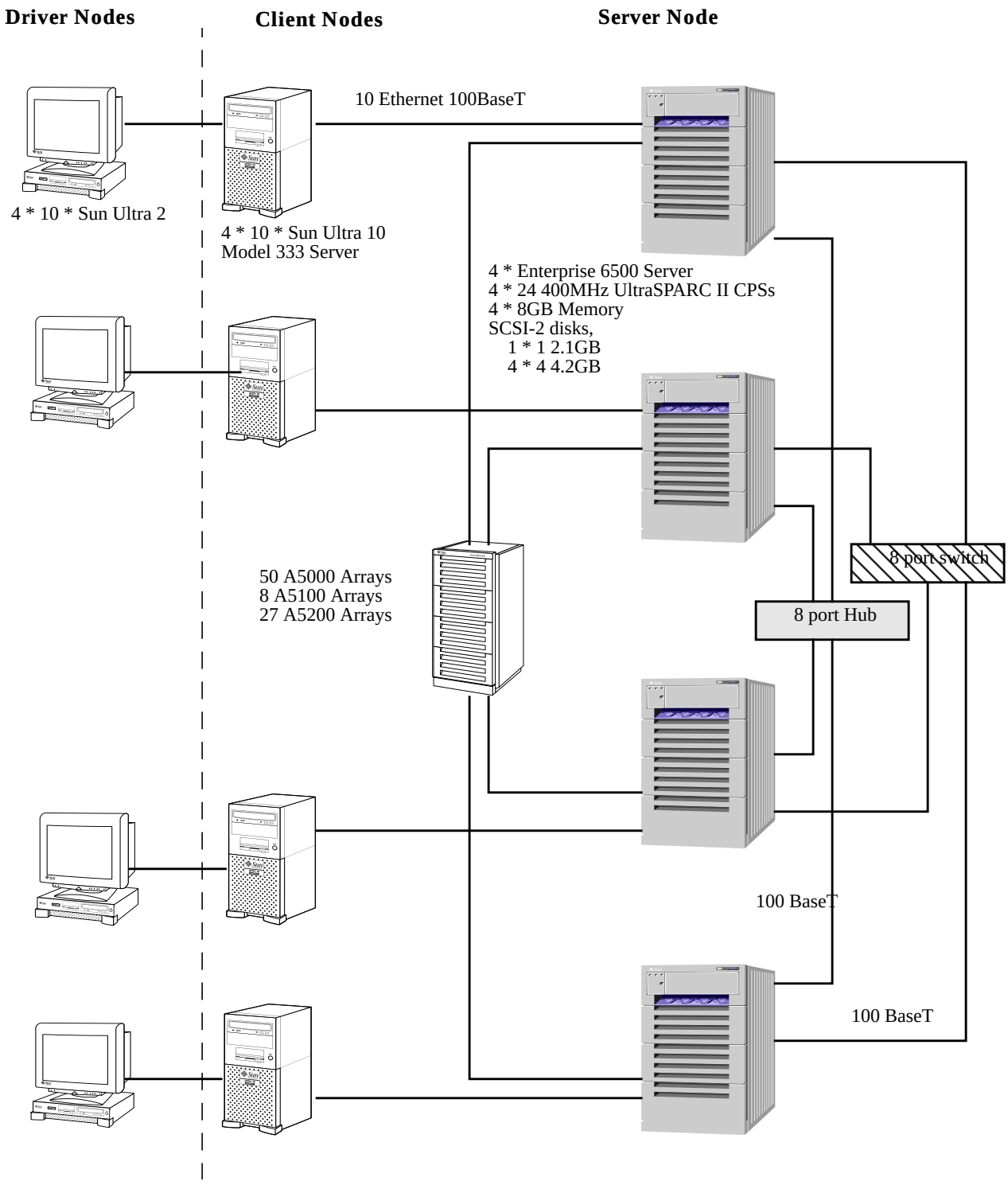


Figure 1 The Sun Enterprise 6500 Cluster Benchmark Configuration

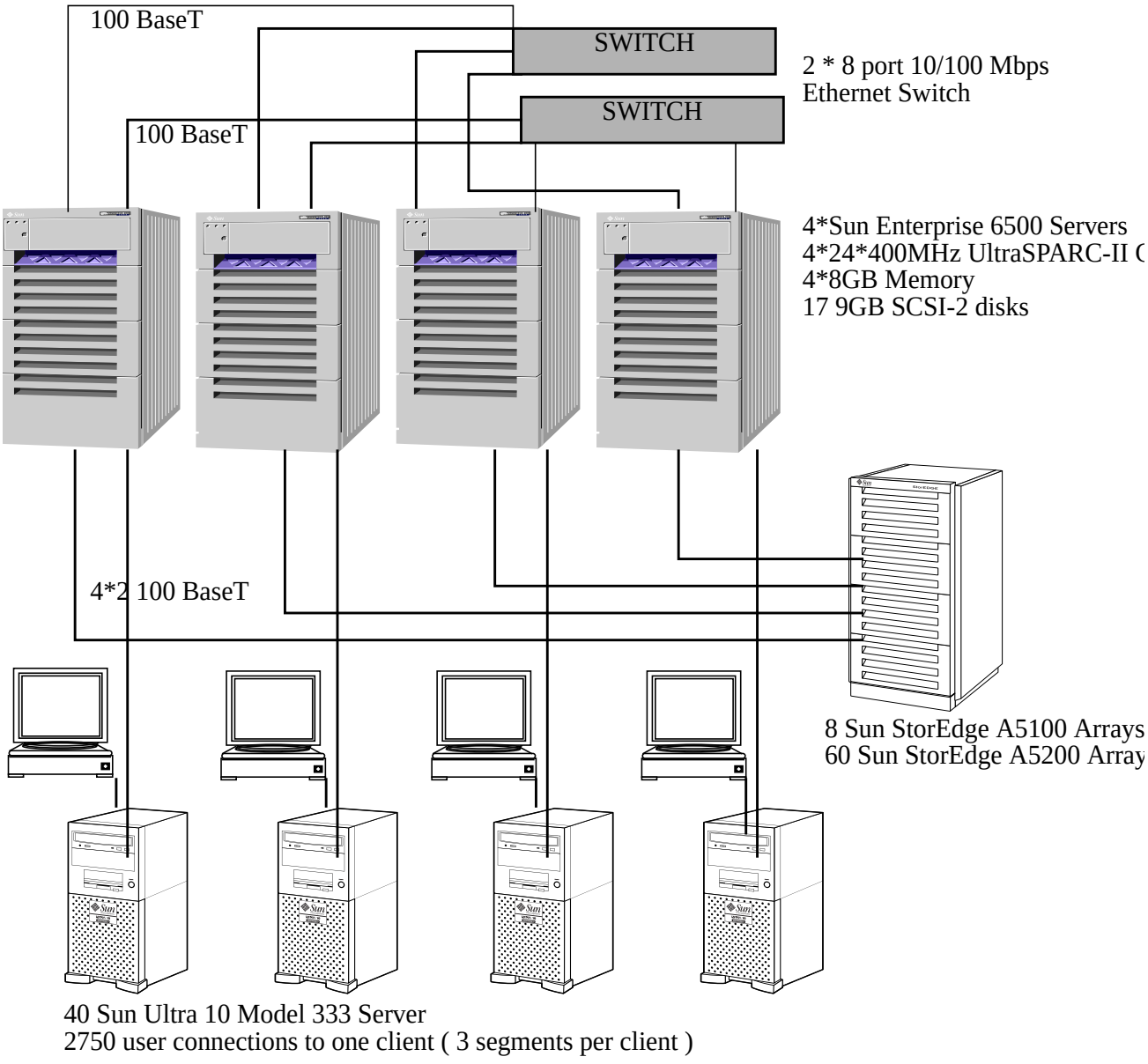


Figure 2 The Sun Enterprise 6500 Cluster Priced Configuration.

2. Clause 1 Related Items

2.1 Table Definitions

Listing must be provided for all table definition statements and all other statements used to set up the database.

Appendix B describes the programs that define, create, and populate an Oracle8 database for TPC-C testing.

2.2 Physical Organization of Database

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

Space was allocated to Oracle8 on the server according to the data in section 5.2.

2.3 Insert and 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 restrictions in the SUT database 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 delete operations to any tables beyond the limits defined in Clause 1.4.11.

2.4 Partitioning

While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark (see Clause 1.6), any such partitioning must be disclosed.

Horizontal partitioning was used for all tables except customer, item, stock and all indexes except clast_idx, icustomer, istock and i_idx. Refer to Appendix B for more details.

2.5 Table Replication

Replication of tables, if used, must be disclosed (see Clause 1.4.6).

No tables were replicated in this implementation.

2.6 Table Attributes

Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance (see Clause 1.4.7).

No additional or duplicate attributes were added to any of the tables.

3. Clause 2 Related Items

3.1 Random Number Generation

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

The Random Number Generator used was the one that appeared in the article titled “Random Number Generators: Good Ones Are Hard To Find” in the communications of the ACM - October 1988, Volume 31, Number 10. The properties of this random number generator are well-known and are documented in the article as producing a uniformly distributed pseudo-random sequence. To generate a random number, the driver programs first use a seed based on the host address, current time and the process-id of the respective session. This guarantees that each emulated user on all the RTE machines is mathematically independent of others.

3.2 Input/Output Screen Layouts

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

The screen layouts are shown in Appendix G.

3.3 Terminal Feature Verification

The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained.

The terminal features were verified by manually exercising each specification on a representative Sun Ultra 10 Model 333 server.

3.4 Presentation Manager or Intelligent Terminal

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

The TPC-C forms module was implemented using the capabilities of an xterm terminal emulator.

3.5 Transaction Statistics

Table 1 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 require.

TABLE 1. Transaction Statistics

Transaction Type	Statistics	Percentage
New Order	Home warehouse	99.00
	Remote warehouse	1.00
	Rolled-back transactions	1.00
	Average items per order	10.00

TABLE 1. Transaction Statistics

Transaction Type	Statistics	Percentage
Payment	Home warehouse	84.99
	Remote warehouse	15.01
	Non-primary key access	60.00
Order Status	Non-primary key access	60.00
Delivery	Skipped transactions	0.00
Transaction Mix	New order	44.75
	Payment	43.11
	Order status	4.04
	Delivery	4.05
	Stock level	4.04

3.6 Queuing Mechanism

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

Delivery transactions were submitted to servers using the same Tuxedo call mechanism that other transactions used. The only difference was that the call was asynchronous - i.e., control returned to the client process immediately and the deferred delivery completed asynchronously.

4. Clause 3 Related Items

4.1 Transaction System Properties (ACID)

The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosing which case was followed for the execution of Isolation Test 7.

The TPC Benchmark C Standard Specification defines a set of transaction processing system properties that a system under test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation, and Durability (ACID). This section defines each of these properties, describes the steps taken to ensure that they were present during the test and describes a series of tests done to demonstrate compliance with the standard.

4.2 Atomicity

The System under test must guarantee that 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.

4.2.1 Completed Transaction

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2.5.1.2) and verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have been changed appropriately.

The test was performed by first retrieving, through the sqlplus utility, the balances from a set of randomly picked warehouse, district, and customer rows (selected by customer number). Then a Payment transaction was submitted through the TPC Benchmark C application. Upon completion of the transaction, the balances of the selected warehouse, district, and customer rows were once again retrieved to verify that the changes had been made.

4.2.2 Aborted Transaction

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2.5.1.2) 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.

For this test, the same warehouse, district, and customer ids used above were used to issue a transaction to a modified version of the TPC Benchmark C application in which the COMMIT command had been replaced by a ROLLBACK command. After the transaction was aborted, the balances of the warehouse, district, and customer rows were retrieved to verify that no changes had been made to the database.

4.3 Consistency

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

The TPC Benchmark C standard requires System Under Test to meet the 12 consistency conditions listed in Clause 3.3.2.

The TPC Benchmark C Standard Specification requires explicit demonstration that the conditions are satisfied for the first four conditions only.

In order to demonstrate the consistency of the application, the following steps were taken:

1. Prior to the start of the benchmark run, the consistency of the database was verified by applying the consistency conditions 1-4 of Clause 3.2.2.
2. A fully-scaled run with a 30-minute steady state period was executed.
3. Upon the completion of the benchmark, the consistency of the database was determined by applying the same consistency conditions used in step 1.

4.4 Isolation

Isolation can be defined in terms of phenomena that can occur during the execution of concurrent transactions. These phenomena are P0 ("Dirty Write"), P1 ("Dirty Read"), P2 ("Non-repeatable Read") and P3 ("Phantom"). The table in Clause 3.4.1 of the TPC-C specifications defines the isolation requirements which must be met by the TPC-C transactions. Sufficient conditions must be enabled at either the system or application level to ensure the required isolation is maintained.

The TPC Benchmark C Standard Specification defines a set of required tests to be performed on the system under test to demonstrate that transaction isolation was present in the system configuration. These tests involve the execution of two transactions on the system and examining the interaction when the results of the transactions are committed to the database and when the results are aborted.

Because the database is shared between transaction processing nodes, each of the tests validating isolation between two transactions was performed on different nodes.

4.5 Isolation Test 1

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

1. At terminal 1, an Order-Status transaction was executed and the order returned was noted. The transaction was COMMITted.
2. At terminal 1, a New-Order transaction for the same customer used in step 1 was started but not COMMITted.
3. At terminal 2, an Order-Status transaction was started for the same customer used in step 1.
4. Terminal 2 transaction completed and was COMMITed without being blocked by terminal 1. The transaction returned the same order as returned by step 1.
5. COMMITted the transaction on terminal 1.
6. At terminal 2, an Order-Status transaction for the same customer used in step 1 was started, and it returned the order inserted by the New-order transaction in step 2.

4.6 Isolation Test 2

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

1. At terminal 1, an Order-Status transaction was executed and the order returned was noted. The transaction was COMMITted.
2. At terminal 1, a New-Order transaction was started for the same customer used in step 1, but not COMMITted.
3. At terminal 2, an Order-Status transaction was started for the same customer used at terminal 1.
4. The terminal 2 transaction completed and was COMMITed without being blocked by terminal 1. The transaction returned the same order as in step 1.
5. A ROLLBACK was executed for the transaction at terminal 1.
6. At terminal 2, an Order-status transaction for the same customer used in step 1 was started, and it returned the same order returned in step 1.

4.7 Isolation Test 3

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

1. At terminal 1, a New-Order transaction was started but not committed.
2. At terminal 2, another New-Order transaction was started for the same customer used at terminal 1.
3. The terminal 2 transaction waited for the terminal 1 transaction to complete.
4. COMMITted the transaction at terminal 1. The transaction at terminal 2 completed.
5. The order number for the terminal 2 transaction was one greater than the terminal 1 transaction. The Next Order Number for the district reflected the results from both transactions.

4.8 Isolation Test 4

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

1. At terminal 1, a New-Order transaction was started but not COMMlTted.
2. At terminal 2, another New-Order transaction was started for the same customer used at terminal 1.
3. The terminal 2 transaction waited for the terminal 1 transaction to complete.
4. A ROLLBACK was executed for the transaction on terminal 1. The transaction on terminal 2 completed.
5. The order number for the terminal 2 transaction was one greater than the previous transaction. The Next Order Number for the district reflected the results from only the terminal 2 transaction. In other words, it had been incremented by one.

4.9 Isolation Test 5

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

1. At terminal 1, a Delivery transaction was started but not COMMITted.
2. At terminal 2, a Payment transaction was started for the same customer used at terminal 1.

3. Terminal 2 transaction waited for the terminal 1 transaction to complete.
4. COMMITted the transaction at terminal 1. The transaction at terminal 2 completed.
5. The customer balance reflected the results from both transactions.

4.10 Isolation Test 6

This test demonstrates isolation for write-write conflicts of Payment and Delivery transaction when the Delivery transaction is ROLLED BACK.

1. At terminal 1, a Delivery transaction was started but not COMMITed.
2. At terminal 2, a Payment transaction was started for the same customer used on terminal 1.
3. Terminal 2 transaction waited for terminal 1 transaction to complete.
4. A ROLLBACK was executed for the transaction on terminal 1. The transaction on terminal 2 completed.
5. The customer balance reflected the result of the Payment transaction only.

4.11 Isolation Test 7

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

1. At terminal 1, New-Order transaction Tl was started and was queried for the price of two items, item x and item y. The transaction completed.
2. At terminal 2, New-Order transaction T2 was started for a group of items including item x twice and item y. The transaction was stopped immediately after retrieving the prices of all items. The prices of items x and y retrieved matched those retrieved in step 1.
3. At terminal 1, an interactive SQL transaction,T3, was started to increase the price of items x and y by ten percent. The transaction completed.
4. Continued terminal 2 transaction, The prices of all items were retrieved again. The prices of x and y matched those retrieved in step 1.
5. At terminal 1, another transaction was started to query the prices of items x and y. Completed the transaction.
6. The prices read by the transaction in step 5 matched those set by transaction T3.

4.12 Isolation Test 8

This test demonstrates isolation for phantom protection between New-Order and Order-Status transactions.

1. At terminal 1, an Order-Status transaction Tl was started for a randomly selected customer.
2. T1 was stopped immediately after reading the order table for the selected customer. The most recent order for that customer was retrieved.

3. At terminal 2, a New-Order transaction T2 was started for the same customer. T2 completed and was committed without being blocked by T1.
4. At terminal 1, T1 was resumed and the order table was read again to determine the most recent order for the same customer. The order retrieved was the same as that retrieved in step 2.
5. T1 completed and was committed.

4.13 Isolation Test 9

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

1. The NO_D_ID of all new_order rows for a randomly selected warehouse and district was changed to 11. The changes were committed.
2. At terminal 1, a delivery transaction T1 was started for the selected warehouse.
3. T1 was stopped immediately after reading the new-order table for the selected warehouse and district. No qualifying row was found.
4. At terminal 2, a New-Order transaction T2 was started for the same warehouse and district. T2 was completed and was committed without being blocked by T1.
5. At terminal 1, T1 was resumed and the new_order table was read again. No qualifying row was found. T1 was completed and committed.
6. The NO_D_ID of all new_order rows for the selected warehouse and district was restored to its original value.

4.14 Durability

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

List of single failures:

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

Instantaneous interruption (system crash/system hang) in processing which requires system reboot to recover.

Failure of all or part of memory (loss of contents)

Failure of the inter-node connection.

Sun Microsystems executed three durability tests to satisfy the durability requirements for this implementation of TPC Benchmark C. The combined test for loss of data, loss of log and failure of inter-node connection was performed with a 10 warehouse database and 100 terminals. To the best of our knowledge, a fully scaled test would also pass all durability requirements. The tests for loss of memory and instantaneous interruption on a single node and all four nodes were performed with a fully scaled database under the full load of terminals.

4.14.1 Permanent Irrecoverable Failure

Three failures - loss of primary interconnect, loss of a recovery log disk and loss of data disk - were induced in the order to demonstrate the durability in case of these failures. The following steps were taken:.

1. The database was loaded with 10 warehouses.
2. The database was archived to disk.
3. The D_NEXT_O_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).
4. A test was executed with 100 terminals. On the driver system, the committed and rolled back New-Order transactions were recorded in a “success” file.
5. After 5 minutes into the measurement period, the switch connecting the primary interconnects was powered off. The system detected the loss of primary interconnect, failed over to the secondary interconnect and continued to process transactions without interruption.
6. A count of users was taken.
7. After 8 minutes into measurement period,, one of the log disks was pulled off from the storage array. Since the log disk is mirrored, the system continued to process transactions without interruption despite the missing disk.
8. A count of users was taken.
9. After 11 minutes into the measurement period, one of the data disks was pulled off from the storage array.
10. The test was aborted on the driver.
11. The database was restored from the archive and recovered using the transaction log.
12. The contents of the “success” file on the driver and the ORDERS table were compared to verify that records in the “success” file for committed New-Order transaction had corresponding records in the ORDERS table.
13. Step 3 was repeated to determine the total number of orders (count2). Count2-count1 was the same as the number of committed records in the “success” file.
14. The contents of the “success” file on the driver and the ORDERS table were compared to verify that records in the “success” file for committed New-Order transaction had corresponding records in the ORDERS table.
15. Consistency condition 3 as specified in Clause 3.3.2.3 was verified at the end of the test.

4.14.2 Instantaneous Interruption and Loss of Memory on One Node

Instantaneous interruption and loss of memory tests were combined because the loss of power erases the contents of memory. This failure was induced by switching the power off on one of the nodes in the cluster while the benchmark was running. The following steps were taken:

1. The D_NEXT_O_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).

2. A fully-scaled test was executed. On the driver system, the committed and rolled back New-Order transactions were recorded in a “success” file.
3. After 10 minutes into the measurement period, one of the nodes in the cluster was powered off.
4. The system recognized the loss of a node and recofigured. Instance recovery was performed by the system on one of the other nodes.
5. Power was restored to the node and a normal system recovery was done. Meanwhile, the test was let to completion.
6. The contents of the “success” file on the driver and the ORDERS table were compared to verify that records in the “success” file for committed New-Order transactions had corresponding records in the ORDERS table.
7. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 was compared with the number of committed records in the “success” file.
8. Consistency condition 3 as specified in Clause 3.3.2.3 was verified at the end of the test.

4.14.3 Instantaneous Interruption and Loss of Memory on All Nodes.

Instantaneous interruption and loss of memory tests were combined because the loss of power erases the contents of memory. This failure was induced by switching the power off on all the nodes in the cluster while the benchmark was running. The following steps were taken:

1. The D_NEXT_O_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).
2. A fully-scaled test was executed. On the driver system, the committed and rolled back New-Order transactions were recorded in a “success” file.
3. After 10 minutes into the measurement period, all the nodes in the cluster were powered off.
4. The test was aborted on the driver.
5. Power was restored to all the nodes and a normal system recovery was done. A recovery was automatically performed by Oracle when the database was restarted and brought on-line.
6. The contents of the “success” file on the driver and the ORDERS table were compared to verify that records in the “success” file for committed New-Order transactions had corresponding records in the ORDERS table.
7. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 was compared with the number of committed records in the “success” file.
8. Consistency condition 3 as specified in Clause 3.3.2.3 was verified at the end of the test.

5. Clause 4 Related Items

5.1 Initial Cardinality of Tables

The Cardinality (e.g. number of rows) of each table, as it existed at the start of the benchmark run (see Clause 4.2), must be disclosed. If the database was over-scaled and inactive rows of the WAREHOUSE table were deleted (see Clause 4.2.2) the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed.

This database was built with 11000 warehouses. The number of active warehouses were the same.

TABLE 2. Cardinality of Tables

Table	Occurrences
Warehouse	11,000
District	110,000
Customer	330,000,000
History	330,000,000
Orders	330,000,000
New order	99,000,000
Order line	3,299,801,912
Stock	1,100,000,000
Item	100,000

5.2 Database Layout

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

Database layout is described in Appendix C.

5.3 Type of Database

A statement must be provided that describes:

1. The data model implemented by the DBMS used (e.g., relational, network hierarchical).

2. The database interface (e.g., embedded, call level) and access language (e.g., SQL, DL/1, COBOL read/write) used to implement the TPC-C transactions. If more than one interface/access language is used to implement TPC-C, each interface/access language must be described and a list of which interface/access language is used with which transaction type must be disclosed.

Oracle8i is a relational database management system.The interface we used was PL*SQL and OCI.

5.4 Mapping of Database

The mapping of database partitions/replications must be explicitly described.

No replication was done. Horizontal partitioning was used for all tables except customer, item, stock and all indexes except clast_idx, icustomer, istock and i_idx. Refer to table/index creation statements in Appendix B for more details.

5.5 180 Day Space Computation

Details of the 180 day space computations along with proof that the database is configured to sustain 8 hours of growth for the dynamic tables (Order, Order-Line, and History) must be disclosed (see Clause 4.2.3).

The 180 day space computation is shown in Appendix E.

6. Clause 5 Related Items

6.1 Measured tpmC

Measured tpmC must be reported.

The measured tpmC was 135,461.40.

6.2 Response Times

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

TABLE 3. Response Times

Type	Average	Maximum	90% percentile
New-Order	0.47	13.53	1.00
Payment	0.34	17.43	0.60
Order-Status	0.34	5.18	0.40
Interactive Delivery	0.30	1.50	0.74
Deferred Delivery	0.28	8.00	2.00
Stock-Level	3.24	12.24	5.00
Menu	0.25	1.09	0.50

6.3 Keying and Think Times

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

TABLE 4. Keying Times

Type	Average	Maximum	Minimum
New-Order	18.02	18.06	18.01
Payment	3.02	3.04	3.01
Order-Status	2.02	2.04	2.01
Interactive Delivery	2.02	2.06	2.01
Stock-Level	2.02	2.05	2.01

TABLE 5. Think Times

Type	Average	Maximum	Minimum
New-Order	12.09	120.80	0.00
Payment	12.08	120.80	0.00
Order-Status	10.08	100.80	0.00
Interactive Delivery	5.08	50.80	0.00
Stock-Level	5.08	50.80	0.00

6.4 Response Time Frequency Distribution Curves

Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type.

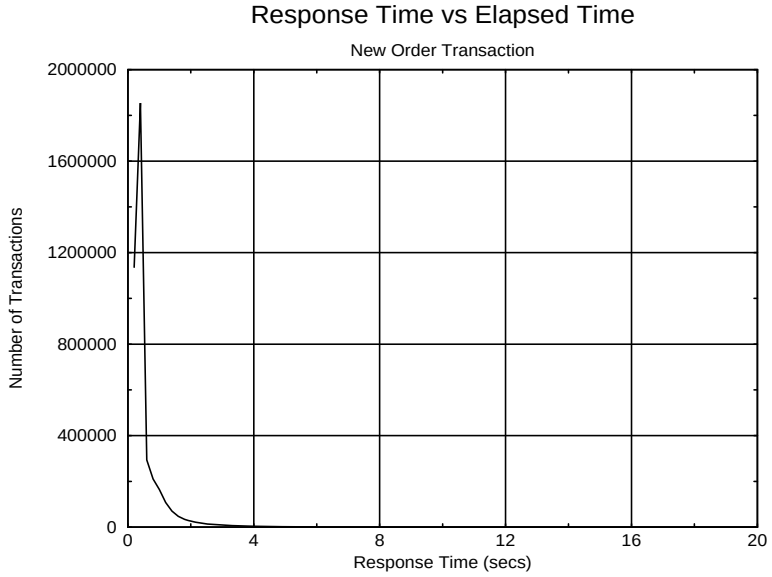


Figure 3 New Order Response Time Distribution

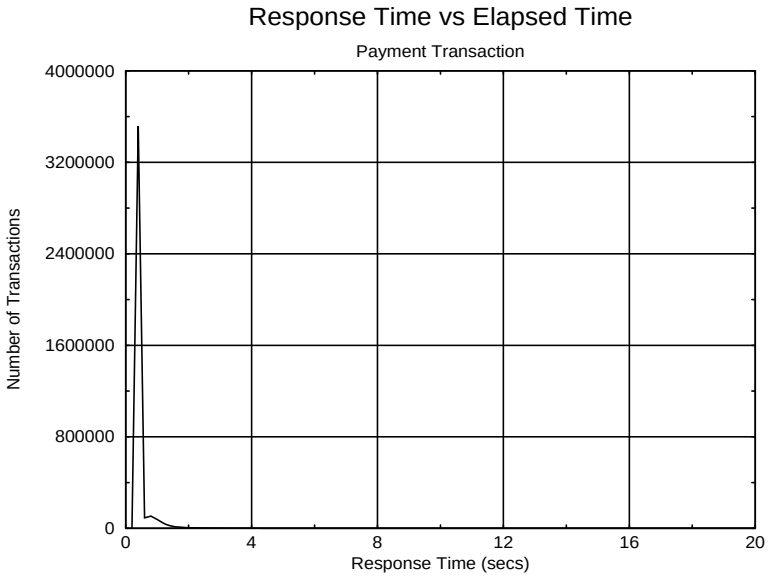


Figure 4 Payment Response Time Distribution

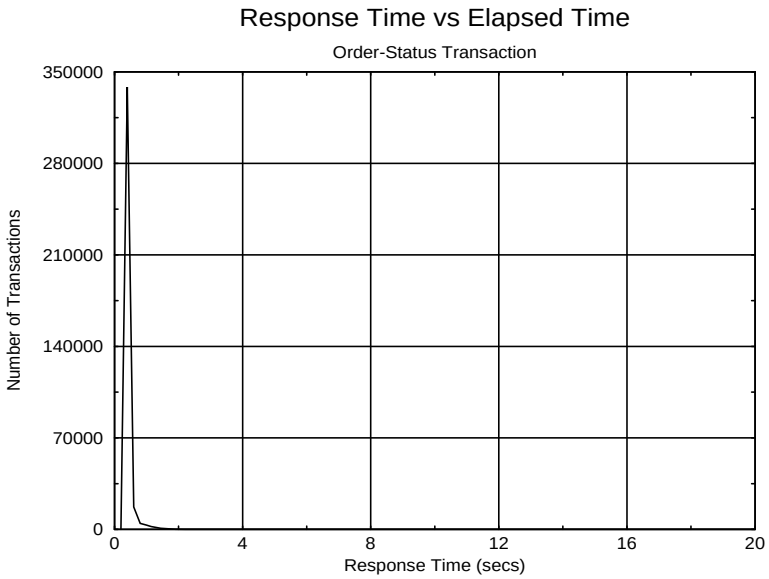


Figure 5 Order Status Response Time Distribution

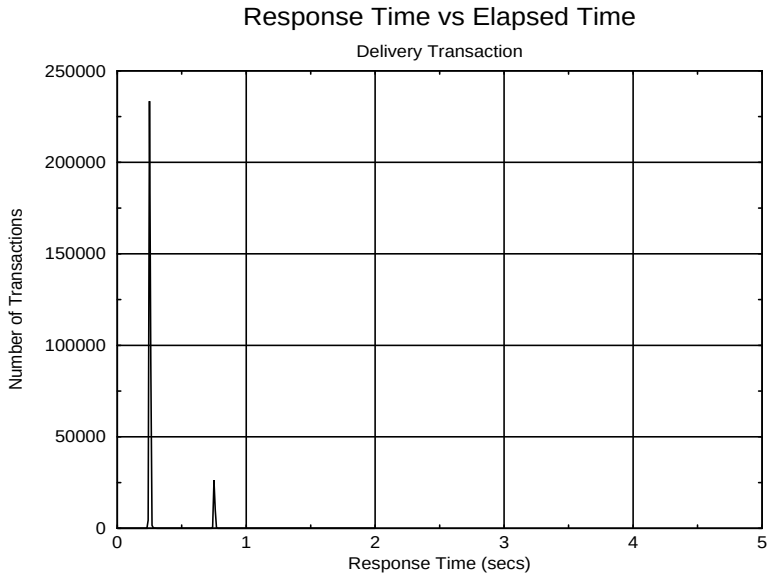


Figure 6 Delivery Response Time Distribution

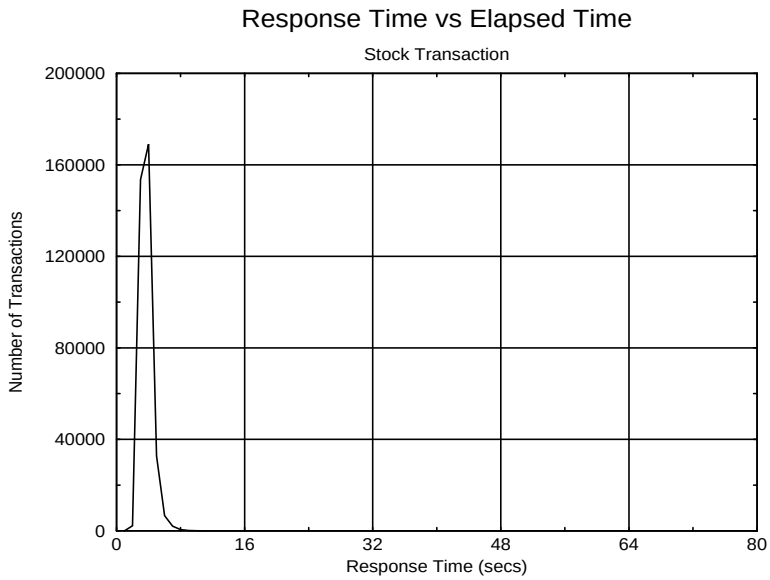


Figure 7 Stock Level Response Time Distribution

6.5 Response time versus throughput

The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New Order transaction.

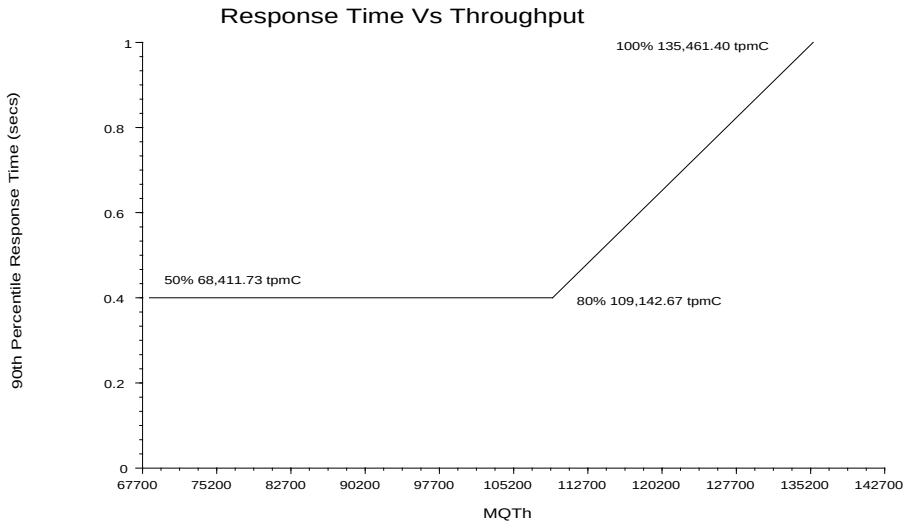


Figure 8 Response Time versus Throughput

6.6 Think Time distribution curves

Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type.

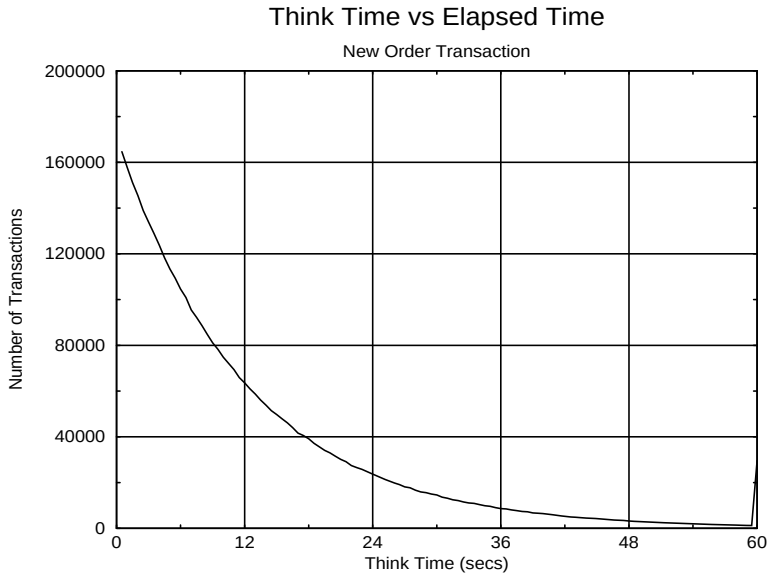


Figure 9 New Order Think Time Distribution

6.7 Throughput versus Elapsed Time

A graph of throughput versus elapsed time (see Clause 6.6.5) must be reported for the New-Order transaction.

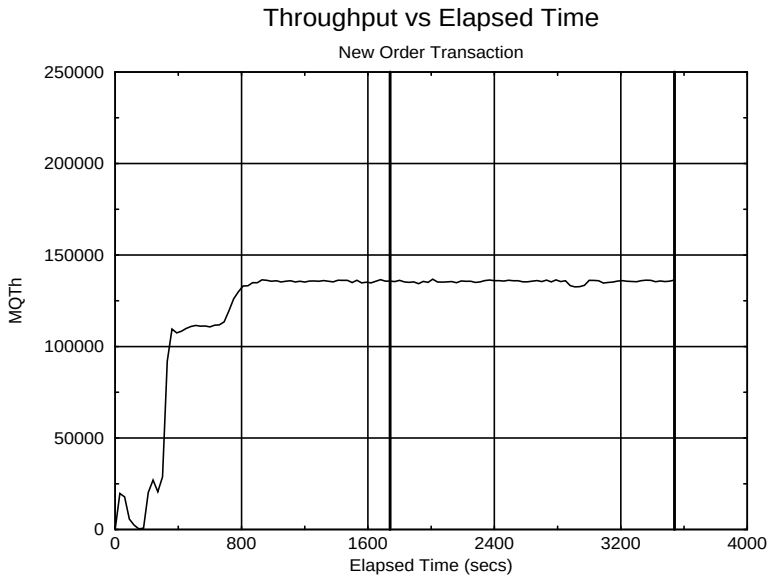


Figure 10 Throughput versus Time

6.8 Steady State Determination

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval (see Clause 5.5) must be described.

The transaction throughput rate (tpmC) and response times were relatively constant after the initial 'ramp up' period. The throughput and response time were verified by examining the throughput (tpmC) graph reported at 30 second intervals for the duration of the benchmark. Ramp up and steady state are clearly discernible in the graph in Figure 10.

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.

Modified database buffers were migrated to disk on a least-recently-used (lru) basis independent of transaction commits. In addition, every block modification was protected by redo log records. These redo log records were written to the redo log buffer in memory and were flushed to a redo log file on disk when either the transaction was completed or when the redo log buffer became full. However, due to the rapid commits during this benchmark the redo log buffer was always flushed by a commit before it became full. Also, because many transactions were committing in a short period of time, a single flush of the redo log buffer resulted in the redo log data of multiple transactions being written to disk. This is called group commit.

6.9.1 Checkpoint

During an Oracle8i checkpoint, all modified blocks in the shared buffer cache which had not been written to disk since the last checkpoint are written to disk.

6.9.2 Checkpoint Conditions

Oracle8i performs a checkpoint under the following conditions:

1. A redo log switch occurs.
2. The amount of data written to a redo log reaches the log_checkpoint_interval.
3. The elapsed time since the last checkpoint reaches the log_checkpoint_timeout.

6.9.3 Checkpoint Implementation

During each benchmark measurement, a log switch was performed in the ramp-up period. After the initial checkpoint, a logswitch was performed every 30 minutes.

6.9.4 Serializable Transactions

Oracle supports serializable transaction isolation in full compliance with the SQL92 and TPC-C requirements. This is implemented by extending the multiversity concurrency control mechanism long supported by Oracle.

Oracle queries take no read locks and see only data committed as of the beginning of the query's execution. This means that the readers and writers coexist without blocking one another, providing a high degree of concurrency and consistency. While this mode does prevent reading dirty data, Oracle's default isolation level also permits a transaction that issues a query twice to see non repeatable reads and phantoms, as defined in SQL92 and TPC-C.

Beginning with Oracle7 release 7.3, a transaction may request a high degree of isolation with the command SET TRANSACTION ISOLATION LEVEL SERIALIZABLE, as defined in SQL92. This transaction mode prevents read/write and write/write conflicts that would cause serializability failures. A session can establish this mode as its default mode, so the SET TRANSACTION command need not be issued in each transaction.

Oracle implements SERIALIZABLE mode by extending the scope of read consistency from individual query to the entire transaction. Instead of limiting a query to data committed at the time the query begins, in a serializable transaction all queries see data as of the beginning of the transaction. Thus, a serializable transaction sees a fixed snapshot of the database, established of the beginning of the transaction.

All reads within a serializable transaction sees only committed data as of the start of that transaction, plus new updates done by the transaction itself. All reads by a serializable transaction are therefore repeatable, as the transaction will access prior versions of data changed (or deleted) by other transactions after the start of the serializable transaction. This behaviour also results in phantom protection since new rows created by other transactions will be invisible to the serializable transaction.

To ensure proper isolation, a serializable transaction cannot modify rows that were changed by other transaction after the beginning of the serializable transaction. If a serializable transaction attempts to update (or delete) a row previously changed by another transaction (serializable or not) since the beginning of the serializable transaction, the update (or delete) statement will fail with error ORA-08177: "Can't serialize access", and the statement will rollback.

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE ;
SELECT ..... ;
SELECT ..... ;
UPDATE ..... ;
IF "Can't serialize access"
 THEN ROLLBACK ; LOOP and retry
ELSE COMMIT ;
```

When a serializable transaction fails with this error, the application may either commit the work executed to that point, execute additional (but different) statements, or rollback the entire transaction. Repeated attempts to execute the same statement will always fail with the error "Can't serialize access", unless the other transaction has rolled back and released its lock. This error and these recovery options are similar to the treatment of deadlocks in systems that use read locks to ensure serializable execution. In both cases, conflicts between transactions cannot be resolved unless one of the transactions rolls back and restarts or commits without re-executing the statment receiving the error.

6.10 Reproducibility

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

In a repeat run, a throughput of 135,404.67 tpmC was achieved.

6.11 Measurement Period Duration

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

The measurement interval was 1,800 seconds. There was 1 checkpoint during the measurement interval.

6.12 Transaction Mix Regulation

The method of regulation of the transaction mix (e.g., card decks or weighted random distribution) must be described. If weighted distribution is used and the RTE adjusts the weights associated with each transaction type, the maximum adjustments to the weight from the initial value must be disclosed.

The weighted distribution algorithm as described in Clause 5.2.4.1 of the TPC-C specification was used to regulate the transaction mix. Weights for the various transactions were statically assigned.

6.13 Numerical Results

The percentage of the total mix for each transaction type must be disclosed.

See Table 1 on page 13 for results.

6.14 New-Orders Rolled-Back

The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed.

See Table 1 on page 13 for results.

6.15 Order-Line Average

The average number of order-lines entered per New-Order transaction must be disclosed.

See Table 1 on page 13 for results.

6.16 Remote Order-Lines

The percentage of remote order-lines entered per New-Order transaction must be disclosed.

See Table 1 on page 13 for results.

6.17 Remote Payments

The percentage of remote payment transactions must be disclosed.

See Table 1 on page 13 for results.

6.18 Customer Lastname

The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed.

See Table 1 on page 13 for results.

6.19 Deliveries Skipped

The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.

See Table 1 on page 13 for results.

6.20 Checkpoints

The number of checkpoints in the Measurement Interval, the time in seconds from the start of the Measurement Interval to the first checkpoint and the Checkpoint Interval must be disclosed.

One checkpoint occurred at 960 seconds after the start of ramp-up and one at 1020 seconds after the start of the measurement interval. The interval between the two checkpoints was 30 minutes.

7. Clause 6 Related Items

7.1 RTE Description

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

The RTE used was developed by Sun Microsystems Computer Company and is proprietary. It consists of a *master_rte* program which forks off the individual RTE processes and controls the run. After the run completes, a separate report generator program collects all the log files and generates the final statistics of a run.

Inputs to the RTE include the names of the RTE machines to run on, client machines to attach to, the database scale, the ramp-up, measurement and ramp-down times. The script used to set these values is shown below:

```
setenv    ramp_up       1740    # ramp_up interval (secs)
setenv    stdy_state    1800    # steady-state/measurement interval (secs)
setenv    ramp_down     60      # ramp_down interval (secs)
setenv    trigger_time  1680    # Trigger time for users to login
setenv    scale         11000   # # of warehouses
setenv    comment       "Fully scaled run"
set users = ( 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750
2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750
2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 )
set rte_machines = ( r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r13 r14 r15 r16 r17
r18 r19 r20 r21 r22 r23 r24 r25 r26 r27 r28 r29 r30 r31 r32 r33 r34 r35 r36 r37
r38 r39 r40 )
set clnt_machines = ( c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17
c18 c19 c20 c21 c22 c23 c24 c25 c26 c27 c28 c29 c30 c31 c32 c33 c34 c35 c36 c37
c38 c39 c40 )
```

The code used to generate the transactions and record response times is shown in Appendix F.

7.2 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. The results of the test described in Clause 6.6.3.4 must be disclosed.

In the priced configuration, workstations are connected to the clients via telnet in the same way as the emulated system. In the tested configuration, the driver system emulates both the terminal and terminal server by making a direct connection to the SUT for each terminal.

7.3 Configuration Diagrams

A complete functional diagram of both the benchmark configuration and the configuration of the proposed (target) system must be disclosed. A detailed list of all software and hardware functionality being performed on the Driver System, and its interface to the SUT must be disclosed (see Clause 6.6.3.6).

Figure 1 on page 10 is a diagram of the benchmarked configuration and Figure 2 on page 11 shows the configuration of the priced configuration. Section 1.4 of this Full Disclosure Report gives details on both configurations.

7.4 Network Configuration

The network configurations of both the tested services and the proposed (target) services which are being represented and a thorough explanation of exactly which parts of the proposed configuration are being replaced with the Driver System must be disclosed (see Clause 6.6.4).

The tested configuration used one 10BaseT LAN for each driver system, connecting the driver system to the corresponding client system, and eight 100BaseT LAN's connecting all the client systems to the server system.

The target system is also priced with eight 100BaseT LAN's. 40 clients are connected to the Sun Enterprise 6500 Cluster; five clients per LAN. There are three 10BaseT LAN's between each client and the corresponding workstation "terminals" with 1016, 1016 and 718 workstation "terminals" on them respectively.

7.5 WAN/LAN Bandwidth

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

Local area networks (LANs) with a bandwidth of hundred (100) megabits per second were used in the tested/priced system between the clients and the server. LANs with a bandwidth of ten (10) megabits per second were used between the clients and the workstation "terminals".

7.6 Operator Intervention

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

The Sun Enterprise 6500 Cluster configuration reported does not require any operator intervention to sustain the reported throughput.

8. Clause 7 Related Items

8.1 System Pricing

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

The Executive Summary lists pricing information for all components. All Sun hardware pricing is from CAT Technologies.

The 100 BaseT hubs pricing are from NETLUX,inc.

8.2 Support Pricing

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.

8.2.1 Sun Hardware and Software Support

The Gold/Silver Programs of the SunService Support Program were used in all Sun pricing calculations. This program provides complete service with both on-site and telephone assistance. Features of this program include telephone assistance from 8:00 am to 5:00 pm, Monday - Friday; and on-site service assistance from 8:00 am to 5:00 pm, Monday - Friday; and Solaris maintenance releases. This service provides live telephone transfer of software fixes and four-hour on-site response for urgent problems.

All Sun hardware except the Sun StorEdge Arrays has a one-year warranty. The Sun StorEdge Arrays have a 6-month warranty.

8.2.2 Oracle Bronze Customer Care

Oracle Bronze Customer Care includes :

- Real-time Telephone Technical Assistance from 5:00 a.m. to 6.00 p.m. PST.
- Product updates.
- Online Access to the Real Time Support System (RTSS).
- Online Access to the Oracle Worldwide Support's Mail Server.
- Subscription to Oracle Worldwide Support's newsletter, and
- Access to Oracle Worldwide Support's private forum on CompuServe.

8.3 Discounts

The following generally available discounts to any buyer with like conditions were applied to the priced configurations:

- a 12% Sun support multi-year contract discount
- a 5% Sun support pre-payment discount
- a 10% Volume discount
- a 10% Help desk discount

8.4 Availability

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 Sun Microsystems Computer Company hardware and software available now. Oracle Software available January 31, 2000.

8.5 TpmC, Price/TpmC

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 included.

The Maximum Qualified Throughput for the Sun Enterprise 6500 Cluster was 135,461.40 tpmC at $97.10 per tpmC. The availability date for the entire configuration is January 31, 2000.

9. Clause 8 Related Items

9.1 Auditor's Report

The auditor's name, address, phone number, and a copy of the auditor's attestation letter indicating compliance must be included in the Full Disclosure Report.

See attached attestation letter for the Sun report as well as the Auditor's name and address.

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL
CERTIFIED AUDITOR

September 24, 1999

Sponsors:

Ganesh Ramamurthy
Manager, PAE
Sun Microsystems, Inc
901 San Antonio Rd
Palo Alto, Ca 94303

Karl Haas
Director OLTP, Performance
Oracle Corporation
500 Oracle Parkway
Redwood Shores, Ca 94065

Platform: Sun Enterprise 6500 Cluster (4 node) c/s
Operating system: Solaris 2.6
Database Manager: Oracle8i v 8.1.6
Transaction Manager: Bea Tuxedo 6.3

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Four node Server: Sun Enterprise 6500 Cluster				
24 x UltraSPARC II per node (400 MHz)	8 GB per node (8 MB L2 Cache per processor)	660 x 9.1 GB (7200) 585 x 9.1 GB (10k) 112 x 18 GB 16 x 4.2 GB 1 x 2.2 GB	1.0 Seconds	135,461.40
Forty Clients: Ultra 10 Model 333 (specification for each)				
1 x UltraSPARC II (333 MHz)	1024 MB	(32) with 1 x 9.1 GB (8) with 1 x 4.2 GB	n/a	n/a

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

- The database records were the proper size
- The database was properly scaled and populated
- The required ACID properties were met
- The transactions were correctly implemented
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.
- All 90% response times were under the specified maximums
- At least 90% of all delivery transactions met the 80 Second completion time limit

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

- The reported measurement interval was 30 minutes (1800 seconds)
- The reported measurement interval was representative of steady state conditions
- One checkpoint was taken during the reported measurement interval
- The repeatability of the measured performance was verified
- The 180 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

There were multiple disk substitutions on the priced configuration, and each substitution was examined for compliance.

(1) The measured system included 9.1 GB disks with 660 at 7,200 rpm and 585 at 10,000 rpm totaling 1,245 disks for data. The priced configuration priced all 10,000 rpm disks drives substituting 660 disks for the 7,200 rpm drives. Both technical specifications and performance data gathered on the measured system demonstrate the substituted disks were clearly better in performance.

(2) The measured system included 25 controller/adapters with disk cabinets daisy–chained off each controller/adapter. Some of the controllers/adapters had 3 cabinets of 22 drives (at 10,000 rpm), while others had 4 cabinets of 14 drives (at 7,200 rpm). In the priced configuration, each controller was priced with 3 cabinets of 22 drives (at 10,000 rpm). The number of disks did not change, only the number of cabinets. Performance data gathered on the measured system demonstrates this substitution was clearly better in performance.

(3) The measured server had a total of 17 drives for OS and swap space. These were 16 disks of 4.2 GB and 1 disk of 2.2 GB (all at 7,200 rpm). The priced configuration substituted 17 disks of 9.1 GB at 10,000 rpm. Performance data clearly demonstrated the I/O to these disks was insignificant during the measured interval.

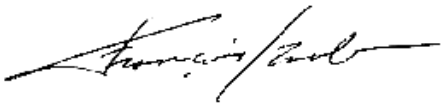
(4) The 40 clients machines measured had 32 disks of 9.1 GB and 8 disks of 4.2 GB (all at 7,200 rpm) used for OS and swap space. The priced configuration substituted 8 disks of 9.1 GB disks at 7,200 rpm. Performance data clearly demonstrated the I/O to these disks was insignificant during the measured interval.

In summary, all data disks priced on the server were 9.1 GB at 10,000 rpm. No changes were made to the log drives.

Based on the specifications of these disks and on additional performance data collected on these disks, it is my opinion that these substitutions do not have a material effect on the reported performance and meet all substitution requirements.

Respectfully Yours,

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



François Raab, President

Lorna Livingtree, Auditor

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

Appendix A. Application Code

This Appendix contains the application source code that implements the transactions and Forms modules.

client/tpcc_client.c

```
/*************************************************************************
tpcc_client.c
*************************************************************************/
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/time.h>
#include<sys/procset.h>
#include<sys/param.h>
#include<limits.h>
#include<errno.h>


#include <stdlib.h>
#include <errno.h>

#include "tpcc_client.h"
#include "tpcc_tux.h"



main()
{
int             menu_selection;
void            do_transaction(int);

initialize();
Send_Menu();

while ((menu_selection = sel_trans()) != 9) {
if ((menu_selection < 1) || (menu_selection > 5))
continue;
do_transaction(menu_selection - 1);
Send_Menu();
}
rundown();
}
initialize()
{
int             menu_selection, start, m, n;
char list[] = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
tty_in = 0;
tty_out = 1;

if (Init_Monitor()) {
fprintf(stderr, "\033[24;1H\033[mUnable to connect to TP Monitor\n\01");
exit(1);
}

get_wd();

set_display();


}

rundown()
{
restore_terminal();
Rundown_Monitor();
}

get_wd(int num)
{
num = 5 ;

setup_wd();
display_screen(num);
get_inputs(num);
}

void
do_transaction(int num)
{
int             status;
char c;

display_screen(num);
status = get_inputs(num);
if (status == 3)
return;
if ( Snd_Txn_To_Monitor(num) ){
 cleanup("\033[24;1H\033[mTransaction error occured");
}
else
display_output(num);
}
```

client/tpcc_client.h

```
/*************************************************************************
tpcc_client.h
*************************************************************************/
#include <time.h>

#include <sys/types.h>
#include <time.h>

#define BOOLEAN int
#define LINEMAX 256

#define FALSE 0
#define TRUE 1
#define NEWORDER 0
#define PAYMENT 1
#define ORDSTAT 2
#define DELIVERY 3
#define STOCKLEV 4
#define WD 5
#define MAX_OL 15
#define TPM_ERROR 1

char            date_field[80];
char            tty_name[11];
int             w_id;
int             d_id;
int             xact_type;

struct no_itm_struct {
int             ol_supply_w_id;
int             ol_i_id;
char            i_name[25];
int             ol_quantity;
int             s_quantity;
char            brand[2];
double          i_price;
doubleol_amount;
};
struct no_struct {
int             w_id;
int             d_id;
int             c_id;
int             o_id;
int             o_ol_cnt;
double          c_discount;
double          w_tax;
double          d_tax;
char            o_entry_d[20];
        char                    c_credit[3];
char            c_last[17];
struct no_itm_struct o_ol[15];
       char     status[25];
double          total;
};
struct pay_struct {
int             w_id;
int             d_id;
int             c_id;
int             c_w_id;
int             c_d_id;
double          h_amount;
double          c_credit_lim;
double          c_balance;
double          c_discount;
char            h_date[20];
char            w_street_1[21];
char            w_street_2[21];
char            w_city[21];
char            w_state[3];
char            w_zip[11];
char            d_street_1[21];
char            d_street_2[21];
char            d_city[21];
char            d_state[3];
char            d_zip[11];
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[11];
char            c_phone[17];
char            c_since[11];
char            c_credit[3];
char c_data_1[51];
char c_data_2[51];
char c_data_3[51];
char c_data_4[51];


};

struct ord_itm_struct {
int             ol_supply_w_id;
int             ol_i_id;
int             ol_quantity;
double          ol_amount;
char            ol_delivery_d[11];
};

struct ord_struct {
int             ol_cnt;
int             w_id;
int             d_id;
int             c_id;
int             o_id;
int             o_carrier_id;
double          c_balance;
char            c_first[17];
char            c_middle[3];
char            c_last[17];
char            o_entry_d[20];
struct ord_itm_struct s_ol[MAX_OL];
};

struct del_struct {
int             w_id;
int             o_carrier_id;
```

```
time_t          queue_time;
};

struct stock_struct {
int             w_id;
int             d_id;
int             threshold;
int             low_stock;
};

struct menu_struct {
intw_id;
intd_id;
};
typedef union info {
struct no_struct neworder;
struct pay_struct payment;
struct ord_struct ordstat;
struct del_struct delivery;
struct stock_struct stocklev;
struct menu_struct wd;
}         info_t;
struct io_tpcc {
int             type;
info_tinfo;
};
```

client/tpcc_forms.c

```
/*******************************************************************************
tpcc_forms.c
*******************************************************************************/
#include <stdio.h>
#include <sys/termio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include "tpcc_client.h"
#include "tpcc_forms.h"
#include "tpcc_tux.h"

static intscreen_bufindex;
static char     screen_buf[SCRBUF_LEN];
extern void     Clog(char *,...);
extern void     SCREENlog(int, char *);
const charblanks[1802] = "                                                  ";

void
setraw()
{
/** put screen in raw mode **/

extern struct tbufsave;
struct termio   tbuf;
int             status;
if (ioctl(tty_in, TCGETA, &tbuf) == -1)
return;
tbufsave = tbuf;
tbuf.c_iflag &= ~(INLCR | ICRNL | IUCLC | ISTRIP | IXON | BRKINT);
tbuf.c_oflag &= ~OPOST;
tbuf.c_lflag &= ~(ICANON | ISIG | ECHO);
tbuf.c_cc[VMIN] = LEAVE_SCREEN_MIN;
tbuf.c_cc[VTIME] = LEAVE_SCREEN_TIMEOUT;

if (ioctl(tty_out, TCSETAF, &tbuf) == -2)
syserr("ioctl_ERROR#2 - setting raw mode for STDIN error");
}
void
restore_terminal()
{/** restore terminal flags **/
extern struct tbufsave;

struct termio   tbuf;
int             status;

if (ioctl(tty_out, TCSETAF, &tbufsave) == -1)
syserr("ioctl_ERROR#3 - restoring original input terminal settings error");

tbuf = tbufsave;
if (ioctl(tty_out, TCSETAF, &tbuf) == -1)
syserr("ioctl_ERROR#4 - Forcing the original settings back for STDIN error");
}

int
sel_trans()
{
int             c, read_count;
static char     inbuf[2] = "\0\0";
int             i = 0;

read_count = read(tty_in, inbuf, 1);
if (read_count == 0)
syserr("TTY lost connection");
if (inbuf[0] == QUIT)
return 9;

switch (inbuf[0]) {
case 'n':
c = 1; break;
case 'p':
c = 2; break;
case 'o':
c = 3; break;
case 'd':
c = 4; break;
case 's':
c = 5; break;
case 'e':
c = 9; break;
}
return c;
}

int         newo_val(int *);
int         paym_val(int *);
int         ords_val(int *);
int         del_val(int *);
int         stock_val(int *);
int         wd_val(int *);
int(*p_check_function[]) () = {
&newo_val,
&paym_val,
&ords_val,
&del_val,
&stock_val,
&wd_val
};

int
get_inputs(int txn_type)
{
int         done = FALSE;
int             i, returned_key;
io_elem        *ioptr;
int             last_input;
floatfloat_h_amount = 0.0;

memset(tuxibuf, '\0', sizeof(info_t ));
int_h_amount = 0;

last_input = Forms[txn_type].num_input_elems - 1;
i = 0;
while (done == FALSE) {

ioptr = &Forms[txn_type].input_elems[i];

if (txn_type == PAYMENT){
if (i == 5)
payment_input = TRUE;
else
payment_input = FALSE;
}

returned_key = (ioptr->fptr) (ioptr->x, ioptr->y, ioptr->len,ioptr->flags, ioptr-
>dptr);


switch (returned_key) {
case BACKTAB:
if (i == 0)
i = last_input;
else
i--;
break;
case TAB:
if (i == last_input)
i = 0;
else
i++;
break;
case QUIT:
done = TRUE;
break;
case SUBMIT:
case LF:
if (screen_bufindex) {
PAINTSCRLEN(screen_buf, screen_bufindex);
screen_bufindex = 0;
}
payment_input = FALSE;
done = (p_check_function[txn_type]) (&i);
break;
}
}
return returned_key;
}
int
newo_val(int *pos)
{
int         done = FALSE;
struct no_itm_struct *ol_ptr, *ol_ptr2;
intblank_line = 0, i, j;
intblank_array[MAX_OL];
char*bufp;

iNO->w_id = w_id;

for (i=0; i<MAX_OL; i++)
blank_array[i] = 0;

if (iNO->d_id <= 0) {
*pos = 0;
message = TRUE;
PAINTSCR(MANDATORY_MSG);

} else if (iNO->c_id <= 0) {
*pos = 1;
message = TRUE;
PAINTSCR(MANDATORY_MSG);

} else {

ol_ptr = iNO->o_ol;

for (i = 0; i < MAX_OL; i++, ol_ptr++) {

if (ol_ptr->ol_i_id || ol_ptr->ol_supply_w_id
    || ol_ptr->ol_quantity)
{
/* and is that data complete */
if (ol_ptr->ol_i_id && ol_ptr->ol_supply_w_id
    && ol_ptr->ol_quantity)
{
iNO->o_ol_cnt++;
```

```
if (blank_line != 0) {
ol_ptr2 = iNO->o_ol;
for (j=0; j < i; j++) {
if (blank_array[j]) {
blank_array[j] = 0;
break;
}
ol_ptr2++;
}
ol_ptr2->ol_i_id =
ol_ptr->ol_i_id;
ol_ptr2->ol_supply_w_id =
ol_ptr->ol_supply_w_id;
ol_ptr2->ol_quantity =
ol_ptr->ol_quantity;
ol_ptr->ol_i_id = 0;
ol_ptr->ol_supply_w_id = 0;
ol_ptr->ol_quantity = 0;
blank_array[i] = 1;
bufp = output_screen;
bufp += DISPLAY_INT(bufp, 4, 3, j+FIRST_OL_ROW, ol_ptr2->ol_supply_w_id);
bufp += DISPLAY_INT(bufp, 5, 11, j+FIRST_OL_ROW, ol_ptr2->ol_i_id);
bufp += DISPLAY_INT(bufp, 2, 45, j+FIRST_OL_ROW, ol_ptr2->ol_quantity);
bufp += DISPLAY(bufp, 3, i+FIRST_OL_ROW, "    ");
bufp += DISPLAY(bufp, 11, i+FIRST_OL_ROW, "     ");
bufp += DISPLAY(bufp, 45, i+FIRST_OL_ROW, "  ");
*bufp++ = '\0';
        PAINTSCRLEN(output_screen, bufp - output_screen);
}
} else {
*pos = 2 + 3 * i;
PAINTSCR(INCOMPLINE_MSG);
message = TRUE;
iNO->o_ol_cnt = 0;
return FALSE;
}

} else {
blank_line++;
blank_array[i] = 1;
}
}

if (!iNO->o_ol_cnt) {
*pos = 2;
PAINTSCR(MANDATORY_MSG);
message = TRUE;
iNO->o_ol_cnt = 0;
return FALSE;
}
done = TRUE;
}
return done;
}

int paym_val(int *pos)
{
int         done = FALSE;
iPT->w_id = w_id;
if (iPT->d_id <= 0) {
*pos = 0;
message = TRUE;
PAINTSCR(MANDATORY_MSG);
} else if (iPT->c_w_id <= 0) {
*pos = 2;
message = TRUE;
PAINTSCR(MANDATORY_MSG);
} else if (iPT->c_d_id <= 0) {
*pos = 3;
message = TRUE;
PAINTSCR(MANDATORY_MSG);
} else if (int_h_amount <= 0) {
*pos = 5;
message = TRUE;
PAINTSCR(MANDATORY_MSG);
} else if (iPT->c_id <= 0) {
if (iPT->c_last[0] == '\0') {
message = TRUE;
PAINTSCR(ID_OR_LAST_MSG);
*pos = 1;
} else {
done = TRUE;
}
} else
done = TRUE;
iPT->h_amount = ((float)int_h_amount)/100.0 ;
return done;
}

int ords_val(int *pos)
{
int         done = FALSE;
iOS->w_id = w_id;
if (iOS->d_id <= 0) {
*pos = 0;
message = TRUE;
PAINTSCR(MANDATORY_MSG);
} else if (iOS->c_id <= 0) {
if (iOS->c_last[0] == '\0') {
message = TRUE;
PAINTSCR(ID_OR_LAST_MSG);
*pos = 1;
} else {
done = TRUE;
}
} else
done = TRUE;
return done;
}

int del_val(int *pos)
{
int         done = FALSE;
iDY->w_id = w_id;
if (iDY->o_carrier_id <= 0) {
message = TRUE;
PAINTSCR(MANDATORY_MSG);
} else {
time(&iDY->queue_time);
done = TRUE;
}
return done;
}

int stock_val(int *pos)
{
int         done = FALSE;
iSL->w_id = w_id;
iSL->d_id = d_id;
if (iSL->threshold <= 0) {
message = TRUE;
PAINTSCR(MANDATORY_MSG);
} else
done = TRUE;
return done;
}

int wd_val(int *pos)
{
int         done = FALSE;

if (iWD->w_id == 0 || iWD->d_id == 0) {
message = TRUE;
PAINTSCR(MANDATORY_MSG);
} else {
w_id = iWD->w_id ;
d_id = iWD->d_id;
done = TRUE;
}
return done;
}
void setup_wd()
{
io_elem        *p;
        char            buf[128];
        void            setup_io_elems();
        setraw();
        setup_screen_buffer(&Forms[5], 5);

p = Forms[WD].input_elems;
        p++->dptr = &iWD->w_id;
        p++->dptr = &iWD->d_id;

CLRSCN(buf);
PAINTSCR(buf);
}

void set_display()
{
int             i;
char            buf[128];
void            setup_io_elems();
for (i = 0; i < MAX_FORMS; i++)
setup_screen_buffer(&Forms[i], i);

setup_io_elems();

CLRSCN(buf);
PAINTSCR(buf);
}

void display_screen(int screen_num)
{
if (PAINTSCRLEN(Forms[screen_num].blank_form,
    Forms[screen_num].blank_formlen) == -1)
syserr("Can't write out form");
}

void Send_Menu()
{
if (PAINTSCRLEN(menu_buf, menu_buflen) == -1)
syserr("Can't send menu");
}

void setup_io_elems()
{
io_elem        *p;
int             i;
p = Forms[NEWORDER].input_elems;
p++->dptr = &iNO->d_id;
p++->dptr = &iNO->c_id;
for (i = 0; i < 15; i++) {
p++->dptr = &iNO->o_ol[i].ol_supply_w_id;
p++->dptr = &iNO->o_ol[i].ol_i_id;
p++->dptr = &iNO->o_ol[i].ol_quantity;
}
p = Forms[PAYMENT].input_elems;
p++->dptr = &iPT->d_id;
p++->dptr = &iPT->c_id;
p++->dptr = &iPT->c_w_id;
p++->dptr = &iPT->c_d_id;
p++->dptr = (int *) &iPT->c_last[0];
p->dptr = &int_h_amount;

p = Forms[ORDSTAT].input_elems;
p++->dptr = &iOS->d_id;
p++->dptr = &iOS->c_id;
p->dptr = (int *) &iOS->c_last[0];
p = Forms[DELIVERY].input_elems;
p->dptr = &iDY->o_carrier_id;
p = Forms[STOCKLEV].input_elems;
p->dptr = &iSL->threshold;
}
int
setup_screen_buffer(struct form_info * form_ptr, int txn_type)
{
FILE           *ifile;
const text_elem      *tbuf;
```

```
char           *bufp;
int             ct;
char            input_display_buf[64];
io_elem        *io_ptr;

bufp = screen_buf;
bufp += CLRSCN(bufp);
tbuf = form_ptr->tp;
while (tbuf->text) {
bufp += DISPLAY(bufp, tbuf->y, tbuf->x, tbuf->text);
tbuf++;
}
bufp += SWITCH_TO_UNDERL(bufp);

ct = 0;
for (io_ptr = form_ptr->input_elems; io_ptr->y != 999; io_ptr++) {

strncpy(input_display_buf, blanks, io_ptr->len);
input_display_buf[io_ptr->len] = '\0';
bufp += DISPLAY(bufp, io_ptr->x, io_ptr->y, input_display_buf);
ct++;

}

form_ptr->num_input_elems = ct;
bufp += SWITCH_TO_NORMAL(bufp);

if (txn_type == PAYMENT)
bufp += DISPLAY_INT(bufp, 5, 12, 4, w_id);
/*
 * shishir - changed for 11k wid
 *bufp += DISPLAY_INT(bufp, 4, 12, 4, w_id);
 */
else if (txn_type != 5)
bufp += DISPLAY_INT(bufp, 5, 12, 2, w_id);
/*
 * shishir - changed for 11k wid
 *bufp += DISPLAY_INT(bufp, 4, 12, 2, w_id);
 */
if (txn_type == STOCKLEV)
bufp += DISPLAY_INT(bufp, 2, 29, 2, d_id);
bufp += SWITCH_TO_UNDERL(bufp);
*bufp++ = '\1';
*bufp = '\0';
form_ptr->blank_formlen = bufp - screen_buf + 1;
if (!form_ptr->blank_form &&
    ((form_ptr->blank_form = malloc(form_ptr->blank_formlen)) == NULL)) {
Clog("setup_screen_buffer: malloc failed\n");
exit(1);
}
memcpy(form_ptr->blank_form, screen_buf, form_ptr->blank_formlen);
memset(screen_buf, '\0', form_ptr->blank_formlen);

}
int
read_integer(col, row, size, flags, data)
int             col, row, size, flags, *data;
{
int         exit_read_function = FALSE, previous_data_exists = FALSE;
int             return_status = TAB, bytes_read = 0, i = 0, j = 0, k = 0,
                size1 = 0, cur_col = col;
char           *bufp, temp[50];
float           q;

char            erase_field[20];

strncpy(temp, " " , 1);
bufp = screen_buf + screen_bufindex;
/* Position cursor at start of field */

if (curbuf_read == read_count || curbuf_read == 0) {
screen_buf[0] = '\0';
bufp += GOTOXY(bufp, col + size - 1, row);
PAINTSCRLEN(screen_buf, bufp - screen_buf);
bufp = screen_buf;
}
size1 = size;

if (*data > 0)
previous_data_exists = TRUE;

while (exit_read_function == FALSE) {

/*
 * Below we read from standard input into the array curbuf.
 * curbuf_read is the pointer to the array curbuf indicating
 * the position upto which the curbuf has been parsed.
 * curbuf_consumed is the number of elements in the buffer
 * temp that holds the array that is to be displayed.
 * Elements of curbuf_consumed is selectively copied from
 * curbuf Note:read_count is the total number of characters
 * in the buffer curbuf. curbuf_read is always less than or
 * equal to read_count.
 */

if (curbuf_read == read_count || curbuf_read == 0) {
curbuf_read = 0;
read_count = read(tty_in, curbuf, sizeof(curbuf));
if (read_count == 0)
syserr("TTY lost connection");
}
/*
 * int message prevents unnecessary display of warning
 * messages
 */

if (message == TRUE) {
bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW, ERASE_MSG);
message = FALSE;
}
if (previous_data_exists == TRUE) {

if (curbuf[curbuf_read] == DELETE) {
previous_data_exists = FALSE;
strncpy(erase_field, blanks, size);
erase_field[size] = '\0';
bufp += DISPLAY(bufp, col, row, erase_field);
bufp += GOTOXY(bufp, col + size - 1, row);
} else {
if (curbuf[curbuf_read] < '0' || curbuf[curbuf_read] > '9') {
exit_read_function = TRUE;
previous_data_exists = FALSE;
return_status = curbuf[curbuf_read];
curbuf[curbuf_read] = '\0';
} else {
previous_data_exists = FALSE;
strncpy(erase_field, blanks, size);
erase_field[size] = '\0';
bufp += DISPLAY(bufp, col, row, erase_field);
/*
 * bufp = screen_buf;
 */
}
}
}/* if previous_data_exists */
while ((curbuf_read < read_count) && (exit_read_function == FALSE)) {
/*
 * intermediate variable size1 for cases when
 * floating point field whose size is less than
 * actual size by 1 because of decimal.
 */
if (payment_input == TRUE)
size1 = size - 1;
/*
 * Test for integer
 */
if ((curbuf[curbuf_read] >= '0' && curbuf[curbuf_read] <= '9') ||
(curbuf[curbuf_read] == '.')) {
/*
 * Consume all integers in buffer
 */
for (; curbuf_read < read_count &&
     ((curbuf[curbuf_read] >= '0'
     && curbuf[curbuf_read] <= '9') || curbuf[curbuf_read] == '.'); curbuf_read++) {
/*
 * below we fill up temp making sure
 * the size limit is not exceeded
 */
if (curbuf_consumed < size1) {
temp[curbuf_consumed] = curbuf[curbuf_read];
curbuf_consumed++;
}
/*
 * number of elements typed in is
 * more than the size of the field
 */
else
OVERFLOW = TRUE;
/*
 * ensure the character is removed
 * after it is read
 */
curbuf[curbuf_read] = '\0';
}/* end of for curbuf is legitimate
 * number */
temp[curbuf_consumed] = '\0';/* terminate temp string */
if (payment_input == TRUE) {/* floating point field */
/* convert the ascii to float */
q = (atof(temp)) ;
bufp += DISPLAY_FLOAT(bufp, 2, (col + size - 4), row, q);
/*
if (curbuf_consumed < 3)
else
bufp += DISPLAY_FLOAT(bufp, 2, (col + size - curbuf_consumed - 1), row, q);
*/
} else {
if (curbuf_consumed < size + 1)
bufp += DISPLAY(bufp, (col + size -
      curbuf_consumed), row,
temp);
return_status = curbuf[curbuf_read];
cur_col++;
}
}
/* if curbuf[] between "0" and "9" */
/*
 * if not integer, then test for movement character
 */
else if (curbuf[curbuf_read] == TAB
  || curbuf[curbuf_read] == LF
  || curbuf[curbuf_read] == BACKTAB
  || curbuf[curbuf_read] == SUBMIT) {
if (message == TRUE) {
bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW, ERASE_MSG);
message = FALSE;
}
temp[curbuf_consumed] = '\0';
if (payment_input == TRUE) {
q = atof(temp);
*data = q*100;
}
else {
*data = atoi(temp);
}
exit_read_function = TRUE;
return_status = curbuf[curbuf_read];
curbuf[curbuf_read] = '\0';
curbuf_read++;
curbuf_consumed = 0;
}
/* if curbuf[] a movement character */
/*
 * if not integer of movement, test for DELETE
 */
else if (curbuf[curbuf_read] == DELETE) {
if (payment_input == TRUE) {/* for floating point
 * field */
if (curbuf_consumed != 0)
```

```
curbuf_consumed--;
if (message == TRUE) {
bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
ERASE_MSG);
message = FALSE;
}
OVERFLOW = FALSE;
PAINTSCR(screen_buf);
temp[curbuf_consumed] = '\0';
q = atof(temp);
curbuf[curbuf_read] = '\0';
strncpy(erase_field, blanks, size);
erase_field[size] = '\0';
bufp = screen_buf;
screen_bufindex = 0;
bufp += DISPLAY(bufp, col, row, erase_field);
if (curbuf_consumed < 3)
bufp += DISPLAY_FLOAT(bufp, 2,
  (col + size - 4), row, q);
else
bufp += DISPLAY_FLOAT(bufp, 2,
      (col + size - curbuf_consumed - 1), row, q);
if (cur_col != 0)
cur_col--;
if (curbuf_read < 40)
curbuf_read++;/* pressed key overflow
 * situations */
bufp += GOTOXY(bufp, col + size, row);
} else {
if (curbuf_consumed != 0)
curbuf_consumed--;
curbuf[curbuf_read] = '\0';
curbuf_read++;
if (message == TRUE) {
bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
ERASE_MSG);
message = FALSE;
}
OVERFLOW = FALSE;
PAINTSCR(screen_buf);
temp[curbuf_consumed] = '\0';
strncpy(erase_field, blanks, size);
erase_field[size] = '\0';
bufp = screen_buf;
screen_bufindex = 0;
bufp += DISPLAY(bufp, col, row, erase_field);
bufp += DISPLAY(bufp, (col + size -
      curbuf_consumed), row, temp);
if (cur_col != 0)
cur_col--;
bufp += GOTOXY(bufp, col + size, row);
}
}
/* end of if DELETE */
 /* could be a ^C */
else if (curbuf[curbuf_read] == QUIT) {
temp[0] = '\0';
return_status = QUIT;
curbuf[curbuf_read] = '\0';
exit_read_function = TRUE;
} else {
/** Any other character entered at the keyboard ... **/
if (message == FALSE) {
bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW, INVALID_MSG);
bufp += GOTOXY(bufp, col + size, row);
PAINTSCR(screen_buf);
bufp = screen_buf;
screen_bufindex = 0;
message = TRUE;
}
curbuf_read++;
}
}/** End of the WHILE loop **/
if (OVERFLOW == TRUE && exit_read_function == FALSE) {
/*
 * if number of characters are exceeding the field
 * limit beep and warning message is necessary
 */
if (message == FALSE) {
bufp += DISPLAY(bufp, MESSAGE_COL,
      MESSAGE_ROW, EXC_FLD_LIM_MSG);
PAINTSCR(screen_buf);
bufp = screen_buf;
screen_bufindex = 0;
message = TRUE;
}
*data = atoi(temp);
return_status = curbuf[curbuf_read];
curbuf[curbuf_read] = '\0';
curbuf_read = 0;
OVERFLOW = FALSE;
} else {
screen_bufindex = bufp - screen_buf;
if ((curbuf_read == read_count) || (curbuf_read == 0)
    || (screen_bufindex > SCRBUF_LEN - CURBUFLEN)) {
PAINTSCRLEN(screen_buf, screen_bufindex);
screen_bufindex = 0;
bufp = screen_buf;
}
}
}
/* ensuring unnecessary warning messages are removed */
if (message == TRUE) {
bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, ERASE_MSG);
message = FALSE;
PAINTSCR(screen_buf);
bufp = screen_buf;
screen_bufindex = 0;
}
return (return_status);
}
```

```
int
read_string(col, row, size, flags, data)
int             col, row, size, flags;
char           *data;
{
int         exit_read_function = FALSE, previous_data_exists = FALSE, data_full =
FALSE;
int             return_status = TAB, bytes_read = 0, i = 0, j = 0,
                size_tot = 0;
char           *bufp, temp[80];
char            erase_field[20];

strncpy(temp, "\0", 1);
curbuf_consumed = 0;
bufp = screen_buf + screen_bufindex;
/* Position cursor at start of field */

if (curbuf_read == read_count || curbuf_read == 0) {
screen_buf[0] = '\0';
bufp += GOTOXY(bufp, col, row);/* Goto input area */
PAINTSCRLEN(screen_buf, bufp - screen_buf);
bufp = screen_buf;
}
if ((*(char *) data) != '\0')
previous_data_exists = TRUE;
while (exit_read_function == FALSE) {
/*
 * Below we read from standard input into the array curbuf.
 * curbuf_read is the pointer to the array curbuf indicating
 * the position upto which the curbuf has been parsed.
 * curbuf_consumed is the number of elements in the buffer
 * temp that holds the array that is to be displayed.
 * Elements of curbuf_consumed is selectively copied from
 * curbuf Note:read_count is the total number of characters
 * in the buffer curbuf. curbuf_read is always less than or
 * equal to read_count.
 */
if (curbuf_read == read_count) {
curbuf_read = 0;
read_count = read(tty_in, curbuf, size - size_tot);
if (read_count == 0)
syserr("TTY lost connection");
}
if (message == TRUE) {
bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, ERASE_MSG);
message = FALSE;
}
if (previous_data_exists == TRUE) {
if (curbuf[curbuf_read] == DELETE) {
previous_data_exists = FALSE;
strncpy(erase_field, blanks, size);
erase_field[size] = '\0';
bufp += DISPLAY(bufp, col, row, erase_field);
bufp += GOTOXY(bufp, col, row);
} else {
if (curbuf[curbuf_read] < ' ' || curbuf[curbuf_read] > '~') {
exit_read_function = TRUE;
previous_data_exists = FALSE;
return_status = curbuf[curbuf_read];
curbuf[curbuf_read] = '\0';

} else {
previous_data_exists = FALSE;
strncpy(erase_field, blanks, size);
erase_field[size] = '\0';
bufp += DISPLAY(bufp, col, row, erase_field);
bufp += GOTOXY(bufp, col, row);
}
}
}
while ((curbuf_read < read_count) && (exit_read_function == FALSE)) {
if (curbuf[curbuf_read] >= ' ' && curbuf[curbuf_read] <= '~') {/** if between ASCII
space (040) through ~ (0176) **/
for (; curbuf[curbuf_read] >= ' '
     && curbuf[curbuf_read] <= '~'; curbuf_read++) {
/*
 * ensuring the curbuf_consumed is
 * not more than field size
 */
if (curbuf_consumed < size) {
temp[curbuf_consumed] = curbuf[curbuf_read];
curbuf_consumed++;
}
/* else overflow condition */
else
OVERFLOW = TRUE;
curbuf[curbuf_read] = '\0';/* erasing characters
 * already read from the
 * buffer */
}
temp[curbuf_consumed] = '\0';/* terminate temp string */
bufp += DISPLAY(bufp, col, row, temp);
return_status = curbuf[curbuf_read];
} else if (curbuf[curbuf_read] == TAB
   || curbuf[curbuf_read] == LF
   || curbuf[curbuf_read] == BACKTAB
   || curbuf[curbuf_read] == SUBMIT) {
if (curbuf_consumed > 0) {
if (message == TRUE) {
bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
ERASE_MSG);
message = FALSE;
}
temp[curbuf_consumed] = '\0';
strcpy(data, temp);
exit_read_function = TRUE;
return_status = curbuf[curbuf_read];
curbuf[curbuf_read] = '\0';
curbuf_read++;
curbuf_consumed = 0;
} else {
if (message == TRUE) {
```

```
bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
ERASE_MSG);
message = FALSE;
}
temp[curbuf_consumed] = '\0';
strcpy(data, temp);
exit_read_function = TRUE;
return_status = curbuf[curbuf_read];
curbuf[curbuf_read] = '\0';
curbuf_read++;
}
} else if (curbuf[curbuf_read] == DELETE) {
for (curbuf_read = curbuf_read; curbuf[curbuf_read] ==
     DELETE
     ; curbuf_read++) {
curbuf[curbuf_read] = '\0';
temp[curbuf_consumed - 1] = '\0';

if (curbuf_consumed != 0)
curbuf_consumed--;
}
if (curbuf_consumed >= 0) {
bufp += BLANK_UNDERLINE(bufp, col, row, " ");
bufp += DISPLAY(bufp, col, row, temp);
PAINTSCR(screen_buf);
bufp = screen_buf;
screen_bufindex = 0;
} else {
if (message == FALSE) {
bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
   EXC_FLD_LIM_MSG);
bufp += BEEP(bufp);
PAINTSCR(screen_buf);
bufp = screen_buf;
screen_bufindex = 0;
message = TRUE;
}
curbuf[curbuf_read] = '\0';
curbuf_read = 0;
}
} else if (curbuf[curbuf_read] == QUIT) {
temp[0] = '\0';
return_status = QUIT;
curbuf[curbuf_read] = '\0';
exit_read_function = TRUE;
} else {/** Any other character entered at the keyboard ... **/
if (message == FALSE) {
bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW, INVALID_MSG);
bufp += GOTOXY(bufp, col, row);
message = TRUE;
}
curbuf_read++;
}
}/** End of the WHILE loop **/
if (OVERFLOW == TRUE && exit_read_function == FALSE)
/** If read enough to fill the size already **/
{
if (message == FALSE) {
bufp += DISPLAY(bufp, MESSAGE_COL,
     MESSAGE_ROW, EXC_FLD_LIM_MSG);
PAINTSCR(screen_buf);
bufp = screen_buf;
screen_bufindex = 0;
message = TRUE;
}
OVERFLOW = FALSE;
temp[curbuf_consumed] = '\0';
strcpy(data, temp);
curbuf_consumed--;
return_status = curbuf[curbuf_read];
} else {
screen_bufindex = bufp - screen_buf;
if ((curbuf_read == read_count) || (curbuf_read == 0)
    || (screen_bufindex > SCRBUF_LEN - CURBUFLEN)) {
PAINTSCRLEN(screen_buf, screen_bufindex);
screen_bufindex = 0;
bufp = screen_buf;
}
}
}
if (message == TRUE) {
bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW,
ERASE_MSG);
message = FALSE;
PAINTSCR(screen_buf);
screen_bufindex = 0;
}
return (return_status);
}
voiddisplay_newo();
voiddisplay_paym();
voiddisplay_ords();
voiddisplay_del();
voiddisplay_stock();
void(*p_print_function[]) () = {
&display_newo,
&display_paym,
&display_ords,
&display_del,
&display_stock
};
display_output(int txn_type)
{
charc;

(p_print_function[txn_type]) ();
read(tty_in, &c, 1);
}
void
display_newo()
{
struct no_itm_struct *ol_ptr, *ool;
char           *bufp;
int             i, r;
double total = 0.0;

bufp = output_screen;

if (oNO->status == '\0') {

PAINTSCR(EXECUTION_STATUS_MSG);
return;

} else {

bufp += SWITCH_TO_NORMAL(bufp);
bufp += DISPLAY(bufp, 61, 2, oNO->o_entry_d);
bufp += DISPLAY(bufp, 25, 3, oNO->c_last);
bufp += DISPLAY(bufp, 52, 3, oNO->c_credit);
bufp += DISPLAY_FLOAT(bufp, 5, 64, 3, oNO->c_discount);
bufp += DISPLAY_INT(bufp, 8, 15, 4, oNO->o_id);
bufp += DISPLAY_INT(bufp, 2, 42, 4, oNO->o_ol_cnt);
bufp += DISPLAY_FLOAT(bufp, 5, 59, 4, oNO->w_tax);
bufp += DISPLAY_FLOAT(bufp, 5, 74, 4, oNO->d_tax);
ol_ptr = iNO->o_ol;
ool = oNO->o_ol;

for (i = 0, r = FIRST_OL_ROW; i < iNO->o_ol_cnt;
    r++, i++, ol_ptr++, ool++) {
bufp += DISPLAY(bufp, 19, r, ool->i_name);
bufp += DISPLAY_INT(bufp, 3, 51, r, ool->s_quantity);
bufp += DISPLAY(bufp, 58, r, ool->brand);
bufp += DISPLAY_MONEY(bufp, 6, 62, r, ool->i_price);
bufp += DISPLAY_MONEY(bufp, 7, 71, r, ool->ol_amount);
}

bufp += DISPLAY_MONEY(bufp, 8, 70, 22, oNO->total);
bufp += DISPLAY(bufp, 19, 22, oNO->status);
bufp += DISPLAY(bufp, 23, 75, "**((");
*bufp++ = '\0';
PAINTSCRLEN(output_screen, bufp - output_screen);
}
#ifdef DEBUG
Clog("DBG: Screen output chars = %d\n", (bufp - &output_screen[0]));
#endif
}
void
display_paym()
{
char           *bufp, temp[51], tempbuf2[201];
char           *make_phone(char *), *make_zip(char *);
bufp = output_screen;
bufp += SWITCH_TO_NORMAL(bufp);/* jr */
bufp += DISPLAY(bufp, 7, 2, oPT->h_date);
bufp += DISPLAY(bufp, 1, 5, oPT->w_street_1);
bufp += DISPLAY(bufp, 1, 6, oPT->w_street_2);
bufp += DISPLAY(bufp, 1, 7, oPT->w_city);
bufp += DISPLAY(bufp, 22, 7, oPT->w_state);
bufp += DISPLAY(bufp, 25, 7, make_zip(oPT->w_zip));
bufp += DISPLAY(bufp, 42, 5, oPT->d_street_1);
bufp += DISPLAY(bufp, 42, 6, oPT->d_street_2);
bufp += DISPLAY(bufp, 42, 7, oPT->d_city);
bufp += DISPLAY(bufp, 63, 7, oPT->d_state);
bufp += DISPLAY(bufp, 66, 7, make_zip(oPT->d_zip));
bufp += DISPLAY_INT(bufp, 4, 11, 9, oPT->c_id);
bufp += DISPLAY(bufp, 29, 10, oPT->c_last);
bufp += DISPLAY(bufp, 9, 10, oPT->c_first);
bufp += DISPLAY(bufp, 26, 10, oPT->c_middle);
bufp += DISPLAY(bufp, 9, 11, oPT->c_street_1);
bufp += DISPLAY(bufp, 9, 12, oPT->c_street_2);
bufp += DISPLAY(bufp, 9, 13, oPT->c_city);
bufp += DISPLAY(bufp, 30, 13, oPT->c_state);
bufp += DISPLAY(bufp, 33, 13, make_zip(oPT->c_zip));
bufp += DISPLAY(bufp, 58, 10, oPT->c_since);
bufp += DISPLAY(bufp, 58, 11, oPT->c_credit);
bufp += DISPLAY_FLOAT(bufp, 5, 58, 12, oPT->c_discount);
bufp += DISPLAY(bufp, 58, 13, make_phone(oPT->c_phone));
bufp += DISPLAY_MONEY(bufp, 14, 55, 15, oPT->c_balance);
bufp += DISPLAY_MONEY(bufp, 13, 17, 16, oPT->c_credit_lim);

/*
if (oPT->c_data_1[0] != ' ') {
*/
if (strncmp(oPT->c_credit,"BC",2) == 0) {
bufp += DISPLAY50(bufp, 12, 18, oPT->c_data_1);
bufp += DISPLAY50(bufp, 12, 19, oPT->c_data_2);
bufp += DISPLAY50(bufp, 12, 20, oPT->c_data_3);
bufp += DISPLAY50(bufp, 12, 21, oPT->c_data_4);
}
if (!oPT->h_date)
bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW - 2, BAD_INPUTS);
bufp += DISPLAY(bufp, 23, 75, "**((");
*bufp++ = '\0';
PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
Clog("DBG: Screen output chars = %d\n", (bufp - &output_screen[0]));
#endif
}
void
display_ords()
{
struct ord_itm_struct *sol;
char           *bufp;
int             i = 0, r = 8;

bufp = output_screen;
bufp += SWITCH_TO_NORMAL(bufp);
bufp += DISPLAY_INT(bufp, 4, 11, 3, oOS->c_id);
bufp += DISPLAY(bufp, 44, 3, oOS->c_last);
bufp += DISPLAY(bufp, 24, 3, oOS->c_first);
bufp += DISPLAY(bufp, 41, 3, oOS->c_middle);
bufp += DISPLAY_MONEY(bufp, 9, 15, 4, oOS->c_balance);
bufp += DISPLAY_INT(bufp, 8, 15, 6, oOS->o_id);
bufp += DISPLAY(bufp, 38, 6, oOS->o_entry_d);
bufp += DISPLAY_INT(bufp, 2, 76, 6, oOS->o_carrier_id);
```

```
for (i = 0; i < oOS->ol_cnt; i++) {

sol = &oOS->s_ol[i];

if (sol->ol_supply_w_id > 0) {
bufp += DISPLAY_INT(bufp, 4, 3, r, sol->ol_supply_w_id);
bufp += DISPLAY_INT(bufp, 6, 14, r, sol->ol_i_id);
bufp += DISPLAY_INT(bufp, 2, 25, r, sol->ol_quantity);
bufp += DISPLAY_MONEY(bufp, 8, 32, r, sol->ol_amount);
bufp += DISPLAY(bufp, 47, r, sol->ol_delivery_d);
r++;
}
}
if (!oOS->ol_cnt)
bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW - 2, BAD_INPUTS);

bufp += DISPLAY(bufp, 23, 75, "**((");
*bufp++ = '\0';
PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
Clog("DBG: Screen output chars = %d\n", (bufp - &output_screen[0]));
#endif
}
void
display_del()
{
char           *bufp;

bufp = output_screen;
/*PAINTSCR(DELIVERY_QUEUED_MSG);*/
bufp += sprintf(bufp,"%s",DELIVERY_QUEUED_MSG);
bufp += DISPLAY(bufp, 23, 75, "**((");
*bufp++ = '\0';
PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
Clog("DBG: Screen output chars = %d\n", (bufp -
 &output_screen[0]));
#endif
}
void
display_stock()
{
char           *bufp;
bufp = output_screen;
bufp += SWITCH_TO_NORMAL(bufp);/* jr */

bufp += DISPLAY_INT(bufp, 3, 12, 6, oSL->low_stock);
bufp += DISPLAY(bufp, 23, 75, "**((");
*bufp++ = '\0';
PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
Clog("DBG: low stock:%d\n", oSL->low_stock);
Clog("DBG: Screen output chars = %d\n", (bufp -
 &output_screen[0]));
#endif
}
char           *
make_phone(char *data)
{
static char     tempphone[20];
strncpy(tempphone, data, 6);
tempphone[6] = '-';
strncpy(&tempphone[7], &data[6], 3);
tempphone[10] = '-';
strncpy(&tempphone[11], &data[9], 3);
tempphone[14] = '-';
strncpy(&tempphone[15], &data[12], 4);
tempphone[19] = '\0';
return tempphone;
}
char           *
make_zip(char *data)
{
static char     temp[10];
strncpy(temp, data, 5);
temp[5] = '-';
strncpy(&temp[6], &data[5], 4);
temp[10] = '\0';
return temp;
}
```

client/tpcc_cforms.h

```
/*****************************************************************************
tpcc_forms.h
*****************************************************************************/

#include <sys/termio.h>
extern int      tty_in;
extern int      tty_out;
#define MAX_FORMS 6
#define MESSAGE_ROW 24
#define MESSAGE_COL 1
#define RTE_SYNCH_CHARACTER '\1'
#define SCRBUF_LEN 1536
#define FIRST_OL_ROW 7
#define CLRSCN(buf) sprintf(buf,"\033[H\033[2J")
#define DISPLAY_INT(buf,wid,x,y,ip) sprintf(buf,"\033[%d;%dH%*.1d", y, x,wid,ip)
#define DISPLAY_MONEY(buf,wid,x,y,fp) sprintf(buf,"\033[%d;%dH$%#*.2f",y,x,wid,fp)
#define DISPLAY_FLOAT(buf,wid,x,y,fp) sprintf(buf,"\033[%d;%dH%#*.2f", y, x,wid,fp)
#define DISPLAY(buf,x,y,txt) sprintf(buf,"\033[%d;%dH%s", y, x,txt)
#define DISPLAY50(buf,x,y,txt) sprintf(buf,"\033[%d;%dH%50.50s", y, x,txt)
#define PAINTSCR(buf) write(tty_out,buf,strlen(buf))
#define PAINTSCRLEN(buf, len) write(tty_out,buf,len)
#define SWITCH_TO_NORMAL(buf) sprintf(buf,"\033[m")
#define SWITCH_TO_UNDERL(buf) sprintf(buf,"\033[4m")
#define GOTOXY(buf,x,y) sprintf(buf,"\033[%d;%dH", y, x)
#define BEEP(buf) sprintf(buf,"\007")
#define BLANK_UNDERLINE(buf,x,y,txt) sprintf(buf,"\033[4m;\033[%d;%dH%s",y,x,txt);

#define CLRSCN_STR "'\033[H\033[2J'
#define DISPLAY_STR(x,y,txt) '\033[/**/y;/**/xH/**/txt'

/** Possible status values returned by read functions **/

#define CANCELLED 3
#define PREVIOUS_FIELD 4

#define BACKTAB 2/** CTRL B **/
#define DELETE 8
#define ESCAPE 27
#define LF 10
#define QUIT 3/** CNTRL-C Key **/
#define SPACE 32
#define SUBMIT 13/** CR Submit **/

#define TAB 9
#define UNDERLINE 95
#define LEAVE_SCREEN_MIN 300/** Minimum # of characters to leave screen **/
#define LEAVE_SCREEN_TIMEOUT 2/** Minimum time to leave screen, 10=1sec **/
static int      curbuf_consumed = 0;
static int      curbuf_read = 0;
static int      read_count = 0;
#defineCURBUFLEN300
static char     curbuf[CURBUFLEN];
static BOOLEAN  OVERFLOW = FALSE;
static BOOLEAN  message;
BOOLEAN         payment_input = FALSE;
static struct termio tbufsave;
extern void     syserr();
void            Init_Screen();
void            display_screen_array(int);
void            Send_Menu();
int             Get_Menu_Input();

typedef struct {
int             y;
int             x;
int             len;
int             flags;
int            *dptr;
int             (*fptr) ();
}               io_elem;

int             int_h_amount;
/* All the possible messages to print out */
const static char     MANDATORY_MSG[] =
"\033[24;1H\033[mMandatory data field! Please enter data.";
const static char     INVALID_MSG[] =
"\007\033[24;1HAn invalid character was entered. Please enter again.";
const static char     ERASE_MSG[] = "\033[24;1H\033[K\033[4m";
const static char     MIN1DIGIT_MSG[] = "\033[24;1H\033[mYou must enter atleast 1
digit. Please reenter.\033[4m\1";
const static char     BAD_INPUTS[] = " #### Bad input data was entered -- Select
again #### \1";
const static char     INCOMPLINE_MSG[] = "\033[24;1H\033[mOrder line is incomplete.
Please complete the whole line.\033[4m\1";
const static char     ID_OR_LAST_MSG[] = "\033[24;1H\033[mYou must enter either the
Last Name or the Customer Number.\033[4m\1";
const static char     EXC_MAX_LFT_DEC_DGT_MSG[] = "\033[24;1H\033[mMaximum digits
left of decimal point already entered. '.' expected\033[4m\1";
const static char     EXC_FLD_LIM_MSG[] = "\007\033[24;1H\033[mMaximum digits already
entered. Tab or <CR> expected\033[4m\1";/* jr */
const static char     EXECUTION_STATUS_MSG[] = "\033[m\033[22;18HItem number is not
valid";
const static char     DELIVERY_QUEUED_MSG[] = "\033[m\033[6;19HDelivery has been
queued";
int             read_integer(int, int, int, int, int *);
int             read_money(int, int, int, int, float *);
int             read_string(int, int, int, int, char *);
char            menu_buf[] = "\033[H\033[J\033[mNew-Order(n)  Payment(p)  Order-
Status(o)  Delivery(d)  Stock-Level(s)  Exit(e)";

intmenu_buflen = sizeof (menu_buf);
io_elem         neworder_inputs[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ---------- ------------------- */
2, 29, 2, 0, 0, &read_integer,
3, 12, 4, 0, 0, &read_integer,
7, 3, 5, 0, 0, &read_integer,
7, 10, 6, 0, 0, &read_integer,
7, 45, 2, 0, 0, &read_integer,
8, 3, 5, 0, 0, &read_integer,
8, 10, 6, 0, 0, &read_integer,
8, 45, 2, 0, 0, &read_integer,
9, 3, 5, 0, 0, &read_integer,
9, 10, 6, 0, 0, &read_integer,
9, 45, 2, 0, 0, &read_integer,
10, 3, 5, 0, 0, &read_integer,
10, 10, 6, 0, 0, &read_integer,
10, 45, 2, 0, 0, &read_integer,
11, 3, 5, 0, 0, &read_integer,
11, 10, 6, 0, 0, &read_integer,
11, 45, 2, 0, 0, &read_integer,
12, 3, 5, 0, 0, &read_integer,
12, 10, 6, 0, 0, &read_integer,
12, 45, 2, 0, 0, &read_integer,
13, 3, 5, 0, 0, &read_integer,
13, 10, 6, 0, 0, &read_integer,
13, 45, 2, 0, 0, &read_integer,
14, 3, 5, 0, 0, &read_integer,
14, 10, 6, 0, 0, &read_integer,
14, 45, 2, 0, 0, &read_integer,
15, 3, 5, 0, 0, &read_integer,
15, 10, 6, 0, 0, &read_integer,
15, 45, 2, 0, 0, &read_integer,
16, 3, 5, 0, 0, &read_integer,
16, 10, 6, 0, 0, &read_integer,
16, 45, 2, 0, 0, &read_integer,
17, 3, 5, 0, 0, &read_integer,
17, 10, 6, 0, 0, &read_integer,
17, 45, 2, 0, 0, &read_integer,
```

```
18, 3, 5, 0, 0, &read_integer,
18, 10, 6, 0, 0, &read_integer,
18, 45, 2, 0, 0, &read_integer,
19, 3, 5, 0, 0, &read_integer,
19, 10, 6, 0, 0, &read_integer,
19, 45, 2, 0, 0, &read_integer,
20, 3, 5, 0, 0, &read_integer,
20, 10, 6, 0, 0, &read_integer,
20, 45, 2, 0, 0, &read_integer,
21, 3, 5, 0, 0, &read_integer,
21, 10, 6, 0, 0, &read_integer,
21, 45, 2, 0, 0, &read_integer,
999
};
io_elem         payment_inputs[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
4, 52, 2, 0, 0, &read_integer,
9, 11, 4, 0, 0, &read_integer,
9, 33, 5, 0, 0, &read_integer,
9, 54, 2, 0, 0, &read_integer,
10, 29, 16, 0, 0, &read_string,
15, 24, 7, 0, 0, &read_integer,
999
};
io_elem         ordstat_inputs[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
2, 29, 2, 0, 0, &read_integer,
3, 11, 4, 0, 0, &read_integer,
3, 44, 16, 0, 0, &read_string,
999
};
io_elem         delivery_inputs[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
4, 17, 2, 0, 0, &read_integer,
999
};
io_elem         stocklev_inputs[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
4, 24, 2, 0, 0, &read_integer,
999
};
io_elem         wd_inputs[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
2, 16, 5, 0, 0, &read_integer,
2, 43, 4, 0, 0, &read_integer,
999
};

typedef struct {
int             x;
int             y;
char           *text;
}               text_elem;

const text_elem       NO_text_elem[] = {
1, 36, "New Order",
2, 1, "Warehouse:",
2, 19, "District:",
2, 55, "Date:",
3, 1, "Customer:",
3, 19, "Name:",
3, 44, "Credit:",
3, 57, "%Disc:",
4, 1, "Order Number:",
4, 25, "Number of Lines:",
4, 52, "W_tax:",
4, 67, "D_tax:",
6, 2, "Supp_W Item_Id Item Name",
6, 45, "Qty Stock B/G Price Amount",
22, 1, "Execution Status:",
22, 62, "Total:",
0
};
const text_elem       PT_text_elem[] = {
1, 38, "Payment",
2, 1, "Date:",
4, 1, "Warehouse:",
4, 42, "District:",
9, 1, "Customer:",
9, 17, "Cust-Warehouse:",
9, 39, "Cust-District:",
10, 1, "Name:",
10, 50, "Since:",
11, 50, "Credit:",
12, 50, "%Disc:",
13, 50, "Phone:",
15, 1, "Amount Paid:",
15, 23, "$",
15, 37, "New Cust-Balance:",
16, 1, "Credit Limit:",
18, 1, "Cust-Data:",
0
};
const text_elem       OS_text_elem[] = {
1, 35, "Order-Status",
2, 1, "Warehouse:",
2, 19, "District:",
3, 1, "Customer:",
3, 18, "Name:",
4, 1, "Cust-Balance:",
6, 1, "Order-Number:",
6, 26, "Entry-Date:",
6, 60, "Carrier_Number:",
7, 1, "Supply-W",
7, 14, "Item-Id",
7, 25, "Qty",
7, 33, "Amount",
7, 45, "Delivery-Date",
0
};
const text_elem       DY_text_elem[] = {
1, 38, "Delivery",
2, 1, "Warehouse:",
4, 1, "Carrier Number:",
6, 1, "Execution Status:",
0
};
const text_elem       SL_text_elem[] = {
1, 38, "Stock-Level",
2, 1, "Warehouse:",
2, 19, "District:",
4, 1, "Stock Level Threshold:",
6, 1, "low stock:",
0
};
const text_elem       WD_text_elem[] = {
2, 1, "Warehouse:",
2, 26, "District:",
0
};
#ifdefMultiple_blank_form
const char WD_blank_form[SCRBUF_LEN] =
CLRSCN_STR/**/DISPLAY_STR(2,1,'Warehouse:')/**/DISPLAY_STR(2,26,'District:')";
#endif
struct form_info {
const text_elem      *tp;
char            *blank_form;
intblank_formlen;
io_elem        *input_elems;
int             num_input_elems;
};

char            output_screen[SCRBUF_LEN];

struct form_info Forms[MAX_FORMS] = {
{NO_text_elem, 0, 0, neworder_inputs, 0},
{PT_text_elem, 0, 0, payment_inputs, 0},
{OS_text_elem, 0, 0, ordstat_inputs, 0},
{DY_text_elem, 0, 0, delivery_inputs, 0},
{SL_text_elem, 0, 0, stocklev_inputs, 0},
{WD_text_elem, 0, 0, wd_inputs, 0}
};
```

client/tpcc_log.c

```
/*****************************************************************************
clientlog.c
*****************************************************************************/
/*
 * ** clientlog.c -- Routine for writing out messages form client processes - *
 * useful for detailed error reporting and for debugging
 */

#include <stdio.h>
#include <stdarg.h>
#define BACKTAB 2/** CTRL B **/
#define DELETE 127
#define ESCAPE 27
#define LF 10
#define QUIT 3/** CNTRL-C **/
#define SPACE 32
#define SUBMIT 13/** CR **/
#define TAB 9
#define RTE_SYNCH_CHARACTER '\1'

static FILE    *clientlog;
static int      Clog_open = 0;
void
Clog(char *fmt,...)
{
}
void
SCREENlog(int *flag, char *screen)
{
char            fname[100];
int             i, char_ct;
if (!Clog_open) {
sprintf(fname, "%s/%s.%d", getenv("TMPDIR"), "CLIENTLOG",
getpid());
clientlog = fopen(fname, "w");
Clog_open = 1;
}
fprintf(clientlog, "** %d **\n", flag);
char_ct = 0;
fprintf(clientlog, "SCR: ");
for (i = 0; screen[i] != 0; char_ct++, i++) {
switch (screen[i]) {
case BACKTAB:
fprintf(clientlog, "<BACKTAB>");
break;
case DELETE:
fprintf(clientlog, "<DEL>");
break;
case ESCAPE:
fprintf(clientlog, "<ESC>");
break;
case LF:
fprintf(clientlog, "<LF>");
break;
case QUIT:
fprintf(clientlog, "<^C>");
break;
case SUBMIT:
fprintf(clientlog, "<CR>");
break;
case TAB:
fprintf(clientlog, "<TAB>");
break;
case RTE_SYNCH_CHARACTER:
```

```
fprintf(clientlog, "<^A>");
break;
default:
fprintf(clientlog, "%c", screen[i]);
}
if (char_ct > 192) {
char_ct = 0;
/* fprintf(screenlog,"\n"); */

}
}
fprintf(clientlog, "\n");
fflush(clientlog);
}

void
syserr(msg)/* print system call error message and
 * terminate */
char           *msg;
{
extern int      errno, sys_nerr;
extern char    *sys_errlist[];
extern char     tty_name[];
fprintf(stderr, "\007ERROR: (%s) %s (%d", tty_name, msg, errno);
if (errno > 0 && errno < sys_nerr)
fprintf(stderr, ";%s)\n", sys_errlist[errno]);
else
fprintf(stderr, ")\n");
exit(1);
}


void
cleanup(msg)                    /* print system call error message and
                                * terminate */
        char           *msg;
{
extern int      tty_out;
extern int      tty_in;
char c;
                write(tty_out,msg,strlen(msg));
        read(tty_in, &c, 1);
}
```

client/tpcc_tux.c

```
/*****************************************************************************
tpcc_tux.c
*****************************************************************************/
/* ** monitor.c -- All functions for Tuxedo call and return */
#include <stdio.h>
#include <stdarg.h>
#include "tpcc_client.h"
#include <atmi.h>
#include "tpcc_tux.h"

const char           *svc_names[] = {"NEWO", "PAYM", "ORDS", "DEL", "STOCK"};
int
Snd_Txn_To_Monitor(int txn_type)
{
int             status;

#if DEBUG
Clog("DBG: In Snd_Txn_To_Monitor\n");
print_input_data(txn_type);
#endif

if (txn_type == DELIVERY) {
if ( tpacall((char *)svc_names[txn_type], tuxibuf, ilen, TPNOREPLY) == -1){
/****
Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
     svc_names[txn_type], tpstrerror(tperrno));
****/
return (TPM_ERROR);
}
return(0);
} else {
if ( tpcall((char *)svc_names[txn_type], (char *)tuxibuf, ilen, &tuxobuf, &olen, 0)
== -1){
/******
Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
     svc_names[txn_type], tpstrerror(tperrno));
*****/
return (TPM_ERROR);
}
/* return user-defined failures */
return (0);
}
}

int Init_Monitor()
{
char           *text;
ilen = sizeof(struct io_tpcc);
olen = sizeof(struct io_tpcc);
if (tpinit(NULL) == -1) {
tpmerror("tpinit", tperrno);
return -1;
}
if ((tuxibuf = tpalloc("CARRAY", NULL, ilen)) == NULL) {
tpmerror("tpalloc", tperrno);
return (-1);
}
if ((tuxobuf = tpalloc("CARRAY", NULL, ilen)) == NULL) {
tpmerror("tpalloc", tperrno);
return (-1);
}
return (NULL);
}
Rundown_Monitor()
{
int             status;

tpfree(tuxibuf);
status = tpterm();

#ifdef DEBUG
Clog("terminated Tuxedo connection with status %d\n", status);
#endif
}
tpmerror(char *service_called, int errnum)
{
char            errmsg[256];
fprintf(stderr, "\033[24;1H\033[mTUXEDO: Failed %s with error: %s\n",
service_called, tpstrerror(errnum));
fprintf(stderr, "\n");
}
#ifdef DEBUG
print_input_data(int type)
{
int             i;
time_t          the_time;
the_time = time(&the_time);
Clog("DBG:=============TIME: %s == == == == == == ==\n",ctime(&the_time));
     switch (type) {
case NEWORDER:
Clog("DBG: NEWORDER INPUTS at %s\n", ctime(&the_time));
Clog("DBG: w_id: %d, d_id: %d, c_id: %d o_ol_cnt: %d\n",
     iNO->w_id, iNO->d_id, iNO->c_id, iNO->o_ol_cnt);
for (i = 0; i < iNO->o_ol_cnt; i++)
Clog("DBG: ol_i_id: %d, ol_supply_w_id: %d, ol_quantity: %d \n ",iNO-
>o_ol[i].ol_i_id, iNO->o_ol[i].ol_supply_w_id,iNO->o_ol[i].ol_quantity);
break;
case PAYMENT:
Clog("DBG: PAYMENT INPUTS at %s \n ",ctime(&the_time));
     Clog("DBG: w_id: %d, d_id: %d\n", iPT->w_id, iPT->d_id);
Clog("DBG: c_last: %s ", iPT->c_last);
Clog(" c_id: %d", iPT->c_id);
Clog(" c_w_id: %d, c_d_id: %d\n", iPT->c_w_id, iPT->c_d_id);
Clog("DBG: h_amount: %f\n", iPT->h_amount);
break;
case ORDSTAT:
Clog("DBG: ORDER STATUS INPUTS at %s \n ",ctime(&the_time));
     Clog("DBG: w_id: %d, d_id: %d\n", iOS->w_id, iOS->d_id);
Clog("DBG: c_id: %d, c_last: %s\n",
     iOS->c_id, iOS->c_last);
break;
case DELIVERY:
Clog("DBG: DELIVERY INPUTS at %s\n", ctime(&the_time));
Clog("DBG: w_id: %d, o_carrier_id: %d\n", iDY->w_id, iDY ->o_carrier_id);
break;
case STOCKLEV:
Clog("DBG: STOCK LEVEL INPUTS at %s \n ",ctime(&the_time));
Clog("DBG: w_id: %d, d_id: %d, threshold: %d\n", iSL ->w_id, iSL->d_id,iSL-
>threshold);
break;
other:
Clog("DBG: Txn_type = %d is illegal at %s \n",type,ctime(&the_time));
     }
return;
}
#endif/* ifdef DEBUG */
```

client/tpcc_tux.h

```
/*****************************************************************************
tpcc_tux.h
*****************************************************************************/
long            ilen;
long            olen;
int tty_in;
int tty_out;

char           *tuxibuf;
char           *tuxobuf;
extern void     Clog(char *,...);
#define oNO (&((info_t *) tuxobuf)->neworder)
#define oPT (&((info_t *) tuxobuf)->payment)
#define oOS (&((info_t *) tuxobuf)->ordstat)
#define oDY (&((info_t *) tuxobuf)->delivery)
#define oSL (&((info_t *) tuxobuf)->stocklev)
#define iNO (&((info_t *) tuxibuf)->neworder)
#define iPT (&((info_t *) tuxibuf)->payment)
#define iOS (&((info_t *) tuxibuf)->ordstat)
#define iDY (&((info_t *) tuxibuf)->delivery)
#define iSL (&((info_t *) tuxibuf)->stocklev)
#define iWD (&((info_t *) tuxibuf)->wd)
```

tuxserver/ora_err.h

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#ifndef ORA_ERR_H
#define ORA_ERR_H

#pragma ident "@(#)ora_err.h1.495/09/14SMI"
/*
 * this kludge is required because Oracle does not provide
 *symbolic constants in a header file
 */

#defineEDEADLOK60
```

```
#defineSQLNOTFOUND1403
#defineCOLUMN_NULL-1405
#defineEDUPLICATE-1
#defineRECOVERR-10
#defineIRRECERR-20
#defineNOERR111
#defineDEL_ERROR-666
#defineDEL_DATE_LEN7
#defineSQL_BUF_SIZE8192

#endif ORA_ERR_H
```

tuxserver/ora_errpt.c

```
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)ora_errrpt.c1.195/09/14SMI"

/*
 * these functions actually belong in ~dbbench/generic/c/msgh_log.c.  We put them
 * here because they have database specific statements.
 */

#include "ora_err.h"
#include "ora_oci.h"

errrpt(lda, cur, sqlvar)
ldadef *lda;
csrdef *cur;
text*sqlvar;
{
text msg[2048];
/*if (cur->rc) { */
oerhms(lda, (sb2) cur->rc, msg, 2048);
userlog("%s sql_variable %s\n", msg, sqlvar);

                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;
  sb4 lstat ;
  ub4 recno=2 ;

  switch (status) {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    (void) userlog("Module %s Line %d\n", fname, lineno);
    (void) userlog("Error - OCI_SUCCESS_WITH_INFO\n");
  break;
  case OCI_NEED_DATA:
    (void) userlog("Module %s Line %d\n", fname, lineno);
    (void) userlog("Error - OCI_NEED_DATA\n");
return (IRRECERR);
  case OCI_NO_DATA:
    /*
      (void) userlog("Module %s Line %d\n", fname, lineno);
      (void) userlog("Error - OCI_NO_DATA\n");
    */
    return IRRECERR;        /* for 8.1.4 */
  break;
  case OCI_ERROR:
    lstat = OCIErrorGet (errhp, (ub4) 1,
                  (text *) NULL, &errcode, errbuf,
                        (ub4) sizeof(errbuf), OCI_HTYPE_ERR);
if (errcode == NOT_SERIALIZABLE) return (errcode);
    while (lstat != OCI_NO_DATA)
    {
    (void) userlog("Module %s Line %d\n", fname, lineno);
    (void) userlog("Error - %s\n", errbuf);
    lstat = OCIErrorGet (errhp, recno++, (text *) NULL, &errcode, errbuf,
                         (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    }
    return (errcode);
  break;
  case OCI_INVALID_HANDLE:
    (void) userlog("Module %s Line %d\n", fname, lineno);
    (void) userlog("Error - OCI_INVALID_HANDLE\n");
  break;
  case OCI_STILL_EXECUTING:
    (void) userlog("Module %s Line %d\n", fname, lineno);
    (void) userlog("Error - OCI_STILL_EXECUTE\n");
   return (IRRECERR);
  case OCI_CONTINUE:
    (void) userlog("Module %s Line %d\n", fname, lineno);
    (void) userlog("Error - OCI_CONTINUE\n");
  return (IRRECERR);
  default:
    (void) userlog("Module %s Line %d\n", fname, lineno);
    (void) userlog("Error - \n");
    return (IRRECERR);
  }
  return RECOVERR;
}
```

tuxserver/ora_oci.h

```
#pragma ident "@(#)oci.h1.195/09/14SMI"

/*+============================================================+
 |        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 <ctype.h>
#include <string.h>

#include <oratypes.h>
#include <oci.h>
/****
#if __STDC__
#include <ociapr.h>
#else
#include <ocikpr.h>
#endif
*****/

typedef struct cda_def csrdef;
typedef struct cda_def ldadef;


#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 */
#defineSNAPSHOT_TOO_OLD 1555

#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 OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,cbf_nodata,cbf_data) \
       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("yufei",__LINE__,(errp), \
            OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)));


#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));
/*       ocierror(__FILE__,__LINE__,(errp), \
             OCIBindArrayOfStruct((bndp),(errp),(progvl),sizeof((indp)[0]),\
    sizeof((alen)[0]),sizeof((arcode)[0]))); */

#define OCIBNDR(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
       ocierror(__FILE__,__LINE__,(errp), \
          OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
       ocierror(__FILE__,__LINE__,(errp), \
          OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
     (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));

#define OCIBNDRAA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode,ms,cu) \
       ocierror(__FILE__,__LINE__, (errp), \
          OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
       ocierror(__FILE__,__LINE__,(errp),\
         OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
       (progv),(progvl),(ftype),(indp),(alen),(arcode),(ms),(cu),OCI_DEFAULT));
/*       ocierror(__FILE__,__LINE__,(errp),
              OCIBindArrayOfStruct((bndp),(errp),(progvl),sizeof((indp)[0]),\
```

```
    sizeof((alen)[0]),sizeof((arcode)[0]))); */

#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(tpcenv,(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
      (dvoid**)0);\
       OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
      (progvl),(ftype),(indp),(alen),\
      (arcode),OCI_DEFAULT);\

#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,sqlvar);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,sqlvar);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,sqlvar);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,(text *)ftype);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,(text *)"OEXFET")==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,(text *)"OOPEN");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,sqlstm);return(-1);}\
    else\
       DISCARD 0

#define OFEN(lda,cursor,nrows)\
    if (ofen((cursor),(nrows)))\
       {if (errrpt(lda,cursor,(text *)"OFEN")==RECOVERR) \
           {orol(lda);return(RECOVERR);} \
        else{orol(lda);return(-1);}}\
    else\
       DISCARD 0

#define OEXEC(lda,cursor)\
    if (oexec((cursor)))\
       {if (errrpt(lda,cursor,(text *)"OEXEC")==RECOVERR) \
           {orol(lda);return(RECOVERR);} \
        else{orol(lda);return(-1);}}\
    else\
       DISCARD 0


#define OCOM(lda,cursor)\
    if (ocom((lda))) \
       {errrpt(lda,cursor,(text *)"OCOM");orol(lda);return(-1);}\
    else\
       DISCARD 0


#define OEXN(lda,cursor,iters,rowoff)\
    if (oexn((cursor),(iters),(rowoff))) \
       {if (errrpt(lda,cursor,(text *)"OEXN")==RECOVERR) \
           {orol(lda);return(RECOVERR);} \
        else{orol(lda);return(-1);}}\
    else\
       DISCARD 0

#endif

/* additions done for 814 -shishir */
#define OCI_ATTR_SRVRCTXT OCI_ATTR_SERVER
#define OCI_ATTR_USERCTXT OCI_ATTR_SESSION
#define OCI_ATTR_ROWCNT OCI_ATTR_ROW_COUNT
#define OCI_HTYPE_ERR OCI_HTYPE_ERROR
#define OCI_HTYPE_STM OCI_HTYPE_STMT
```

tuxserver/tpcc_srv_del.c

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcso_srv_del.pc1.594/12/07SMI"

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

/*
 * File: delivery.pc
 * Delivery transaction code for Oracle using Message Handler
 * This program is different from the other servers, in that it
 * records transaction info in a results file.
 * Author : Shanti S
 * Date   : 4/18/94
 */

#include <sys/signal.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/utsname.h>
#include <errno.h>
#include <stdio.h>
#include "ora_err.h"

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"


#define MOVETO(element, struct_name) element = struct_name -> element
#define MOVEBACK(element, struct_name) struct_name -> element = element

static int w_id;
static int o_carrier_id;

/*static struct msgh_req message;*/         /* Transaction message */
int my_qid, my_id;
char my_name[] = "Del";

static int tx_count = 0;/* Transaction counter */
static FILE *delfile;

static char outbuf[2048];/* Buffer for results file */


get_del_tx_cnt()
{
return tx_count;
}

#include "ora_oci.h"

unsigned char cr_date[7];

#defineSQLTXT"alter session set isolation_level = serializable"

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
#define SQLTXT0 "SELECT substr(value,1,5) FROM v$parameter \
   WHERE name = 'instance_number'"
#endif

# ifdef DMLRETDEL
#define SQLTXT1 "DELETE FROM new_order WHERE no_d_id = :d_id \
                 AND no_w_id = :w_id and rownum <= 1 \
                RETURNING no_o_id into :o_id "
#endif
#ifdef DMLRETDEL
#define SQLTXT3 "UPDATE orders SET o_carrier_id = :carrier_id \
                  WHERE o_id = :o_id and o_d_id = :d_id and o_w_id = :w_id \
                  returning o_c_id into :o_c_id"
#else
#define SQLTXT3 "UPDATE orders SET o_carrier_id = :carrier_id \
   WHERE rowid = :o_rowid"
#endif

#ifdef DMLRETDEL
#define SQLTXT4 "UPDATE /*+ buffer */ order_line SET ol_delivery_d = :cr_date \
   WHERE ol_w_id = :w_id AND ol_d_id = :d_id AND ol_o_id = :o_id \
   RETURNING ol_amount into :ol_amount "
#endif

#define SQLTXT6 "UPDATE customer SET c_balance = c_balance + :amt, \
   c_delivery_cnt = c_delivery_cnt + 1 WHERE c_w_id = :w_id AND \
   c_d_id = :d_id AND c_id = :c_id"

#define NITEMS 15 /* .... Added by Ravi. */

#define NDISTS 10
#define ROWIDLEN 20
#defineDEL_DATE_LEN7


struct delctx {
   sb2 del_o_id_ind[NDISTS];
   sb2 cons_ind[NDISTS];
   sb2 w_id_ind[NDISTS];
   sb2 d_id_ind[NDISTS];
   sb2 c_id_ind[NDISTS];
   sb2 del_date_ind[NDISTS];
   sb2 carrier_id_ind[NDISTS];
```

```
   sb2 amt_ind[NDISTS];
   sb2 no_rowid_ind[NDISTS];
   sb2 o_rowid_ind[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   sb2 inum_ind;
#endif

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

   ub2 cons_len[NDISTS];
   ub2 w_id_len[NDISTS];
   ub2 d_id_len[NDISTS];
/*
   ub4 del_o_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_ptr[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 *w_id_bp3 ;
   OCIBind *w_id_bp4 ;
   OCIBind *w_id_bp5 ;
   OCIBind *w_id_bp6 ;
   OCIBind *d_id_bp;
   OCIBind *d_id_bp3 ;
   OCIBind *d_id_bp4 ;
   OCIBind *d_id_bp5 ;
   OCIBind *d_id_bp6 ;
   OCIBind *o_id_bp;
   OCIBind *cr_date_bp;
   OCIBind *c_id_bp;
   OCIBind *c_id_bp3;
   OCIBind *no_rowid_bp;
   OCIBind *carrier_id_bp;
   OCIBind *o_rowid_bp;
   OCIBind *del_o_id_bp;
   OCIBind *del_o_id_bp3;
   OCIBind *amt_bp;
   OCIBind *bstr1_bp[10];
   OCIBind *bstr2_bp[10];
   OCIDefine *inum_dp;
   OCIDefine *d_id_dp;
   OCIDefine *del_o_id_dp;
   OCIDefine *no_rowid_dp;
   OCIDefine *c_id_dp;
   OCIDefine *o_rowid_dp;
   OCIDefine *cons_dp;
   OCIDefine *amt_dp;

   int norow;
};

typedef struct delctx delctx;
delctx *dctx;

static int proc_no ;


OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
char *uid = "dbbench";
char *pwd = "dbbench";

#ifdef DMLRETDEL

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

#endif

#ifdef DMLRETDEL

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

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

#endif

int
init_del_tx()
{
/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

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

OCIStmt *curi;

   OCIInitialize(OCI_DEFAULT,(dvoid *)0,0,0,0);
   OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid
**)0);
   OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
   OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SRVRCTXT, errhp);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
      OCI_ATTR_PASSWORD, errhp);
   OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));

   OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_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 */


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

#ifdef DMLRETDEL
   actx = (amtctx *) malloc (sizeof(amtctx));
   memset(actx,(char)0,sizeof(amtctx));
#endif

   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 ((uchar *) 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 */

# ifdef DMLRETDEL
   OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd1), OCI_HTYPE_STMT, 0,
(dvoid**)0));
   sprintf ((char *) stmbuf, "%s",    SQLTXT1);
   OCIStmtPrepare(dctx->curd1, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);

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

   OCIBNDRAD(dctx->curd1, dctx->del_o_id_bp, errhp, ":o_id",
             SIZ(int),SQLT_INT,NULL,
             &dctx->oid_ctx,no_data,TPC_oid_data);
# endif

   /* open third cursor */

   OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd3), OCI_HTYPE_STMT, 0, (dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT3);
   OCIStmtPrepare(dctx->curd3, errhp, stmbuf, strlen((char *)stmbuf),
  OCI_NTV_SYNTAX, OCI_DEFAULT);


   /* bind variables */

   OCIBNDRA(dctx->curd3, dctx->carrier_id_bp,errhp,":carrier_id",dctx->carrier_id,
          SIZ(dctx->carrier_id[0]),SQLT_INT,dctx->carrier_id_ind,
          dctx->carrier_id_len,dctx->carrier_id_rcode);

# ifdef DMLRETDEL
   OCIBNDRA(dctx->curd3, dctx->w_id_bp3, errhp, ":w_id", dctx->w_id,SIZ(dctx-
>w_id[0]),
            SQLT_INT, dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
   OCIBNDRA(dctx->curd3, dctx->d_id_bp3, errhp, ":d_id", dctx->d_id,SIZ(int),
            SQLT_INT,NULL, NULL, NULL);
   OCIBNDRA(dctx->curd3, dctx->del_o_id_bp3, errhp, ":o_id", dctx->del_o_id,
            SIZ(int), SQLT_INT,NULL,NULL,NULL);
   OCIBNDRAD(dctx->curd3, dctx->c_id_bp3, errhp, ":o_c_id", SIZ(int),
            SQLT_INT,NULL,&dctx->cid_ctx,no_data, cid_data);
# endif


   /* open fourth cursor */

   OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd4), OCI_HTYPE_STMT, 0, (dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT4);
   OCIStmtPrepare(dctx->curd4, errhp, stmbuf, strlen((char *)stmbuf),
  OCI_NTV_SYNTAX, OCI_DEFAULT);

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

   /* open sixth cursor */

   OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd6), OCI_HTYPE_STMT, 0, (dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT6);
   OCIStmtPrepare(dctx->curd6, errhp, stmbuf, strlen((char *)stmbuf),
  OCI_NTV_SYNTAX, OCI_DEFAULT);

   /* bind variables */


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

#ifdef TKPROF
EXEC SQL ALTER SESSION SET SQL_TRACE = TRUE;
#endif/* TKPROF */

/****************************
 * END BLOCK OF COMMON CODE
 ****************************/

/*proc_stat_msg("init_del_tx()\n");
proc_stat(); */

return(0);
}


       /* Structure used to queue delivery transaction */
       struct req_struct {
       int w_id;
       int o_carrier_id;
       time_t qtime;           /* Time transaction was queued */
       } ;

delivery_tx(rqst)
TPSVCINFO *rqst;
{
   int i, j,v;
   int invalid;
   int tmp_id;
   int rpc, rcount, errcode, execstatus;
   int count ;

   /* float tmp_amt; changed form float to int */
   int tmp_amt;
   int del_o_id[10];
   ub4 attr_size;

intlen;
intretries=0, err = 0;

struct req_struct *delp;
       delp = (struct req_struct *)(rqst->data);

/****************************
 * BEGIN BLOCK OF COMMON CODE
 ****************************/

   /*int rpc, rcount, errcode, execstatus;*/
MOVETO(w_id, delp);
MOVETO(o_carrier_id, delp);
vgetdate(cr_date);

tx_count++;
sprintf(outbuf, "Starting transaction %d queued at %d\n",
tx_count, delp->qtime);

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

   OCIStmtExecute(tpcsvc, dctx->curd0, errhp, 1,0,0,0, OCI_DEFAULT);
   sysdate (sdate);
   userlog ("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->no_rowid_ptr_len[i] = SIZ(dctx->no_rowid_ptr[0]);
      dctx->o_rowid_ptr_len[i] = SIZ(dctx->o_rowid_ptr[0]);

      dctx->w_id[i] = w_id;
      dctx->d_id[i] = i+1; /* Added new by Ravi.... */
      dctx->carrier_id[i] = o_carrier_id;
      memcpy(dctx->del_date[i],cr_date,DEL_DATE_LEN);
   }

#ifdef DMLRETDEL /* VMM 1/13/98 */
   memset(actx,(char)0,sizeof(amtctx));
#endif /* DMLRETDEL */

   /* array select from new_order and orders tables */

   execstatus=OCIStmtExecute(tpcsvc,dctx->curd1,errhp,NDISTS,0,0,0,OCI_DEFAULT);
   if((execstatus != OCI_SUCCESS) && (execstatus != OCI_NO_DATA)) {
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     errcode = OCIERROR(errhp,execstatus);
     if(errcode == NOT_SERIALIZABLE) {
        retries++;
        goto retry;
     } else if (errcode == RECOVERR) {
        retries++;
        goto retry;
     } else {
        return -1;
     }
   }

   /* mark districts with no new order */
   attr_size = sizeof(int);
   OCIAttrGet(dctx->curd1,OCI_HTYPE_STMT,&rcount,&attr_size,OCI_ATTR_ROWCNT,errhp);
   rpc = rcount;

/* Code not present in TPCSO ---- Ravi
   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)
            userlog ("Delivery [dist %d] found no new order at %s\n", i, sdate);
      }
      if (reread) {
         sleep (60);
         sysdate (sdate);
         userlog ("Delivery wake up at %s\n", sdate);
         reread = 0;
         goto iso;
      }
   }
#endif

   /* array delete of new_order table */

   execstatus=OCIStmtExecute(tpcsvc,dctx->curd3,errhp,rpc,0,0,0,OCI_DEFAULT);
   if(execstatus != OCI_SUCCESS) {
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     errcode = OCIERROR(errhp,execstatus);
     if(errcode == NOT_SERIALIZABLE) {
        retries++;
        goto retry;
     } else if (errcode == RECOVERR) {
        retries++;
        goto retry;
     } else {
        return -1;
     }
   }

   /* mark districts with no new order */

   OCIAttrGet(dctx->curd3,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROW_COUNT,errhp);

   if (rcount != rpc) {
#ifdef TPCSO
      write_log ("Del %d: %d rows selected, %d ords updated\n",
                 my_id, rpc, rcount);
#else
      fprintf (stderr,
               "Error in TPC-C server %d: %d rows selected, %d ords updated\n",
               proc_no, rpc, rcount);
#endif
      OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      return (-1);
   }

   /* array update of order_line table */

   execstatus=OCIStmtExecute(tpcsvc,dctx->curd4,errhp,rpc,0,0,0,OCI_DEFAULT);
   if(execstatus != OCI_SUCCESS) {
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     errcode = OCIERROR(errhp,execstatus);
     if(errcode == NOT_SERIALIZABLE) {
        retries++;
        goto retry;
     } else if (errcode == RECOVERR) {
        retries++;
        goto retry;
     } else {
        return -1;
     }
   }

#ifdef DMLRETDEL
  OCIAttrGet(dctx->curd4,OCI_HTYPE_STMT,&rcount,NULL,OCI_ATTR_ROW_COUNT,errhp);
/* add up amounts */
   count=0;
   for (i=0;i<rpc;i++)
   {
     dctx->amt[i]=0;
     for (j=0;j<actx->ol_cnt[i];j++)
       if ( actx->ol_amt_rcode[i][j] == 0)
       {
         dctx->amt[i] = dctx->amt[i] + actx->ol_amt[i][j];
         count = count+1;
       }
   }
   if (rcount > rpc*NITEMS) {
#ifdef TPCSO
      write_log ("Del %d: %d ordnrs updated, %d ordl updated\n",
                 my_id, rpc, rcount);
#endif
   }
#endif

#if defined(ISO5) || defined(ISO6)
   userlog ("d_id:amount\n");
   for (i = 0; i < rpc; i++)
      userlog ("%d:%.2f ", dctx->d_id[i], dctx->amt[i]);
   userlog ("\n");
#endif

   /* array update of customer table */
#if defined(ISO5) || defined (ISO6)
   execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,0,0,
                             OCI_DEFAULT);
#else
   execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,0,0,
OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
#endif
   if(execstatus != OCI_SUCCESS) {
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     errcode = OCIERROR(errhp,execstatus);
     if(errcode == NOT_SERIALIZABLE) {
        retries++;
        goto retry;
     } else if (errcode == RECOVERR) {
        retries++;
        goto retry;
     } else {
        return -1;
     }
   }

   attr_size = sizeof(int);
   OCIAttrGet(dctx->curd6,OCI_HTYPE_STMT,&rcount,&attr_size,OCI_ATTR_ROWCNT,errhp);

   if (rcount != rpc) {
/*******
#ifdef TPCSO
      write_log ("Del %d: %d rows selected, %d cust updated\n",
                 my_id, rpc, curd6.rpc);
#else
      userlog ("%d rows selected, %d cust updated\n", rpc, curd6.rpc);
#endif
*******/
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      return (-1);
   }

#if defined(ISO5) || defined(ISO6)
   sysdate (sdate);
#ifdef ISO5
   userlog ("Delivery sleep before commit at %s\n", sdate);
#else
   userlog ("Delivery sleep before abort at %s\n", sdate);
#endif
   sleep (60);
   sysdate (sdate);
   userlog ("Delivery wake up at %s\n", sdate);
#endif

#ifdef ISO6
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
#endif
#ifdef ISO5
     OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
#endif
#if defined(ISO5) || defined(ISO6)
   sysdate (sdate);
   userlog ("Delivery completed at: %s\n", sdate);
#endif

   /* return o_id's in district id order */

   for (i = 0; i < NDISTS; i++)
      del_o_id[i] = 0;
   for (i = 0; i < rpc; i++)
      del_o_id[dctx->d_id[i] - 1] = dctx->del_o_id[i];

for (i = 0; i < 10; i++) {
if (del_o_id[i] == 0) {
/* No order found for this district */
sprintf(outbuf+strlen(outbuf),
"Delivery for District %d skipped\n", i+1);
}
```

```
else {
sprintf(outbuf+strlen(outbuf),
"Delivered order %d for district %d, warehouse %d, carrier %d\n",
del_o_id[i], i+1, w_id, o_carrier_id);
}
}

sprintf(outbuf+strlen(outbuf), "Transaction completed at %d\n", time(0));
fwrite(outbuf, strlen(outbuf), 1, delfile);
fflush(delfile);

/****************************
 * END BLOCK OF COMMON CODE
 ****************************/
return(0);
}


void
cleanup(code)
{
   if (dctx)
      free (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);

   /* log off */

   OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
   OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
   OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
   OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
   OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);

    if (my_qid >= 0)
        msgctl(my_qid, IPC_RMID, 0);
    userlog("Del %d: Exiting\n", my_id);
    exit(code);
}

/* Tuxedo */
tpsvrinit(argc, argv)
char **argv;
{
        char *p;
        char filename[200];
        int proc_no, count;
        struct utsname name;

        if ((p = getenv("TMPDIR")) == (char *)NULL) {
                userlog("TMPDIR environment variable not set\n");
                exit(1);
        }

        proc_no = atoi(argv[optind]);   /* Needs argument which is the proc_no */

        /* Get hostname of our machine  and create results file */
        uname( &name);
        strcpy(filename, p);
        sprintf(filename+strlen(filename), "/%s.del%d", name.nodename, proc_no);
        delfile = fopen(filename, "w");
        if (delfile == NULL) {
                userlog("Cannot create file %s\n", filename);
        }
        return(init_del_tx());          /* Prepare transaction */
}

void
tpsvrdone()
{
        fclose(delfile);        /* Close results file */
}

DEL(rqst)
TPSVCINFO *rqst;
{
        if (delivery_tx(rqst))
        tpreturn(TPFAIL, 0, rqst->data, sizeof(struct req_struct), 0);
else
        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct req_struct), 0);
}
```

tuxserver/tpcc_srv_newo.c

```
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcso_srv_newo.c1.1497/01/02SMI"

/*+=================================================================+
 |        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 "ora_oci.h"

#include <signal.h>
#include <stdio.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "./ora_err.h"

/* Tuxedo includes */
#include "atmi.h"
#include "userlog.h"

static inttx_count = 0;

#define MOVETO(element, struct_name) element = struct_name->element
#define MOVEBACK(element, struct_name) struct_name->element = element
#define MOVECBACK(element, cnt, struct_name) strncpy(struct_name->element, element,
cnt)


/* Lists of items on an order */
/* These structures should match the struct definitions for no_struct
 * defined in tpcc_client.h exactly.
 * Any change to those, should be reflected here
 */

struct items_inf {
int ol_supply_w_id;
int ol_i_id;
char i_name[25];
int ol_quantity;
int s_quantity;
char brand[2];
double i_price;
double ol_amount;
} ;

/* List of fields in neworder */

struct newo_inf {
int w_id;
int d_id;
int c_id;
int o_id;
int o_ol_cnt;
double c_discount;
double w_tax;
double d_tax;
char o_entry_d[20];
char c_credit[3];
char c_last[17];
struct items_inf n_items[15];
char status_mesg[25];
double total;
} ;

#ifndef AVOID_DEADLOCK
int indx[15];
void swap(struct newo_inf *str, int i, int j);
void q_sort(int *arr, struct newo_inf *str, int left, int right);
#endif

/*struct msgh_req message; */
char blank_mesg[25] = "                         ";

int my_qid, my_id;
char my_name[] = "Newo";


/****************************
 * BEGIN BLOCK OF COMMON CODE
 ****************************/

/* struct newo_inf */

int w_id;
int d_id;
int c_id;
int o_id;
int o_ol_cnt;
int c_discount;
int w_tax;
int d_tax;
char o_entry_d[20];
char c_credit[3];
char c_last[17];
char status_mesg[25];
double total;

int nol_i_id[15];
int nol_supply_w_id[15];
int nol_quantity[15];
int nol_amount[15];

char i_name[15][25];
int s_quantity[15];
char brand_gen[15][2];
char brand_generic[15];
int i_price[15];
int o_all_local;
int retries;

char cr_date[7];

#defineSQLTXT"alter session set isolation_level = serializable"

#define SQLTXT1 "BEGIN neworder.enterorder (:w_id, :d_id, :c_id, :o_ol_cnt, \
   :o_all_local, :c_discount, :c_last, :c_credit, :d_tax, :w_tax, :o_id, \
   :retry, :cr_date); END;"

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

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


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];
  sb2 s_bg_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 s_bg_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];
  ub2 s_bg_rcode[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 *curn1;
  OCIBind *ol_i_id_bp;
  OCIBind *ol_supply_w_id_bp;
  OCIBind *i_price_bp;
  OCIBind *i_name_bp;
  OCIBind *s_bg_bp;
  OCIBind *s_data_bp;
  OCIBind *i_data_bp;
  ub4 nol_i_count;
  ub4 nol_s_count;
  ub4 nol_q_count;
  ub4 nol_item_count;
  ub4 nol_name_count;
  ub4 nol_qty_count;
  ub4 nol_bg_count;
  ub4 nol_am_count;
  ub4 s_remote_count;
  ub4 s_data_count;
  ub4 i_data_count;
  OCIStmt *curn;
  OCIStmt *curn2;
  OCIStmt *curn3[10];
  OCIBind *ol_i_id_bp4;
  OCIBind *ol_supply_w_id_bp4;
  OCIBind *ol_quantity_bp;
  OCIBind *ol_quantity_bp4;
  OCIBind *s_remote_bp;
  OCIBind *s_quantity_bp;
  OCIStmt *curn4;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *c_id_bp;
  OCIBind *o_all_local_bp;
  OCIBind *o_all_cnt_bp;
  OCIBind *w_tax_bp;
  OCIBind *d_tax_bp;
  OCIBind *o_id_bp;
  OCIBind *c_discount_bp;
  OCIBind *c_credit_bp;
  OCIBind *c_last_bp;
  OCIBind *retries_bp;
  OCIBind *cr_date_bp;
  OCIBind *s_rowid_bp;
  OCIBind *id_bp[10][15];
  OCIBind *sd_bp[10][15];
  OCIDefine *Dcons[10];
  OCIDefine *Ds_rowid[10];
  OCIDefine *Di_price[10];
  OCIDefine *Di_data[10];
  OCIDefine *Ds_dist_info[10];
  OCIDefine *Ds_data[10];
  OCIDefine *Ds_quantity[10];
  OCIDefine *Di_name[10];
  OCIBind *ol_o_id_bp;
  OCIBind *ol_d_id_bp;
  OCIBind *ol_w_id_bp;
  OCIBind *ol_number_bp;
  OCIBind *ol_amount_bp;
  OCIBind *ol_dist_info_bp;
  OCIBind *null_date_bp;
  sb2 w_id_ind;
  ub2 w_id_len;
  ub2 w_id_rc;

  sb2 d_id_ind;
  ub2 d_id_len;
  ub2 d_id_rc;

  sb2 c_id_ind;
  ub2 c_id_len;
  ub2 c_id_rc;

  sb2 o_all_local_ind;
  ub2 o_all_local_len;
  ub2 o_all_local_rc;

  sb2 o_ol_cnt_ind;
  ub2 o_ol_cnt_len;
  ub2 o_ol_cnt_rc;

  sb2 w_tax_ind;
  ub2 w_tax_len;
  ub2 w_tax_rc;

  sb2 d_tax_ind;
  ub2 d_tax_len;
  ub2 d_tax_rc;

  sb2 o_id_ind;
  ub2 o_id_len;
  ub2 o_id_rc;

  sb2 c_discount_ind;
  ub2 c_discount_len;
  ub2 c_discount_rc;

  sb2 c_credit_ind;
  ub2 c_credit_len;
  ub2 c_credit_rc;

  sb2 c_last_ind;
  ub2 c_last_len;
  ub2 c_last_rc;

  sb2 retries_ind;
  ub2 retries_len;
  ub2 retries_rc;

  sb2 cr_date_ind;
  ub2 cr_date_len;
  ub2 cr_date_rc;


  int cs;
  int norow;
};

typedef struct newctx newctx;

newctx *nctx;

OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
char *uid = "dbbench";
char *pwd = "dbbench";

/*
 * Initialize the neworder transaction
 */

int status, execstatus, errcode;

int
init_newo_tx()
{

   int i, j;
   text stmbuf[2*SQL_BUF_SIZE];
   char id[4];
   char sd[4];
```

```
OCIStmt *curi;

   OCIInitialize(OCI_DEFAULT,(dvoid *)0,0,0,0);
   OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid
**)0);
   OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
   OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SRVRCTXT, errhp);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
      OCI_ATTR_PASSWORD, errhp);
   OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));

   OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_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 */

vgetdate(cr_date);

   nctx = (newctx *) malloc (sizeof(newctx));
memset(nctx, (char)0, sizeof(newctx));
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(w_id);
  nctx->d_id_ind = TRUE;
  nctx->d_id_len = sizeof(d_id);
  nctx->c_id_ind = TRUE;
  nctx->c_id_len = sizeof(c_id);
  nctx->o_all_local_ind = TRUE;
  nctx->o_all_local_len = sizeof(o_all_local);
  nctx->o_ol_cnt_ind = TRUE;
  nctx->o_ol_cnt_len = sizeof(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(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(retries);
  nctx->cr_date_ind = TRUE;
  nctx->cr_date_len = sizeof(cr_date);

  /* open first cursor */
  OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&nctx->curn1),
           OCI_HTYPE_STMT, 0, (dvoid**)0));

  sqlfile("pnew.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(w_id),SIZ(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(d_id),SIZ(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(c_id),SIZ(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(o_all_local), SIZ(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(o_ol_cnt),
  SIZ(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(w_tax),SIZ(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(d_tax),SIZ(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(o_id),SIZ(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(c_discount), SIZ(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",c_credit,
          SIZ(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",c_last,SIZ(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(retries),
          SIZ(retries),SQLT_INT,
  &nctx->retries_ind, &nctx->retries_len, &nctx->retries_rc);
 OCIBNDR(nctx->curn1, nctx->cr_date_bp, errhp, ":cr_date",cr_date,SIZ(cr_date),
SQLT_DAT, &nctx->cr_date_ind, &nctx->cr_date_len, &nctx->cr_date_rc);

  OCIBNDRAA(nctx->curn1, nctx->ol_i_id_bp,errhp,":ol_i_id",nol_i_id,
           SIZ(int), SQLT_INT, nctx->nol_i_id_ind,nctx->nol_i_id_len,
           nctx->nol_i_id_rcode,NITEMS,&nctx->nol_i_count);
  OCIBNDRAA(nctx->curn1, nctx->ol_supply_w_id_bp, errhp, ":ol_supply_w_id",
           nol_supply_w_id,SIZ(int),SQLT_INT, nctx->nol_supply_w_id_ind,
           nctx->nol_supply_w_id_len, nctx->nol_supply_w_id_rcode,
           NITEMS, &nctx->nol_s_count);
 OCIBNDRAA(nctx->curn1, nctx->ol_quantity_bp,errhp,":ol_quantity",nol_quantity,
SIZ(int),SQLT_INT,nctx->nol_quantity_ind,nctx->nol_quantity_len,
           nctx->nol_quantity_rcode,NITEMS,&nctx->nol_q_count);
  OCIBNDRAA(nctx->curn1, nctx->i_price_bp,errhp,":i_price",i_price,SIZ(int),
            SQLT_INT, nctx->i_price_ind,nctx->i_price_len,nctx->i_price_rcode,
            NITEMS, &nctx->nol_item_count);
  OCIBNDRAA(nctx->curn1, nctx->i_name_bp,errhp,":i_name",i_name,
             SIZ(i_name[0]),SQLT_STR, nctx->i_name_ind,nctx->i_name_len,
             nctx->i_name_rcode,NITEMS,&nctx->nol_name_count);
  OCIBNDRAA(nctx->curn1, nctx->s_quantity_bp,errhp,":s_quantity",s_quantity,
           SIZ(int), SQLT_INT,nctx->s_quant_ind,nctx->s_quant_len,
           nctx->s_quant_rcode,NITEMS,&nctx->nol_qty_count);
  OCIBNDRAA(nctx->curn1, nctx->s_bg_bp,errhp,":brand_generic",brand_generic,
           SIZ(char), SQLT_CHR,nctx->s_bg_ind,nctx->s_bg_len,
           nctx->s_bg_rcode,NITEMS,&nctx->nol_bg_count);
  OCIBNDRAA(nctx->curn1, nctx->ol_amount_bp,errhp,":ol_amount",nol_amount,
           SIZ(int),SQLT_INT, nctx->nol_amount_ind,nctx->nol_amount_len,
           nctx->nol_amount_rcode,NITEMS,&nctx->nol_am_count);
  OCIBNDRAA(nctx->curn1, nctx->s_remote_bp,errhp,":s_remote",nctx->s_remote,
           SIZ(int),SQLT_INT, nctx->s_remote_ind,nctx->s_remote_len,
           nctx->s_remote_rcode,NITEMS,&nctx->s_remote_count);


  /* open second cursor */
  OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&nctx->curn2), OCI_HTYPE_STMT,
                                0, (dvoid**)0));
  sprintf ((char *) stmbuf, SQLTXT2);
  OCIERROR(errhp,OCIStmtPrepare(nctx->curn2, errhp, stmbuf,
           strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));


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

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


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


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

/****************************
 * END BLOCK OF COMMON CODE
 ****************************/


get_newo_tx_cnt()
{
        return tx_count;
}


/*
 * This function executes the neworder transaction
 */


#if ACID
#include <sys/types.h>
#include <time.h>
time_t curtime, *timep = &curtime;
#endif

neworder_tx(rqst)
TPSVCINFO *rqst;
{

/****************************
 * BEGIN BLOCK OF COMMON CODE
 ****************************/
```

```
int i, j, k;
int rpc, rpc3, rowoff, iters;
int rcount;
ub4 flags;
ub4 attr_size;
        struct newo_inf *neworder_p;

#if ACID
int reread;
char sdate[30];
time(timep);
userlog("ACID NEWORDER started at %s\n", ctime(timep));
#endif

        neworder_p = (struct newo_inf *)(rqst->data);

    MOVETO(w_id, neworder_p);
    MOVETO(d_id, neworder_p);
    MOVETO(c_id , neworder_p);
    MOVETO(o_ol_cnt, neworder_p);
tx_count++;

strcpy(neworder_p->status_mesg, blank_mesg);
vgetdate(cr_date);
retry:
status = 0;

o_all_local = 1;
for (i = 0; i < o_ol_cnt; i++) {
nol_supply_w_id[i] = neworder_p->n_items[i].ol_supply_w_id;
if (nol_supply_w_id[i] == w_id) {
nctx->s_remote[i] = 0;
}
else {
nctx->s_remote[i] = 1;
o_all_local = 0;
}
nol_i_id[i] = neworder_p->n_items[i].ol_i_id;
nol_quantity[i] = neworder_p->n_items[i].ol_quantity;
}
for (; i < 15; i++) {
nol_supply_w_id[i] = 0;
nol_i_id[i] = 0;
}

#ifndef AVOID_DEADLOCK
   for(i=0;i<15;i++)
     indx[i] = i;

   q_sort(nol_i_id, neworder_p, 0, o_ol_cnt - 1);
#endif

   nctx->w_id_ind = TRUE;
   nctx->w_id_len = sizeof(w_id);
   nctx->d_id_ind = TRUE;
   nctx->d_id_len = sizeof(d_id);
   nctx->c_id_ind = TRUE;
   nctx->c_id_len = sizeof(c_id);
   nctx->o_all_local_ind = TRUE;
   nctx->o_all_local_len = sizeof(o_all_local);
   nctx->o_ol_cnt_ind = TRUE;
   nctx->o_ol_cnt_len = sizeof(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(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(retries);
   nctx->cr_date_ind = TRUE;
   nctx->cr_date_len = sizeof(cr_date);
   /* this is the row count */
   rcount = o_ol_cnt;
   nctx->nol_i_count = o_ol_cnt;
   nctx->nol_q_count = o_ol_cnt;
   nctx->nol_s_count = o_ol_cnt;
   nctx->s_remote_count = o_ol_cnt;

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

   /* initialization for array operations */
   for (i = 0; i < o_ol_cnt; i++) {
      nctx->ol_w_id[i] = w_id;
      nctx->ol_d_id[i] = d_id;
      nctx->ol_number[i] = i + 1;
      nctx->null_date_ind[i]= TRUE;
      nctx->nol_i_id_ind[i] = 0;
      nctx->nol_supply_w_id_ind[i] = TRUE;
      nctx->nol_quantity_ind[i] = TRUE;
      nctx->nol_amount_ind[i] = TRUE;
      nctx->ol_w_id_ind[i] = TRUE;
      nctx->ol_d_id_ind[i] = TRUE;
      nctx->ol_o_id_ind[i] = TRUE;
      nctx->ol_number_ind[i] = TRUE;
      nctx->ol_dist_info_ind[i] = TRUE;
      nctx->s_remote_ind[i] = TRUE;
      nctx->s_data_ind[i] = TRUE;
      nctx->i_data_ind[i] = TRUE;
      nctx->s_quant_ind[i] = TRUE;
      nctx->s_bg_ind[i] = TRUE;
      nctx->cons_ind[i] = TRUE;
      nctx->s_rowid_ind[i] = TRUE;
      nctx->nol_i_id_len[i] = sizeof(int);
      nctx->nol_supply_w_id_len[i] = sizeof(int);
      nctx->nol_quantity_len[i] = sizeof(int);
      nctx->nol_amount_len[i] = sizeof(int);
      nctx->ol_w_id_len[i] = sizeof(int);
      nctx->ol_d_id_len[i] = sizeof(int);
      nctx->ol_o_id_len[i] = sizeof(int);
      nctx->ol_number_len[i] = sizeof(int);
      nctx->ol_dist_info_len[i] = nctx->s_dist_info_len[i];
      nctx->null_date_len[i]=sizeof(OCIDate);
      nctx->s_remote_len[i] = sizeof(int);
      nctx->s_data_len[i] = sizeof(int);
      nctx->i_data_len[i] = sizeof(int);
      nctx->s_quant_len[i] = sizeof(int);
      nctx->s_rowid_len[i] = sizeof(nctx->s_rowid_ptr[0]);
      nctx->cons_len[i] = sizeof(int);
      nctx->i_name_len[i]=0;
      nctx->s_bg_len[i] = 0;
   }
   for (i = o_ol_cnt; i < NITEMS; i++) {
      nctx->nol_i_id_ind[i] = NA;
      nctx->nol_supply_w_id_ind[i] = NA;
      nctx->nol_quantity_ind[i] = NA;
      nctx->nol_amount_ind[i] = NA;
      nctx->ol_w_id_ind[i] = NA;
      nctx->ol_d_id_ind[i] = NA;
      nctx->ol_o_id_ind[i] = NA;
      nctx->ol_number_ind[i] = NA;
      nctx->ol_dist_info_ind[i] = NA;
      nctx->null_date_ind[i]= NA;
      nctx->s_remote_ind[i] = NA;
      nctx->s_data_ind[i] = NA;
      nctx->i_data_ind[i] = NA;
      nctx->s_quant_ind[i] = NA;
      nctx->s_bg_ind[i] = NA;
      nctx->cons_ind[i] = NA;
      nctx->s_rowid_ind[i] = NA;

      nctx->nol_i_id_len[i] = 0;
      nctx->nol_supply_w_id_len[i] = 0;
      nctx->nol_quantity_len[i] = 0;
      nctx->nol_amount_len[i] = 0;
      nctx->ol_w_id_len[i] = 0;
      nctx->ol_d_id_len[i] = 0;
      nctx->ol_o_id_len[i] = 0;
      nctx->ol_number_len[i] = 0;
      nctx->ol_dist_info_len[i] = 0;
      nctx->null_date_len[i]= 0;
      nctx->s_remote_len[i] = 0;
      nctx->i_data_len[i] = 0;
      nctx->s_data_len[i] = 0;
      nctx->s_quant_len[i] = 0;
      nctx->s_rowid_len[i] = 0;
      nctx->cons_len[i] = 0;
      nctx->i_name_len[i]=0;
      nctx->s_bg_len[i] = 0;
   }

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


   if(execstatus != OCI_SUCCESS) {
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     errcode = OCIERROR(errhp,execstatus);
     if(errcode == NOT_SERIALIZABLE) {
retries++;
goto retry;
     } else if (errcode == RECOVERR) {
retries++;
goto retry;
     } else {
return -1;
     }
   }

   /* did the txn succeed ? */
   if (rcount != o_ol_cnt)
   {
     status = rcount - o_ol_cnt;
     o_ol_cnt = rcount;
   }
   if (status) {
        strcpy(neworder_p->status_mesg, "Item number is not valid");
   }


   total = 0.0;
   for (i = 0; i < o_ol_cnt; i++) {
neworder_p->n_items[i].s_quantity = s_quantity[i];
neworder_p->n_items[i].i_price = ((double)i_price[i]) / 100;
neworder_p->n_items[i].ol_amount = ((double)nol_amount[i])/ 100;
strcpy(neworder_p->n_items[i].i_name, i_name[i]);
brand_gen[i][0] = brand_generic[i];
        brand_gen[i][1] = '\0';
strcpy(neworder_p->n_items[i].brand, brand_gen[i]);
total = total + nol_amount[i];
        }
        total *= ((double)(10000  - c_discount)/10000) *
     (1.0 + ((double)(d_tax)/10000) + ((double)(w_tax)/10000));
        total = total/100;

/* fill in date for o_entry_d from time in beginning of txn */
cvtdmyhms(cr_date, o_entry_d);
MOVEBACK(o_id , neworder_p);
neworder_p->c_discount = ((double)c_discount) / 100;
neworder_p->w_tax = ((double)w_tax) / 100;
neworder_p->d_tax = ((double)d_tax) / 100;
MOVECBACK(o_entry_d, 20, neworder_p);
MOVECBACK(c_credit, 2, neworder_p);
```

```
MOVECBACK(c_last, 16, neworder_p);
MOVEBACK(total, neworder_p);

#ifndef AVOID_DEADLOCK
    q_sort(indx, neworder_p, 0,o_ol_cnt-1);
#endif


/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/

#if ACID
        time(timep);
        userlog("ACID NEWORDER w_id=%d, d_id=%d, c_id=%d, o_id=%d, total=%f\n",
                w_id, d_id, c_id, o_id, total);
        userlog("ACID NEWORDER completed at %s\n", ctime(timep));
#endif
return(0);
}


/* 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 */

void
cleanup(code)
int code;
{
int i;


   if (nctx)
      free (nctx);

   OCIHandleFree((dvoid *)nctx->curn1,OCI_HTYPE_STMT);
   OCIHandleFree((dvoid *)nctx->curn2,OCI_HTYPE_STMT);
   for (i = 0; i < 10; i++)
     OCIHandleFree((dvoid *)(nctx->curn3)[i],OCI_HTYPE_STMT);
   OCIHandleFree((dvoid *)nctx->curn4,OCI_HTYPE_STMT);

   /* log off */

   OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
   OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
   OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
   OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
   OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);

   if (my_qid >= 0)
msgctl(my_qid, IPC_RMID, 0);
#if DEBUG
write_log("Newo %d: Exiting. Completed %d transactions\n", my_id, tx_count);
#endif
exit(code);
}


#ifndef AVOID_DEADLOCK
void q_sort(int *arr,struct newo_inf *str,int left, int right)
{
  int i;

  if(left >= right)
    return;
  for(i=left+1;i<=right;i++)
    if(arr[i] < arr[left])
      swap(str,left,i);
  q_sort(arr,str,left+1,right);
}

void swap(struct newo_inf *str, int i, int j)
{
  int temp;
  double dtemp;
  char tmpstr[25];
  char tmpch[2];

  temp = indx[i];
  indx[i] = indx[j];
  indx[j] = temp;

  temp = nol_i_id[i];
  nol_i_id[i] = nol_i_id[j];
  nol_i_id[j] = temp;
 temp = nol_supply_w_id[i];
  nol_supply_w_id[i] = nol_supply_w_id[j];
  nol_supply_w_id[j] = temp;

  temp = nol_quantity[i];
  nol_quantity[i] = nol_quantity[j];
  nol_quantity[j] = temp;

  strncpy(tmpstr,str->n_items[i].i_name, 25);
  strncpy(str->n_items[i].i_name,str->n_items[j].i_name, 25);
  strncpy(str->n_items[j].i_name,tmpstr, 25);

  temp = str->n_items[i].s_quantity;
  str->n_items[i].s_quantity = str->n_items[j].s_quantity;
  str->n_items[j].s_quantity = temp;

/*
  tmpch = str->n_items[i].brand;
  str->n_items[i].brand= str->n_items[j].brand;
  str->n_items[j].brand= tmpch;
*/
  strncpy( tmpch ,str->n_items[i].brand, 2);
  strncpy(str->n_items[i].brand ,str->n_items[j].brand, 2);
  strncpy(str->n_items[j].brand ,tmpch, 2);

  dtemp = str->n_items[i].i_price;
  str->n_items[i].i_price = str->n_items[j].i_price;
  str->n_items[j].i_price = dtemp;

  dtemp = str->n_items[i].ol_amount;
  str->n_items[i].ol_amount = str->n_items[j].ol_amount;
  str->n_items[j].ol_amount = dtemp;

}
#endif /* AVOID_DEADLOCK */


/* Start of Tuxedo code */
int
tpsvrinit(argc, argv)
char **argv;
{
        return(init_newo_tx());         /* Prepare transaction */
}

void
tpsvrdone()
{
}

NEWO(rqst)
TPSVCINFO *rqst;
{
    if (neworder_tx(rqst)) {
        tpreturn(TPFAIL, 0, rqst->data, sizeof(struct newo_inf), 0);
    }
    else {
        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct newo_inf), 0);
    }
}
```

tuxserver/tpcc_srv_ords.c

```
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcso_srv_ords.c1.1797/01/02SMI"

/*+=============================================================+
 |        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA       |
 |                    OPEN SYSTEMS PERFORMANCE GROUP                |
 |                         All Rights Reserved                      |
 +=============================================================+
 | FILENAME
 |    plord.c
 | DESCRIPTION
 |    OCI version (using PL/SQL stored procedure) of
 |    ORDER STATUS transaction in TPC-C benchmark.
 +=============================================================*/

#include "ora_oci.h"

#include <signal.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "./ora_err.h"

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

struct ord_itm_inf {
int ol_supply_w_id;
int ol_i_id;
intol_quantity;
double ol_amount;
char ol_delivery_d[11];
};

struct ord_inf {
int o_ol_cnt;
int w_id;
int d_id;
int c_id;
int o_id;
int o_carrier_id;
double c_balance;
char c_first[17];
char c_middle[3];
char c_last[17];
char o_entry_d[20];
struct ord_itm_inf o_items[15];
};

#define SQLTXT1 "alter session set isolation_level = serializable"

#define MOVETO(element, struct_name) element = struct_name->element
#define MOVEBACK(element, struct_name) struct_name->element = element
#define MOVECBACK(element, cnt, struct_name) strncpy(struct_name->element, element,
cnt)

/* List of fields in ordstat */
/* This structure should be EXACTLY identical to the one declared in client.h */
/* Lists of items on an order */


static int tx_count = 0;       /* Transaction counter */

#if ACID
#include <sys/types.h>
#include <time.h>
time_t curtime;
time_t *timep = &curtime;
#endif
```

```
/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

#define NITEMS 15

struct ordctx {
   sb2 c_rowid_ind[100];
   sb2 ol_supply_w_id_ind[NITEMS];
   sb2 ol_i_id_ind[NITEMS];
   sb2 ol_quantity_ind[NITEMS];
   sb2 ol_amount_ind[NITEMS];
   sb2 ol_delivery_d_ind[NITEMS];
   sb2 ol_w_id_ind;
   sb2 ol_d_id_ind;
   sb2 ol_o_id_ind;
   sb2 c_id_ind;
   sb2 c_first_ind;
   sb2 c_middle_ind;
   sb2 c_balance_ind;
   sb2 c_last_ind;
   sb2 o_id_ind;
   sb2 o_entry_d_ind;
   sb2 o_carrier_id_ind;
   sb2 o_ol_cnt_ind;

   ub2 c_rowid_len[100];
   ub2 ol_supply_w_id_len[NITEMS];
   ub2 ol_i_id_len[NITEMS];
   ub2 ol_quantity_len[NITEMS];
   ub2 ol_amount_len[NITEMS];
   ub2 ol_delivery_d_len[NITEMS];
   ub2 ol_w_id_len;
   ub2 ol_d_id_len;
   ub2 ol_o_id_len;

   ub2 c_rowid_rcode[100];
   ub2 ol_supply_w_id_rcode[NITEMS];
   ub2 ol_i_id_rcode[NITEMS];
   ub2 ol_quantity_rcode[NITEMS];
   ub2 ol_amount_rcode[NITEMS];
   ub2 ol_delivery_d_rcode[NITEMS];
   ub2 ol_w_id_rcode;
   ub2 ol_d_id_rcode;
   ub2 ol_o_id_rcode;

   ub4 ol_supply_w_id_csize;
   ub4 ol_i_id_csize;
   ub4 ol_quantity_csize;
   ub4 ol_amount_csize;
   ub4 ol_delivery_d_csize;
   ub4 ol_w_id_csize;
   ub4 ol_d_id_csize;
   ub4 ol_o_id_csize;

   OCIStmt *curo0;
   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 *o_id_bp;
   OCIBind *c_rowid_bp;
   OCIDefine *c_rowid_dp;
   OCIDefine *c_last_dp;
   OCIDefine *c_last_dp1;
   OCIDefine *c_id_dp;
   OCIDefine *c_id_dp1;
   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[100];
   OCIRowid *middle_cust;
   int cs;
   int cust_idx;
   int norow;
};

typedef struct ordctx ordctx;
ordctx *octx;

unsigned char o_entry_d_base[7];
unsigned char ol_d_base[15][7];

/* struct ord_inf elements */
int w_id;
int d_id;
int c_id, bylastname;
int o_id;
int o_carrier_id;
int o_ol_cnt;
double c_balance;
char c_first[17];
char c_middle[3];
char c_last[17];
char o_entry_d[20];

intol_supply_w_id[15];
intol_i_id[15];
intol_quantity[15];
intol_amount[15];
charol_delivery_d[15][11];


OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
char *uid = "dbbench";
char *pwd = "dbbench";

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/


get_ords_tx_cnt()
{
        return tx_count;
}

/*
 * Function: init ordstat transaction
 * Prepare the ordstat transaction
 */

int
init_ords_tx()
{

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

#ifdef ISO9
#define SQLISO9 "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

#defineSQLTXT"alter session set isolation_level = serializable"


#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, c_id \
                 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"



   int i;
   text stmbuf[SQL_BUF_SIZE];

OCIStmt *curi;

   OCIInitialize(OCI_DEFAULT,(dvoid *)0,0,0,0);
   OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid
**)0);
   OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
   OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SRVRCTXT, errhp);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
      OCI_ATTR_PASSWORD, errhp);
   OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));

   OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_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 */

   octx = (ordctx *) malloc (sizeof(ordctx));
   memset(octx,(char)0,sizeof(ordctx));
   octx->cs = 1;
octx->norow = 0;

/* get the rowid handles */
   for(i=0;i<100;i++) {
    OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&octx->c_rowid_ptr[i],
       OCI_DTYPE_ROWID,0,(dvoid**)0));
  }


#ifdef ISO9
   OCIERROR(errhp,
      OCIHandleAlloc(tpcenv,&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 ISO9
   sprintf((char *) stmbuf, SQLISO9);
   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(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 ISO9

   OCIBND(octx->curo0,octx->w_id_bp,errhp,":w_id",ADR(w_id),SIZ(w_id),SQLT_INT);
   OCIBND(octx->curo0, octx->d_id_bp, errhp,":d_id",ADR(d_id),SIZ(d_id),SQLT_INT);
   OCIBND(octx->curo0, octx->c_id_bp, errhp,":c_id",ADR(c_id),SIZ(c_id),SQLT_INT);
   OCIBND(octx->curo0, octx->byln_bp, errhp,":byln",ADR(bylastname),
SIZ(bylastname),SQLT_INT);
   OCIBND(octx->curo0, octx->c_last_bp, errhp,":c_last",c_last,SIZ(c_last),SQLT_STR);
   OCIBND(octx->curo0, octx->c_first_bp,
errhp,":c_first",c_first,SIZ(c_first),SQLT_STR);
   OCIBND(octx->curo0, octx->c_middle_bp,
errhp,":c_middle",c_middle,SIZ(c_middle),SQLT_STR);
   OCIBND(octx->curo0, octx->c_balance_bp,
errhp,":c_balance",ADR(c_balance),SIZ(c_balance),SQLT_FLT);
   OCIBND(octx->curo0, octx->o_id_bp, errhp,":o_id",ADR(o_id),SIZ(o_id),SQLT_INT);
   OCIBND(octx->curo0, octx->o_entry_d_bp,
errhp,":o_entry_d_base",o_entry_d_base,SIZ(o_entry_d_base),SQLT_DAT);
   OCIBND(octx->curo0, octx->o_cr_id_bp,
errhp,":o_cr_id",ADR(o_carrier_id),SIZ(o_carrier_id),
          SQLT_INT);
   OCIBND(octx->curo0, octx->o_ol_cnt_bp,
errhp,":o_ol_cnt",ADR(o_ol_cnt),SIZ(o_ol_cnt),SQLT_INT);
   OCIBNDRAA(octx->curo0, octx->ol_s_w_id_bp,
errhp,":ol_s_w_id",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",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",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",ol_amount,SIZ(int),SQLT_INT,
          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",ol_delivery_d,SIZ(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(w_id),SIZ(int),SQLT_INT);
   OCIBND(octx->curo0,octx->d_id_bp0,errhp,":d_id",ADR(d_id),SIZ(int),SQLT_INT);
   OCIBND(octx->curo0,octx->c_last_bp,errhp,":c_last",c_last,SIZ(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->middle_cust, sizeof( octx->middle_cust),SQLT_RDD);
   OCIDEF(octx->curo1,octx->c_id_dp,errhp,1,ADR(c_id),SIZ(int),SQLT_INT);
   OCIDEF(octx->curo1,octx->c_balance_dp1,errhp,2,ADR(c_balance),
          SIZ(double),SQLT_FLT);
   OCIDEF(octx->curo1,octx->c_first_dp1,errhp,3,c_first,SIZ(c_first),
          SQLT_STR);
   OCIDEF(octx->curo1,octx->c_middle_dp1,errhp,4,c_middle,
          SIZ(c_middle),SQLT_STR);
   OCIDEF(octx->curo1,octx->o_id_dp1,errhp,5,ADR(o_id),SIZ(int),SQLT_INT);
   OCIDEF(octx->curo1,octx->o_entry_d_dp1,errhp,6,
          o_entry_d_base,SIZ(o_entry_d_base),SQLT_DAT);
   OCIDEF(octx->curo1,octx->o_cr_id_dp1,errhp,7,ADR(o_carrier_id),
          SIZ(int),SQLT_INT);
   OCIDEF(octx->curo1,octx->o_ol_cnt_dp1,errhp,8,ADR(o_ol_cnt),
          SIZ(int),SQLT_INT);
   OCIDEF(octx->curo1,octx->c_last_dp1,errhp,9,c_last,SIZ(c_last), SQLT_STR);

/* Bind for third cursor , no-zero customer id */
   OCIBND(octx->curo2,octx->w_id_bp2,errhp,":w_id",ADR(w_id),SIZ(int),SQLT_INT);
   OCIBND(octx->curo2,octx->d_id_bp2,errhp,":d_id",ADR(d_id),SIZ(int),SQLT_INT);
   OCIBND(octx->curo2,octx->c_id_bp,errhp,":c_id",ADR(c_id),SIZ(int),SQLT_INT);
   OCIDEF(octx->curo2,octx->c_balance_dp2,errhp,1,ADR(c_balance),
          SIZ(double),SQLT_FLT);
   OCIDEF(octx->curo2,octx->c_first_dp2,errhp,2,c_first,SIZ(c_first),
          SQLT_STR);
   OCIDEF(octx->curo2,octx->c_middle_dp2,errhp,3,c_middle,
          SIZ(c_middle),SQLT_STR);
   OCIDEF(octx->curo2,octx->c_last_dp,errhp,4,c_last,SIZ(c_last), SQLT_STR);
   OCIDEF(octx->curo2,octx->o_id_dp2,errhp,5,ADR(o_id),SIZ(int),SQLT_INT);
   OCIDEF(octx->curo2,octx->o_entry_d_dp2,errhp,6,
          o_entry_d_base,SIZ(o_entry_d_base),SQLT_DAT);
   OCIDEF(octx->curo2, octx->o_cr_id_dp2,errhp,7,ADR(o_carrier_id),
          SIZ(int), SQLT_INT);
   OCIDEF(octx->curo2,octx->o_ol_cnt_dp2,errhp,8,ADR(o_ol_cnt),
          SIZ(int),SQLT_INT);
   OCIDEF(octx->curo2,octx->c_id_dp1,errhp,9,ADR(c_id),SIZ(int),SQLT_INT);

/* Bind for last cursor */

   OCIBND(octx->curo3,octx->w_id_bp3,errhp,":w_id",ADR(w_id),SIZ(int),SQLT_INT);
   OCIBND(octx->curo3,octx->d_id_bp3,errhp,":d_id",ADR(d_id),SIZ(int),SQLT_INT);
   OCIBND(octx->curo3,octx->o_id_bp,errhp,":o_id",ADR(o_id),SIZ(int),SQLT_INT);

   OCIDFNRA(octx->curo3, octx->ol_i_id_dp, errhp, 1, 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, 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, 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,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,ol_d_base,7, SQLT_DAT,
            octx->ol_delivery_d_ind,octx->ol_delivery_d_len,
    octx->ol_delivery_d_rcode);
#endif
   return (0);

}

/****************************
 * END BLOCK OF COMMON CODE
```

```
 ****************************/

ordstat_tx(rqst)
TPSVCINFO *rqst;
{
int i;
int execstatus, rcount,errcode;
struct ord_inf *ordstat_p;
ordstat_p = (struct ord_inf *)(rqst->data);

MOVETO(w_id, ordstat_p);
MOVETO(d_id, ordstat_p);
MOVETO(c_id, ordstat_p);

tx_count++;
#if ACID
time(timep);
userlog("ACID ORDSTAT Transaction begun at %s\n", ctime(timep));
#endif


/****************************
 * BEGIN BLOCK OF COMMON CODE
 ****************************/

if (c_id == 0) {
bylastname = 1;
strcpy(c_last, ordstat_p->c_last);
}
else {
bylastname = 0;
c_last[1] = '\0';
}
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(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 ISO9
   OCIERROR(errhp,
      OCIStmtExecute(tpcsvc,octx->curo0,errhp,1,0,0,0,OCI_DEFAULT));
#else
   if(bylastname) {
   execstatus = OCIStmtExecute(tpcsvc,octx->curo0,errhp,100,0,0,0,OCI_DEFAULT);
     if (execstatus != OCI_NO_DATA)    /* will get OCI_NO_DATA if <100 found */
     {
         OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
         errcode = OCIERROR(errhp,execstatus);
         if(errcode == NOT_SERIALIZABLE) {
             goto retry;
         } else if (errcode == RECOVERR) {
             goto retry;
         } else {
           return -1;
         }
     }
      /* get rowcount, find middle one */
     OCIAttrGet(octx->curo0,OCI_HTYPE_STM,&rcount,NULL,OCI_ATTR_ROWCNT,errhp);

      /*check if rowcount = 0*/
     if ( rcount == 0 ) return -1;

     octx->cust_idx=(rcount-1)/2 ;
     octx->middle_cust = octx->c_rowid_ptr[octx->cust_idx];
     execstatus = OCIStmtExecute(tpcsvc,octx->curo1,errhp,1,0,0,0,OCI_DEFAULT);
     if (execstatus != OCI_SUCCESS)
     {
         OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
         errcode = OCIERROR(errhp,execstatus);
         if(errcode == NOT_SERIALIZABLE) {
             goto retry;
         } else if (errcode == RECOVERR) {
             goto retry;
         } else {
           return -1;
         }
     }
   } else {
     execstatus = OCIStmtExecute(tpcsvc,octx->curo2,errhp,1,0,0,0,OCI_DEFAULT);
     if (execstatus != OCI_SUCCESS)
     {
         OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
         errcode = OCIERROR(errhp,execstatus);
         if(errcode == NOT_SERIALIZABLE) {
             goto retry;
         } else if (errcode == RECOVERR) {
             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);

   execstatus = OCIStmtExecute(tpcsvc,octx->curo3,errhp,o_ol_cnt,0,0,0,
                        OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
   if (execstatus != OCI_SUCCESS)
     {
         OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
         errcode = OCIERROR(errhp,execstatus);
         if(errcode == NOT_SERIALIZABLE) {
             goto retry;
         } else if (errcode == RECOVERR) {
             goto retry;
         } else {
           return -1;
         }
     }
#ifdef NOTMORE
   OCIERROR(errhp,
     OCITransCommit(tpcsvc,errhp,OCI_DEFAULT));
#endif

for (i = 0; i < o_ol_cnt; i++) {
ordstat_p->o_items[i].ol_supply_w_id = ol_supply_w_id[i];
ordstat_p->o_items[i].ol_i_id = ol_i_id[i];
ordstat_p->o_items[i].ol_quantity = ol_quantity[i];
ordstat_p->o_items[i].ol_amount = ((double)ol_amount[i]) / 100;
if ( octx->ol_delivery_d_ind[i] == -1)/* null date in field */
strncpy(ordstat_p->o_items[i].ol_delivery_d,"01-01-1811",10);
else
cvtdmy(ol_d_base[i], ordstat_p->o_items[i].ol_delivery_d);
}
#endif /* ISO9 */
cvtdmyhms(o_entry_d_base, ordstat_p->o_entry_d);

/****************************
 * END BLOCK OF COMMON CODE
 ****************************/

#if ACID
time(timep);
userlog("ACID ORDSTAT for w_id = %d, d_id = %d, c_id = %d, o_id = %d\n",
w_id, d_id, c_id, o_id);
userlog("ACID ORDSTAT Transaction completed at %s\n", ctime(timep));
#endif

MOVEBACK(o_id, ordstat_p);
MOVEBACK(o_carrier_id, ordstat_p);
MOVEBACK(o_ol_cnt, ordstat_p);
MOVEBACK(c_balance, ordstat_p);
MOVECBACK(c_first, 16, ordstat_p);
MOVECBACK(c_middle, 2, ordstat_p);
MOVECBACK(c_last, 16, ordstat_p);
/***** Copied earlier
MOVECBACK(o_entry_d, 19, ordstat_p);
****/
/* for search by clastname
 */
 MOVEBACK(c_id, ordstat_p);
return(0);
}


void
cleanup(code)
{
if (octx)
free(octx);

   /* log off */

   OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
   OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
   OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
   OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
   OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);

    exit(code);
}

tpsvrinit(argc, argv)
char **argv;
{
        if (init_ords_tx())             /* Prepare transaction */
                return(1);
        else
                return(0);
}

void
tpsvrdone()
{
   /*oclose (&curo0);
   oclose (&curo1);
   oclose (&curo2); */

   /* log off */
   /*ologof (&tpclda);*/
}

ORDS(rqst)
TPSVCINFO *rqst;
{
        if (ordstat_tx(rqst) )
                tpreturn(TPFAIL, 0, rqst->data, sizeof(struct ord_inf), 0);
        else
                tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct ord_inf), 0);
}
```

tuxserver/tpcc_srv_paym.c

```
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
```

```
 */

#pragma ident "@(#)tpcso_srv_paym.c1.1797/01/02SMI"

/*+=================================================================+
 |        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 "ora_oci.h"

#include <signal.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

#include "./ora_err.h"

static int tx_count = 0;

/*#include "proc_stat.h" */

#define SQLTXT1 "alter session set isolation_level = serializable"

#define MOVETO(element, struct_name) \
element = struct_name -> element
#define MOVEBACK(element, struct_name) \
struct_name -> element = element
#define MOVECTO(element, cnt, struct_name) {\
inti;\
strncpy(element, struct_name -> element, cnt); \
element[cnt] = '\0';\
for(i=0; i<=cnt; i++)\
{\
if(isspace(element[i]))\
{\
element[i] = '\0';\
break;\
}\
}\
}
#define MOVECBACK(element, cnt, struct_name) \
strncpy(struct_name -> element, element, cnt)
struct pay_inf {
intw_id;
intd_id;
intc_id;
intc_w_id;
intc_d_id;
double h_amount;
double c_credit_lim;
double c_balance;
double c_discount;
char h_date[20];
char w_street_1[21];
char w_street_2[21];
char w_city[21];
char w_state[3];
char w_zip[11];
char d_street_1[21];
char d_street_2[21];
char d_city[21];
char d_state[3];
char d_zip[11];
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[11];
char c_phone[17];
char c_since[11];
char c_credit[3];
char c_data_1[51];
char c_data_2[51];
char c_data_3[51];
char c_data_4[51];
};


#if ACID
#include <sys/types.h>
#include <time.h>
time_t curtime;
time_t *timep = &curtime;
#endif


/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

unsigned char cr_date[7];
unsigned char c_since[7];

/* List of fields in payment */

int retry;
char c_data[201];

intw_id;
intd_id;
intc_id, bylastname;
intc_w_id;
intc_d_id;
int h_amount;
int c_credit_lim;
double c_balance;
int c_discount;
char h_date[20];
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_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_d[11];
char c_credit[3];
int retries;

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;



payctx *pctx;

/****************************
 * END BLOCK OF COMMON CODE
 ****************************/


get_paym_tx_cnt()
{
        return tx_count;
}


/*
 * Function: init payment transaction
 * Prepare the payment transaction
 */

/****************************
 * BEGIN BLOCK OF COMMON CODE
 ****************************/

OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
char *uid = "dbbench";
char *pwd = "dbbench";

int
init_paym_tx()
{
#defineSQLTXT"alter session set isolation_level = serializable"
#defineSQLTXT_INIT"BEGIN pay.pay_init; END;"

text stmbuf[SQL_BUF_SIZE];

   OCIStmt *curi;

   OCIInitialize(OCI_DEFAULT,(dvoid *)0,0,0,0);
   OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid
**)0);
   OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
   OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SRVRCTXT, errhp);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
      OCI_ATTR_PASSWORD, errhp);
   OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));

   OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_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 */

   pctx = (payctx *)malloc(sizeof(payctx));
   memset(pctx,(char)0,sizeof(payctx));

/* 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
   sqlfile("paynz_abort.sql",stmbuf);
#else
   sqlfile("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
   sqlfile("payz_abort.sql",stmbuf);
#else
   sqlfile("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(w_id);
  pctx->d_id_ind = TRUE;
  pctx->d_id_len = SIZ(d_id);
  pctx->c_w_id_ind = TRUE;
  pctx->c_w_id_len = SIZ(c_w_id);
  pctx->c_d_id_ind = TRUE;
  pctx->c_d_id_len = SIZ(c_d_id);
  pctx->c_id_ind = TRUE;
  pctx->c_id_len = 0;
  pctx->h_amount_len = SIZ(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(w_id),SIZ(int),
          SQLT_INT, &pctx->w_id_ind, NULL, NULL);
  OCIBNDR(pctx->curp0, pctx->d_id_bp, errhp,":d_id",ADR(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(c_w_id),SIZ(int),
          SQLT_INT);
  OCIBND(pctx->curp0, pctx->c_d_id_bp, errhp,":c_d_id",ADR(c_d_id),SIZ(int),
          SQLT_INT);
  OCIBND(pctx->curp0, pctx->c_id_bp, errhp,":c_id",ADR(c_id),SIZ(int),
          SQLT_INT);
  OCIBNDR(pctx->curp0, pctx->h_amount_bp, errhp,":h_amount",ADR(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",c_last,SIZ(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",w_street_1,
          SIZ(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",w_street_2,
          SIZ(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",w_city,SIZ(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",w_state,SIZ(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",w_zip,SIZ(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",d_street_1,
          SIZ(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",d_street_2,
          SIZ(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",d_city,SIZ(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",d_state,SIZ(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",d_zip,SIZ(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",c_first,SIZ(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",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",c_street_1,
          SIZ(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",c_street_2,
          SIZ(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",c_city,SIZ(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",c_state,SIZ(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",c_zip,SIZ(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",c_phone,SIZ(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",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",c_credit,
          SIZ(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(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(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(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",c_data,SIZ(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",h_date,SIZ(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(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(cr_date),
          SIZ(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(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(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(c_w_id),SIZ(int),
          SQLT_INT);
  OCIBND(pctx->curp1, pctx->c_d_id_bp1, errhp,":c_d_id",ADR(c_d_id),SIZ(int),
          SQLT_INT);
  OCIBNDR(pctx->curp1, pctx->c_id_bp1, errhp,":c_id",ADR(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(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",c_last,SIZ(c_last),
          SQLT_STR);
  OCIBNDR(pctx->curp1, pctx->w_street_1_bp1, errhp,":w_street_1",w_street_1,
          SIZ(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",w_street_2,
          SIZ(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",w_city,SIZ(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",w_state,SIZ(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",w_zip,SIZ(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",d_street_1,
          SIZ(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",d_street_2,
          SIZ(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",d_city,SIZ(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",d_state,
          SIZ(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",d_zip,SIZ(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",c_first,
            SIZ(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",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",c_street_1,
            SIZ(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",c_street_2,
            SIZ(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",c_city,SIZ(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",c_state,SIZ(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",c_zip,SIZ(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",c_phone,SIZ(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",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",c_credit,
            SIZ(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(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(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(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",c_data,SIZ(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",h_date,SIZ(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(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(cr_date),
            SIZ(cr_date),SQLT_DAT, &pctx->cr_date_ind, &pctx->cr_date_len,
            &pctx->cr_date_rc);

return(0);
}

int execstatus,errcode;

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/


payment_tx(rqst)
TPSVCINFO *rqst;
{

struct pay_inf *payment_p;
        payment_p = (struct pay_inf *)(rqst->data);

MOVETO(w_id, payment_p);
MOVETO(d_id, payment_p);
MOVETO(c_id, payment_p);
MOVETO(c_w_id, payment_p);
MOVETO(c_d_id, payment_p);

h_amount = (int)(payment_p->h_amount * 100);
strcpy(c_last, payment_p->c_last);
tx_count++;


#if ACID
time(timep);
userlog("ACID PAYMENT Transaction Begun at %s\n", ctime(timep));
#endif
/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/
retryp:
vgetdate(cr_date);
  pctx->w_id_ind = TRUE;
  pctx->w_id_len = SIZ(w_id);
  pctx->d_id_ind = TRUE;
  pctx->d_id_len = SIZ(d_id);
  pctx->c_w_id_ind = TRUE;
  pctx->c_w_id_len = SIZ(c_w_id);
  pctx->c_d_id_ind = TRUE;
  pctx->c_d_id_len = SIZ(c_d_id);
  pctx->c_id_ind = TRUE;
  pctx->c_id_len = SIZ(c_id);
  pctx->h_amount_ind = TRUE;
  pctx->h_amount_len = SIZ(h_amount);
  pctx->c_last_ind = TRUE;
  pctx->c_last_len = SIZ(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 (c_id == 0) {
bylastname = 1;
    execstatus=OCIStmtExecute(tpcsvc,pctx->curp1,errhp,1,0,0,0,OCI_DEFAULT);
  } else {
bylastname = 0;
    execstatus=OCIStmtExecute(tpcsvc,pctx->curp0,errhp,1,0,0,0,OCI_DEFAULT);
  }
  if (execstatus != OCI_SUCCESS)
     {
         OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
         errcode = OCIERROR(errhp,execstatus);
         if(errcode == NOT_SERIALIZABLE) {
             retries++;
             goto retryp;
         } else if (errcode == RECOVERR) {
             retries++;
             goto retryp;
         } else {
           return -1;
         }
     }
cvtdmyhms(cr_date, h_date);
cvtdmy(c_since, c_since_d);

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/

MOVEBACK(c_id, payment_p);
payment_p->c_credit_lim = ((double)c_credit_lim) / 100;
payment_p->c_discount = ((double)c_discount) / 100;
payment_p->c_balance = c_balance / 100; /* convert to dollars & cents */
#if ACID
time(timep);
userlog("w_id %d, d_id %d, c_id %d, h_amount = %d, c_balance = %f\n",
w_id, d_id, c_id, h_amount, c_balance);
userlog("ACID PAYMENT Transaction completed at %s\n", ctime(timep));
#endif
strcpy(payment_p->c_since, c_since_d);
MOVECBACK(h_date, 20, payment_p);
MOVECBACK(w_street_1, 21, payment_p);
MOVECBACK(w_street_2, 21, payment_p);
MOVECBACK(w_city, 21, payment_p);
MOVECBACK(w_state, 3, payment_p);
MOVECBACK(w_zip, 11, payment_p);
MOVECBACK(d_street_1, 21, payment_p);
MOVECBACK(d_street_2, 21, payment_p);
MOVECBACK(d_city, 21, payment_p);
MOVECBACK(d_state, 3, payment_p);
MOVECBACK(d_zip, 11, payment_p);
MOVECBACK(c_first, 17, payment_p);
MOVECBACK(c_middle, 3, payment_p);
MOVECBACK(c_last, 17, payment_p);
MOVECBACK(c_street_1, 21, payment_p);
MOVECBACK(c_street_2, 21, payment_p);
MOVECBACK(c_city, 21, payment_p);
MOVECBACK(c_state, 3, payment_p);
MOVECBACK(c_zip, 11, payment_p);
MOVECBACK(c_phone, 17, payment_p);
MOVECBACK(c_credit, 3, payment_p);
strncpy(payment_p->c_data_1, c_data, 50);
strncpy(payment_p->c_data_2, c_data+50, 50);
strncpy(payment_p->c_data_3, c_data+100, 50);
strncpy(payment_p->c_data_4, c_data+150, 50);

return(0);
}

/* Tuxedo code */
tpsvrinit(argc, argv)
char **argv;
{
        return(init_paym_tx());         /* Prepare transaction */
```

```
}

void
tpsvrdone()
{
}


PAYM(rqst)
TPSVCINFO *rqst;
{
        if (payment_tx(rqst) )
                tpreturn(TPFAIL, 0, rqst->data, sizeof(struct pay_inf), 0);
        else
                tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct pay_inf), 0);
}
```

tuxserver/tpcc_srv_stock.c

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcso_srv_stock.c1.695/04/12SMI"

/*+=================================================================+
 |        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 "ora_oci.h"

#include <signal.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "./ora_err.h"
/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

static int tx_count = 0;

#define MOVETO(element, struct_name) element = struct_name -> element
#define MOVEBACK(element, struct_name) struct_name -> element = element

/* List of fields in stock */
/* This structure should be EXACTLY identical to the one declared in client.h */
/* List of fields in stock-level */

struct stock_inf {
intw_id;
int d_id;
int threshold;
int low_stock;
};


/*struct msgh_req message;*/
int my_qid, my_id;
char my_name[] = "Stock";

#define SQLTXT "SELECT  /*+ nocache(stock) */ \
count (DISTINCT s_i_id) \
            FROM order_line, stock, district \
            WHERE d_id = :d_id AND d_w_id = :w_id AND \
                  d_id = ol_d_id AND d_w_id = ol_w_id AND \
                  ol_i_id = s_i_id AND ol_w_id = s_w_id AND \
                  s_quantity < :threshold AND \
                  ol_o_id BETWEEN (d_next_o_id - 20) AND (d_next_o_id - 1)"


#define SQLTXTTEST "BEGIN stocklevel.getstocklevel (:w_id, :d_id, \
:threshold); END;"


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;
stoctx *sctx;

int w_id;
int d_id;
int threshold;
int low_stock;

/*
 * Initialize transaction
 */
get_stock_tx_cnt()
{
        return tx_count;
}

OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
char *uid = "dbbench";
char *pwd = "dbbench";

int
init_stock_tx()
{

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

   text stmbuf[SQL_BUF_SIZE];
OCIStmt *curi;

   OCIInitialize(OCI_DEFAULT,(dvoid *)0,0,0,0);
   OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid
**)0);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid
**)0);
   OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
   OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SRVRCTXT, errhp);
   OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
   OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
      OCI_ATTR_PASSWORD, errhp);
   OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));

   OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_USERCTXT, errhp);

#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 */

   sctx = (stoctx *)malloc(sizeof(stoctx));
   memset(sctx,(char)0,sizeof(stoctx));
   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(w_id),sizeof(int),
          SQLT_INT);
   OCIBND(sctx->curs,sctx->d_id_bp,errhp, ":d_id", ADR(d_id),sizeof(int),
      SQLT_INT);
   OCIBND(sctx->curs,sctx->threshold_bp,errhp, ":threshold", ADR(threshold),
      sizeof(int),SQLT_INT);
   OCIDEFINE(sctx->curs,sctx->low_stock_bp,errhp, 1, ADR(low_stock),
      sizeof(int), SQLT_INT);

#ifdef TKPROF
EXEC SQL ALTER SESSION SET SQL_TRACE = TRUE;
#endif

/*proc_stat_msg("init_stock_tx()\n");
proc_stat(); */

return(0);

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/
}


/*
 * Function: do stocklevel transaction
 * Input is the stocklevel structure. Output is low_stock field
 */
stocklevel_tx(rqst)
TPSVCINFO *rqst;
{

int err, execstatus,errcode;
        struct stock_inf *stocklevel_p;
        stocklevel_p = (struct stock_inf *)(rqst->data);

MOVETO(w_id, stocklevel_p);
MOVETO(d_id, stocklevel_p);
MOVETO(threshold, stocklevel_p);

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

tx_count++;
retry:
  execstatus=OCIStmtExecute(tpcsvc,sctx->curs,errhp,1,0,0,0,OCI_COMMIT_ON_SUCCESS);
  if (execstatus != OCI_SUCCESS)
     {
        OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
```

```
        errcode = OCIERROR(errhp,execstatus);
        if(errcode == NOT_SERIALIZABLE) {
            goto retry;
        } else if (errcode == RECOVERR) {
            goto retry;
        } else {
          return -1;
        }
     }

/****************************
 * END BLOCK OF COMMON CODE
 ****************************/

MOVEBACK(low_stock, stocklevel_p);
        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct stock_inf), 0);
return(0);
}


void
cleanup(code)
{
   /* log off */

   OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
   OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
   OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
   OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
   OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);

if (sctx) free (sctx);
    if (my_qid >= 0)
        msgctl(my_qid, IPC_RMID, 0);
    userlog("Stock %d: Exiting\n", my_id);
    exit(code);
}

/* Tuxedo */
tpsvrinit(argc, argv)
char **argv;
{
        return(init_stock_tx());                /* Prepare transaction */
}

void
tpsvrdone()
{
}


STOCK(rqst)
TPSVCINFO *rqst;
{
        stocklevel_tx(rqst);
}
```

tuxserver/tpcc_srv_util.c

```
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcso_srv_util.c1.1797/01/02SMI"

/*+=================================================================+
  |        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
  |                    OPEN SYSTEMS PERFORMANCE GROUP               |
  |                         All Rights Reserved                     |
  +=================================================================*/
/* Common utility functions used by all tpcso_srv* programs */

#include <stdio.h>
#include <sys/types.h>
#include <sys/file.h>

#include "ora_oci.h"
#include "ora_err.h"

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 *oradt)
{
 struct tm *loctime;
 time_t  int_time;

 struct ORADATE {
 unsigned charcentury;
 unsigned charyear;
 unsigned charmonth;
 unsigned charday;
 unsigned charhour;
 unsigned charminute;
 unsigned charsecond;
 } Date;
int century;
int cnvrtOK;

/* assume convert is successful */
cnvrtOK = 1;

/* get the current date and time as an integer */
time( &int_time);

/* Convert the current date and time into local time */
loctime = localtime( &int_time);

century = (1900+loctime->tm_year) / 100;

Date.century = (unsigned char)(century + 100);
if (Date.century < 119 || Date.century > 120) cnvrtOK = 0;
Date.year  = (unsigned char)(loctime->tm_year+100);
if (Date.year < 100 || Date.year > 199) cnvrtOK = 0;
Date.month = (unsigned char)(loctime->tm_mon + 1);
if (Date.month < 1 || Date.month > 12) cnvrtOK = 0;
Date.day   = (unsigned char)loctime->tm_mday;
if (Date.day < 1 || Date.day > 31) cnvrtOK = 0;
Date.hour  = (unsigned char)(loctime->tm_hour + 1);
if (Date.hour < 1 || Date.hour > 24) cnvrtOK = 0;
Date.minute= (unsigned char)(loctime->tm_min + 1);
if (Date.minute < 1 || Date.minute > 60) cnvrtOK = 0;
Date.second= (unsigned char)(loctime->tm_sec + 1);
if (Date.second < 1 || Date.second > 60) cnvrtOK = 0;


if (cnvrtOK)
  memcpy(oradt,&Date,7);
  else
    *oradt = '\0';

 return;

}


void cvtdmy (unsigned char *oradt, char *outdate)
{

        struct ORADATE {
                unsigned char   century;
                unsigned char   year;
                unsigned char   month;
                unsigned char   day;
                unsigned char   hour;
                unsigned char   minute;
                unsigned char   second;
        } Date;


        int day,month,year;

        memcpy(&Date,oradt,7);

        year = (Date.century-100)*100 + Date.year-100;
        month = Date.month;
        day = Date.day;
        sprintf(outdate,"%02d-%02d-%4d",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",
                        day,month,year,hour,min,sec);

        return;
}
```

tuxserver/blocks/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;
```

tuxserver/blocks/paynz.sql

```
DECLARE /* paynz */
--      cust_rowid              ROWID;
--      dist_name               VARCHAR2(11);
--      ware_name               VARCHAR2(11);
      not_serializable        EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
      deadlock                EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old        EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
   BEGIN
      LOOP BEGIN
            UPDATE warehouse
               SET w_ytd = w_ytd + :h_amount
             WHERE w_id = :w_id
        RETURNING w_name, w_street_1, w_street_2, w_city, w_state, w_zip
             INTO pay.ware_name, :w_street_1, :w_street_2, :w_city,
                  :w_state, :w_zip;

          UPDATE  customer
             SET  c_balance = c_balance - :h_amount,
                  c_ytd_payment = c_ytd_payment + :h_amount,
                  c_payment_cnt = c_payment_cnt+1
           WHERE  c_id = :c_id AND c_d_id = :c_d_id AND
                  c_w_id = :c_w_id
        RETURNING rowid, c_first, c_middle, c_last, c_street_1,
                  c_street_2, c_city, c_state, c_zip, c_phone,
                  c_since, c_credit, c_credit_lim,
                  c_discount, c_balance
             INTO 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/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;


         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;
           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;
```

tuxserver/blocks/payz.sql

```
DECLARE /* payz */
      not_serializable        EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
      deadlock                EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old        EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
   BEGIN
      LOOP BEGIN
         UPDATE warehouse
            SET w_ytd = w_ytd+:h_amount
            WHERE w_id = :w_id
            RETURNING w_name,
                      w_street_1, w_street_2, w_city, w_state, w_zip
              INTO pay.ware_name,
                   :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


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

--Store number of rows processed
       pay.c_num := sql%rowcount;
       pay.cust_rowid := pay.row_id((pay.c_num) / 2);

        UPDATE customer
          SET c_balance = c_balance - :h_amount,
              c_ytd_payment = c_ytd_payment+ :h_amount,
              c_payment_cnt = c_payment_cnt+1
        WHERE rowid = 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;

        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);

--Sanjay-No commit needed iff Commit on Success done
--       COMMIT;
        COMMIT;
        EXIT;

        EXCEPTION
           WHEN not_serializable OR deadlock OR snapshot_too_old THEN
              ROLLBACK;
              :retry := :retry + 1;
        END;

     END LOOP;
   END;
```

tuxserver/blocks/initnew.sql

```
-- The initnew package for storing variables used in the
-- New Order anonymous block
```

```
CREATE OR REPLACE PACKAGE initnew
AS
 TYPE intarray IS TABLE OF INTEGER index by binary_integer;
 TYPE distarray IS TABLE OF VARCHAR(24) index by binary_integer;
 nulldate       DATE;
 s_distdistarray;
 idx1arrintarray;
 s_remoteintarray;
 PROCEDURE new_init(idxarr intarray);
END initnew;
/
show errors;

CREATE OR REPLACE PACKAGE BODY initnew AS
  PROCEDURE new_init (idxarr  intarray)
  IS
  BEGIN
        -- initialize null date
   nulldate := TO_DATE('01-01-1811', 'MM-DD-YYYY');
    idx1arr := idxarr;
  END new_init;
END initnew;
/
show errors
exit
```

tuxserver/blocks/pnew.sql

```
-- New Order Anonymous block

  DECLARE
      idx                       BINARY_INTEGER;
      dummy_local               BINARY_INTEGER;
      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);
   PROCEDURE u1 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_01,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
:brand_generic;

   END u1;

   PROCEDURE u2 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_02,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
                          :brand_generic;
   END u2;

   PROCEDURE u3 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_03,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
                          :brand_generic;
   END u3;

   PROCEDURE u4 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_04,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
                          :brand_generic;
   END u4;

   PROCEDURE u5 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_05,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
                          :brand_generic;
   END u5;

   PROCEDURE u6 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_06,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
                          :brand_generic;
   END u6;

   PROCEDURE u7 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_07,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
                          :brand_generic;
   END u7;

   PROCEDURE u8 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_08,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
                          :brand_generic;
   END u8;

   PROCEDURE u9 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_09,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
                          :brand_generic;
   END u9;

   PROCEDURE u10 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
```

```
         AND s_w_id = :ol_supply_w_id(idx)
         RETURNING i_price, i_name, s_quantity, s_dist_10,
  DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                  DECODE(instr(s_data,'ORIGINAL'), 0, 'G', 'B'))
 BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
                        :brand_generic;
   END u10;

   PROCEDURE fix_items IS
     rows_lost                    BINARY_INTEGER;
     max_index                    BINARY_INTEGER;
     temp_index                   BINARY_INTEGER;
   BEGIN
-- gotta shift price, name, s_quantity, brand_generic, s_dist, ol_amount
       idx := 1;
-- found 0 bad rows
       rows_lost := 0;
-- so many rows in out array to begin with
max_index := sql%rowcount;

        WHILE (max_index != :o_ol_cnt) LOOP

-- find item where item ids dont match
  WHILE (idx <= sql%rowcount AND
                sql%bulk_rowcount(idx + rows_lost) = 1)
         LOOP
    idx := idx + 1;
         END LOOP;

-- shift the items please
         temp_index := max_index;
         WHILE (temp_index >= idx + rows_lost) LOOP
    :i_price(temp_index + 1)       :=  :i_price(temp_index);
           :i_name(temp_index + 1)        :=  :i_name(temp_index);
    :s_quantity(temp_index + 1)    :=  :s_quantity(temp_index);
           initnew.s_dist(temp_index + 1) :=  initnew.s_dist(temp_index);
    :brand_generic(temp_index + 1) :=  :brand_generic(temp_index);
    temp_index := temp_index - 1;
         END LOOP;

--  values for the non-existent items if not at end
         IF (idx + rows_lost <= :o_ol_cnt) THEN
    :i_price(idx + rows_lost)     :=   0;
    :i_name(idx + rows_lost)      :=   NULL;
           :s_quantity(idx + rows_lost)  :=   0;
           initnew.s_dist(idx + rows_lost) := NULL;
           :brand_generic(idx + rows_lost) := NULL;

--  one more bad row
           rows_lost := rows_lost + 1;
           max_index := max_index + 1;
   END IF;

        END LOOP;
   END fix_items;

   BEGIN
      LOOP BEGIN
         UPDATE district SET d_next_o_id = d_next_o_id + 1
          WHERE d_id = :d_id AND  d_w_id = :w_id
          RETURNING d_tax, d_next_o_id-1
          INTO :d_tax, :o_id;

         SELECT c_discount, c_last, c_credit, w_tax
          INTO :c_discount, :c_last, :c_credit , :w_tax
          FROM customer , warehouse
          WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id
          AND  w_id = :w_id;


         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_d_id, o_w_id, o_c_id, o_entry_d,
                               o_carrier_id, o_ol_cnt, o_all_local)
          VALUES (:o_id, :d_id, :w_id, :c_id,
                  :cr_date, 11, :o_ol_cnt, :o_all_local);

         dummy_local :=  :d_id;

         IF (dummy_local = 1) THEN u1; END IF;

         IF (dummy_local = 2) THEN u2; END IF;

         IF (dummy_local = 3) THEN u3; END IF;

         IF (dummy_local = 4) THEN u4; END IF;

         IF (dummy_local = 5) THEN u5; END IF;

         IF (dummy_local = 6) THEN u6; END IF;

         IF (dummy_local = 7) THEN u7; END IF;

         IF (dummy_local = 8) THEN u8; END IF;

         IF (dummy_local = 9) THEN u9; END IF;

         IF (dummy_local = 10) THEN u10; END IF;

         dummy_local := sql%rowcount;


-- fix the rows if necessary
         IF (dummy_local != :o_ol_cnt )  THEN fix_items; END IF;

-- calculate ol_amount

        FOR idx IN 1 ..:o_ol_cnt LOOP
         :ol_amount(idx):=:ol_quantity(idx)*:i_price(idx);
        END LOOP;

        FORALL idx IN 1..:o_ol_cnt
         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 (:o_id, :d_id, :w_id, initnew.idx1arr(idx), initnew.nulldate,
                  :ol_i_id(idx), :ol_supply_w_id(idx),
          :ol_quantity(idx), :ol_amount(idx), initnew.s_dist(idx));

IF (dummy_local != :o_ol_cnt) THEN
           :o_ol_cnt := dummy_local;
           ROLLBACK;
         END IF;

         EXIT;

         EXCEPTION
            WHEN not_serializable OR deadlock OR snapshot_too_old THEN
                ROLLBACK;
                :retry := :retry + 1;
          END;
       END LOOP;
    END;
```

tuxserver/blocks/views.sql

```
create or replace view wh_cust
(w_id, w_tax, c_id, c_d_id, c_w_id, c_discount, c_last, c_credit)
as select w.w_id, w.w_tax,
          c.c_id, c.c_d_id, c.c_w_id, c.c_discount, c.c_last, c.c_credit
   from customer c, warehouse w
   where w.w_id = c.c_w_id
/

create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
   from district d, warehouse w
   where w.w_id = d.d_w_id
/

create  or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
 select i.i_id, s_w_id, i.i_price, i.i_name, i.i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
  from stock s, item i
 where i.i_id = s.s_i_id
/
exit
```

Appendix B. Database Design

```
build_dir/ddl
==> createdb.ddl <==

CONNECT internal

CREATE DATABASE tpcc
  CONTROLFILE REUSE
  DATAFILE '/oradev/sys000' size 100M REUSE
  LOGFILE '/oradev/node1_log1' size 19999M REUSE,  '/oradev/node1_log2' size 19999M
REUSE
  MAXDATAFILES 750
  MAXINSTANCES  4
  MAXLOGFILES  20
;

CREATE ROLLBACK SEGMENT r01
  STORAGE (INITIAL 500K
           MINEXTENTS 2
           NEXT 500K);


CREATE ROLLBACK SEGMENT r02
  STORAGE (INITIAL 500K
           MINEXTENTS 2
           NEXT 500K);


CREATE ROLLBACK SEGMENT r03
  STORAGE (INITIAL 500K
           MINEXTENTS 2
           NEXT 500K);


CREATE ROLLBACK SEGMENT r04
  STORAGE (INITIAL 500K
           MINEXTENTS 2
           NEXT 500K);


CREATE ROLLBACK SEGMENT r05
  STORAGE (INITIAL 500K
           MINEXTENTS 2
           NEXT 500K);


CREATE ROLLBACK SEGMENT r06
  STORAGE (INITIAL 500K
           MINEXTENTS 2
           NEXT 500K);


CREATE ROLLBACK SEGMENT r07
  STORAGE (INITIAL 500K
           MINEXTENTS 2
           NEXT 500K);


CREATE ROLLBACK SEGMENT r08
  STORAGE (INITIAL 500K
           MINEXTENTS 2
           NEXT 500K);


CREATE ROLLBACK SEGMENT r09
  STORAGE (INITIAL 500K
           MINEXTENTS 2
           NEXT 500K);


@?/rdbms/admin/catalog;
@?/rdbms/admin/catexp.sql
@?/rdbms/admin/catldr.sql
@?/rdbms/admin/catproc.sql
@?/rdbms/admin/catparr
DISCONNECT;

CONNECT internal;
@?/rdbms/admin/utlmontr;

CREATE USER dbbench
  IDENTIFIED BY dbbench
  ;

GRANT CONNECT, RESOURCE TO dbbench;
alter user dbbench temporary tablespace temp;

==> addlogs.ddl <==

for n in 2 3 4 ; do
(
echo connect internal
echo ALTER DATABASE ADD LOGFILE THREAD $n
echo \'/oradev/node${n}_log1\' size 19999M REUSE,
echo \'/oradev/node${n}_log2\' size 19999M REUSE\;
echo alter database enable thread $n \;
) | nohup timex svrmgrl  > $0.out.$n 2>&1 &
sleep 30
done

==> tspaces.ddl <==
connect system/manager


create tablespace roll_N0 datafile '/oradev/roll000' size 199M
extent management local
uniform size 100K nologging;


create tablespace roll_N1 datafile '/oradev/roll001' size 199M
extent management local
uniform size 100K nologging;


create tablespace roll_N2 datafile '/oradev/roll002' size 199M
extent management local
uniform size 100K nologging;


create tablespace roll_N3 datafile '/oradev/roll003' size 199M
extent management local
uniform size 100K nologging;


create tablespace item datafile '/oradev/item000' size 60M
extent management local
uniform size 59M nologging;
alter tablespace item add datafile '/oradev/item001' size 60M ;
alter tablespace item add datafile '/oradev/item002' size 60M ;
alter tablespace item add datafile '/oradev/item003' size 60M ;


create tablespace cust datafile '/oradev/cust000' size 4830M
extent management local
uniform size 4829M nologging;
alter tablespace cust add datafile '/oradev/cust001' size 4830M ;
alter tablespace cust add datafile '/oradev/cust002' size 4830M ;
alter tablespace cust add datafile '/oradev/cust003' size 4830M ;
alter tablespace cust add datafile '/oradev/cust004' size 4830M ;
alter tablespace cust add datafile '/oradev/cust005' size 4830M ;
alter tablespace cust add datafile '/oradev/cust006' size 4830M ;
alter tablespace cust add datafile '/oradev/cust007' size 4830M ;
alter tablespace cust add datafile '/oradev/cust008' size 4830M ;
alter tablespace cust add datafile '/oradev/cust009' size 4830M ;
alter tablespace cust add datafile '/oradev/cust010' size 4830M ;
alter tablespace cust add datafile '/oradev/cust011' size 4830M ;
alter tablespace cust add datafile '/oradev/cust012' size 4830M ;
alter tablespace cust add datafile '/oradev/cust013' size 4830M ;
alter tablespace cust add datafile '/oradev/cust014' size 4830M ;
alter tablespace cust add datafile '/oradev/cust015' size 4830M ;
alter tablespace cust add datafile '/oradev/cust016' size 4830M ;
alter tablespace cust add datafile '/oradev/cust017' size 4830M ;
alter tablespace cust add datafile '/oradev/cust018' size 4830M ;
alter tablespace cust add datafile '/oradev/cust019' size 4830M ;
alter tablespace cust add datafile '/oradev/cust020' size 4830M ;
alter tablespace cust add datafile '/oradev/cust021' size 4830M ;
alter tablespace cust add datafile '/oradev/cust022' size 4830M ;
alter tablespace cust add datafile '/oradev/cust023' size 4830M ;
alter tablespace cust add datafile '/oradev/cust024' size 4830M ;
alter tablespace cust add datafile '/oradev/cust025' size 4830M ;
alter tablespace cust add datafile '/oradev/cust026' size 4830M ;
alter tablespace cust add datafile '/oradev/cust027' size 4830M ;
alter tablespace cust add datafile '/oradev/cust028' size 4830M ;
alter tablespace cust add datafile '/oradev/cust029' size 4830M ;
alter tablespace cust add datafile '/oradev/cust030' size 4830M ;
alter tablespace cust add datafile '/oradev/cust031' size 4830M ;
alter tablespace cust add datafile '/oradev/cust032' size 4830M ;
alter tablespace cust add datafile '/oradev/cust033' size 4830M ;
alter tablespace cust add datafile '/oradev/cust034' size 4830M ;
alter tablespace cust add datafile '/oradev/cust035' size 4830M ;
alter tablespace cust add datafile '/oradev/cust036' size 4830M ;
alter tablespace cust add datafile '/oradev/cust037' size 4830M ;
alter tablespace cust add datafile '/oradev/cust038' size 4830M ;
alter tablespace cust add datafile '/oradev/cust039' size 4830M ;
alter tablespace cust add datafile '/oradev/cust040' size 4830M ;
alter tablespace cust add datafile '/oradev/cust041' size 4830M ;
alter tablespace cust add datafile '/oradev/cust042' size 4830M ;
alter tablespace cust add datafile '/oradev/cust043' size 4830M ;
alter tablespace cust add datafile '/oradev/cust044' size 4830M ;
alter tablespace cust add datafile '/oradev/cust045' size 4830M ;
alter tablespace cust add datafile '/oradev/cust046' size 4830M ;
alter tablespace cust add datafile '/oradev/cust047' size 4830M ;
alter tablespace cust add datafile '/oradev/cust048' size 4830M ;
alter tablespace cust add datafile '/oradev/cust049' size 4830M ;
alter tablespace cust add datafile '/oradev/cust050' size 4830M ;
alter tablespace cust add datafile '/oradev/cust051' size 4830M ;
alter tablespace cust add datafile '/oradev/cust052' size 4830M ;
alter tablespace cust add datafile '/oradev/cust053' size 4830M ;
alter tablespace cust add datafile '/oradev/cust054' size 4830M ;
alter tablespace cust add datafile '/oradev/cust055' size 4830M ;
alter tablespace cust add datafile '/oradev/cust056' size 4830M ;
alter tablespace cust add datafile '/oradev/cust057' size 4830M ;
alter tablespace cust add datafile '/oradev/cust058' size 4830M ;
alter tablespace cust add datafile '/oradev/cust059' size 4830M ;
alter tablespace cust add datafile '/oradev/cust060' size 4830M ;
alter tablespace cust add datafile '/oradev/cust061' size 4830M ;
alter tablespace cust add datafile '/oradev/cust062' size 4830M ;
alter tablespace cust add datafile '/oradev/cust063' size 4830M ;
alter tablespace cust add datafile '/oradev/cust064' size 4830M ;
alter tablespace cust add datafile '/oradev/cust065' size 4830M ;
alter tablespace cust add datafile '/oradev/cust066' size 4830M ;
alter tablespace cust add datafile '/oradev/cust067' size 4830M ;
alter tablespace cust add datafile '/oradev/cust068' size 4830M ;
alter tablespace cust add datafile '/oradev/cust069' size 4830M ;
alter tablespace cust add datafile '/oradev/cust070' size 4830M ;
alter tablespace cust add datafile '/oradev/cust071' size 4830M ;


create tablespace stk datafile '/oradev/stk000' size 4440M
extent management local
uniform size 4439M nologging;
alter tablespace stk add datafile '/oradev/stk001' size 4440M ;
alter tablespace stk add datafile '/oradev/stk002' size 4440M ;
alter tablespace stk add datafile '/oradev/stk003' size 4440M ;
alter tablespace stk add datafile '/oradev/stk004' size 4440M ;
alter tablespace stk add datafile '/oradev/stk005' size 4440M ;
alter tablespace stk add datafile '/oradev/stk006' size 4440M ;
alter tablespace stk add datafile '/oradev/stk007' size 4440M ;
alter tablespace stk add datafile '/oradev/stk008' size 4440M ;
alter tablespace stk add datafile '/oradev/stk009' size 4440M ;
alter tablespace stk add datafile '/oradev/stk010' size 4440M ;
alter tablespace stk add datafile '/oradev/stk011' size 4440M ;
alter tablespace stk add datafile '/oradev/stk012' size 4440M ;
alter tablespace stk add datafile '/oradev/stk013' size 4440M ;
alter tablespace stk add datafile '/oradev/stk014' size 4440M ;
alter tablespace stk add datafile '/oradev/stk015' size 4440M ;
```

```
alter tablespace stk add datafile '/oradev/stk016' size 4440M ;
alter tablespace stk add datafile '/oradev/stk017' size 4440M ;
alter tablespace stk add datafile '/oradev/stk018' size 4440M ;
alter tablespace stk add datafile '/oradev/stk019' size 4440M ;
alter tablespace stk add datafile '/oradev/stk020' size 4440M ;
alter tablespace stk add datafile '/oradev/stk021' size 4440M ;
alter tablespace stk add datafile '/oradev/stk022' size 4440M ;
alter tablespace stk add datafile '/oradev/stk023' size 4440M ;
alter tablespace stk add datafile '/oradev/stk024' size 4440M ;
alter tablespace stk add datafile '/oradev/stk025' size 4440M ;
alter tablespace stk add datafile '/oradev/stk026' size 4440M ;
alter tablespace stk add datafile '/oradev/stk027' size 4440M ;
alter tablespace stk add datafile '/oradev/stk028' size 4440M ;
alter tablespace stk add datafile '/oradev/stk029' size 4440M ;
alter tablespace stk add datafile '/oradev/stk030' size 4440M ;
alter tablespace stk add datafile '/oradev/stk031' size 4440M ;
alter tablespace stk add datafile '/oradev/stk032' size 4440M ;
alter tablespace stk add datafile '/oradev/stk033' size 4440M ;
alter tablespace stk add datafile '/oradev/stk034' size 4440M ;
alter tablespace stk add datafile '/oradev/stk035' size 4440M ;
alter tablespace stk add datafile '/oradev/stk036' size 4440M ;
alter tablespace stk add datafile '/oradev/stk037' size 4440M ;
alter tablespace stk add datafile '/oradev/stk038' size 4440M ;
alter tablespace stk add datafile '/oradev/stk039' size 4440M ;
alter tablespace stk add datafile '/oradev/stk040' size 4440M ;
alter tablespace stk add datafile '/oradev/stk041' size 4440M ;
alter tablespace stk add datafile '/oradev/stk042' size 4440M ;
alter tablespace stk add datafile '/oradev/stk043' size 4440M ;
alter tablespace stk add datafile '/oradev/stk044' size 4440M ;
alter tablespace stk add datafile '/oradev/stk045' size 4440M ;
alter tablespace stk add datafile '/oradev/stk046' size 4440M ;
alter tablespace stk add datafile '/oradev/stk047' size 4440M ;
alter tablespace stk add datafile '/oradev/stk048' size 4440M ;
alter tablespace stk add datafile '/oradev/stk049' size 4440M ;
alter tablespace stk add datafile '/oradev/stk050' size 4440M ;
alter tablespace stk add datafile '/oradev/stk051' size 4440M ;
alter tablespace stk add datafile '/oradev/stk052' size 4440M ;
alter tablespace stk add datafile '/oradev/stk053' size 4440M ;
alter tablespace stk add datafile '/oradev/stk054' size 4440M ;
alter tablespace stk add datafile '/oradev/stk055' size 4440M ;
alter tablespace stk add datafile '/oradev/stk056' size 4440M ;
alter tablespace stk add datafile '/oradev/stk057' size 4440M ;
alter tablespace stk add datafile '/oradev/stk058' size 4440M ;
alter tablespace stk add datafile '/oradev/stk059' size 4440M ;
alter tablespace stk add datafile '/oradev/stk060' size 4440M ;
alter tablespace stk add datafile '/oradev/stk061' size 4440M ;
alter tablespace stk add datafile '/oradev/stk062' size 4440M ;
alter tablespace stk add datafile '/oradev/stk063' size 4440M ;
alter tablespace stk add datafile '/oradev/stk064' size 4440M ;
alter tablespace stk add datafile '/oradev/stk065' size 4440M ;
alter tablespace stk add datafile '/oradev/stk066' size 4440M ;
alter tablespace stk add datafile '/oradev/stk067' size 4440M ;
alter tablespace stk add datafile '/oradev/stk068' size 4440M ;
alter tablespace stk add datafile '/oradev/stk069' size 4440M ;
alter tablespace stk add datafile '/oradev/stk070' size 4440M ;
alter tablespace stk add datafile '/oradev/stk071' size 4440M ;
alter tablespace stk add datafile '/oradev/stk072' size 4440M ;
alter tablespace stk add datafile '/oradev/stk073' size 4440M ;
alter tablespace stk add datafile '/oradev/stk074' size 4440M ;
alter tablespace stk add datafile '/oradev/stk075' size 4440M ;
alter tablespace stk add datafile '/oradev/stk076' size 4440M ;
alter tablespace stk add datafile '/oradev/stk077' size 4440M ;
alter tablespace stk add datafile '/oradev/stk078' size 4440M ;
alter tablespace stk add datafile '/oradev/stk079' size 4440M ;
alter tablespace stk add datafile '/oradev/stk080' size 4440M ;
alter tablespace stk add datafile '/oradev/stk081' size 4440M ;
alter tablespace stk add datafile '/oradev/stk082' size 4440M ;
alter tablespace stk add datafile '/oradev/stk083' size 4440M ;
alter tablespace stk add datafile '/oradev/stk084' size 4440M ;
alter tablespace stk add datafile '/oradev/stk085' size 4440M ;
alter tablespace stk add datafile '/oradev/stk086' size 4440M ;
alter tablespace stk add datafile '/oradev/stk087' size 4440M ;
alter tablespace stk add datafile '/oradev/stk088' size 4440M ;
alter tablespace stk add datafile '/oradev/stk089' size 4440M ;
alter tablespace stk add datafile '/oradev/stk090' size 4440M ;
alter tablespace stk add datafile '/oradev/stk091' size 4440M ;
alter tablespace stk add datafile '/oradev/stk092' size 4440M ;
alter tablespace stk add datafile '/oradev/stk093' size 4440M ;
alter tablespace stk add datafile '/oradev/stk094' size 4440M ;
alter tablespace stk add datafile '/oradev/stk095' size 4440M ;
alter tablespace stk add datafile '/oradev/stk096' size 4440M ;
alter tablespace stk add datafile '/oradev/stk097' size 4440M ;
alter tablespace stk add datafile '/oradev/stk098' size 4440M ;
alter tablespace stk add datafile '/oradev/stk099' size 4440M ;
alter tablespace stk add datafile '/oradev/stk100' size 4440M ;
alter tablespace stk add datafile '/oradev/stk101' size 4440M ;
alter tablespace stk add datafile '/oradev/stk102' size 4440M ;
alter tablespace stk add datafile '/oradev/stk103' size 4440M ;
alter tablespace stk add datafile '/oradev/stk104' size 4440M ;
alter tablespace stk add datafile '/oradev/stk105' size 4440M ;
alter tablespace stk add datafile '/oradev/stk106' size 4440M ;
alter tablespace stk add datafile '/oradev/stk107' size 4440M ;


create tablespace cl_ix datafile '/oradev/cl_ix000' size 2500M
extent management local
uniform size 2495M nologging;
alter tablespace cl_ix add datafile '/oradev/cl_ix001' size 2500M ;
alter tablespace cl_ix add datafile '/oradev/cl_ix002' size 2500M ;
alter tablespace cl_ix add datafile '/oradev/cl_ix003' size 2500M ;
alter tablespace cl_ix add datafile '/oradev/cl_ix004' size 2500M ;
alter tablespace cl_ix add datafile '/oradev/cl_ix005' size 2500M ;
alter tablespace cl_ix add datafile '/oradev/cl_ix006' size 2500M ;
alter tablespace cl_ix add datafile '/oradev/cl_ix007' size 2500M ;


create tablespace wd_N0 datafile '/oradev/wd000' size 100M
extent management local
uniform size 1M nologging;


create tablespace wd_N1 datafile '/oradev/wd001' size 100M
extent management local
uniform size 1M nologging;


create tablespace wd_N2 datafile '/oradev/wd002' size 100M
extent management local
uniform size 1M nologging;


create tablespace wd_N3 datafile '/oradev/wd003' size 100M
extent management local
uniform size 1M nologging;



create tablespace newo_N0 datafile '/oradev/newo000' size 700M
extent management local
uniform size 699M nologging;


create tablespace newo_N1 datafile '/oradev/newo001' size 700M
extent management local
uniform size 699M nologging;


create tablespace newo_N2 datafile '/oradev/newo002' size 700M
extent management local
uniform size 699M nologging;


create tablespace newo_N3 datafile '/oradev/newo003' size 700M
extent management local
uniform size 699M nologging;


create tablespace newo_ix_N0 datafile '/oradev/newo_ix000' size 1200M
       extent management local
       uniform size 1199M nologging;


create tablespace newo_ix_N1 datafile '/oradev/newo_ix001' size 1200M
       extent management local
       uniform size 1199M nologging;


create tablespace newo_ix_N2 datafile '/oradev/newo_ix002' size 1200M
       extent management local
       uniform size 1199M nologging;


create tablespace newo_ix_N3 datafile '/oradev/newo_ix003' size 1200M
       extent management local
       uniform size 1199M nologging;


create tablespace ord_ix_N0 datafile '/oradev/ord_ix000' size 1400M
extent management local
uniform size 1399M nologging;
alter tablespace ord_ix_N0 add datafile '/oradev/ord_ix001' size 1400M ;


create tablespace ord_ix_N1 datafile '/oradev/ord_ix002' size 1400M
extent management local
uniform size 1399M nologging;
alter tablespace ord_ix_N1 add datafile '/oradev/ord_ix003' size 1400M ;


create tablespace ord_ix_N2 datafile '/oradev/ord_ix004' size 1400M
extent management local
uniform size 1399M nologging;
alter tablespace ord_ix_N2 add datafile '/oradev/ord_ix005' size 1400M ;


create tablespace ord_ix_N3 datafile '/oradev/ord_ix006' size 1400M
extent management local
uniform size 1399M nologging;
alter tablespace ord_ix_N3 add datafile '/oradev/ord_ix007' size 1400M ;


create tablespace ord2_ix_N0 datafile '/oradev/ord2_ix000' size 2047M
extent management local
uniform size 2046M nologging;
alter tablespace ord2_ix_N0 add datafile '/oradev/ord2_ix001' size 2047M ;


create tablespace ord2_ix_N1 datafile '/oradev/ord2_ix002' size 2047M
extent management local
uniform size 2046M nologging;
alter tablespace ord2_ix_N1 add datafile '/oradev/ord2_ix003' size 2047M ;


create tablespace ord2_ix_N2 datafile '/oradev/ord2_ix004' size 2047M
extent management local
uniform size 2046M nologging;
alter tablespace ord2_ix_N2 add datafile '/oradev/ord2_ix005' size 2047M ;


create tablespace ord2_ix_N3 datafile '/oradev/ord2_ix006' size 2047M
extent management local
uniform size 2046M nologging;
alter tablespace ord2_ix_N3 add datafile '/oradev/ord2_ix007' size 2047M ;


create tablespace hist_N0 datafile '/oradev/hist000' size 1685M
extent management local
uniform size 1684M nologging;
alter tablespace hist_N0 add datafile '/oradev/hist001' size 1685M ;
alter tablespace hist_N0 add datafile '/oradev/hist002' size 1685M ;
alter tablespace hist_N0 add datafile '/oradev/hist003' size 1685M ;


create tablespace hist_N1 datafile '/oradev/hist004' size 1685M
extent management local
uniform size 1684M nologging;
alter tablespace hist_N1 add datafile '/oradev/hist005' size 1685M ;
alter tablespace hist_N1 add datafile '/oradev/hist006' size 1685M ;
```

```
alter tablespace hist_N1 add datafile '/oradev/hist007' size 1685M ;


create tablespace hist_N2 datafile '/oradev/hist008' size 1685M
extent management local
uniform size 1684M nologging;
alter tablespace hist_N2 add datafile '/oradev/hist009' size 1685M ;
alter tablespace hist_N2 add datafile '/oradev/hist010' size 1685M ;
alter tablespace hist_N2 add datafile '/oradev/hist011' size 1685M ;


create tablespace hist_N3 datafile '/oradev/hist012' size 1685M
extent management local
uniform size 1684M nologging;
alter tablespace hist_N3 add datafile '/oradev/hist013' size 1685M ;
alter tablespace hist_N3 add datafile '/oradev/hist014' size 1685M ;
alter tablespace hist_N3 add datafile '/oradev/hist015' size 1685M ;


create tablespace ol_ix_N0 datafile '/oradev/ol_ix000' size 4095M
extent management local
uniform size 4094M nologging;
alter tablespace ol_ix_N0 add datafile '/oradev/ol_ix001' size 4095M ;
alter tablespace ol_ix_N0 add datafile '/oradev/ol_ix002' size 4095M ;
alter tablespace ol_ix_N0 add datafile '/oradev/ol_ix003' size 4095M ;
alter tablespace ol_ix_N0 add datafile '/oradev/ol_ix004' size 4095M ;
alter tablespace ol_ix_N0 add datafile '/oradev/ol_ix005' size 4095M ;
alter tablespace ol_ix_N0 add datafile '/oradev/ol_ix006' size 4095M ;
alter tablespace ol_ix_N0 add datafile '/oradev/ol_ix007' size 4095M ;


create tablespace ol_ix_N1 datafile '/oradev/ol_ix008' size 4095M
extent management local
uniform size 4094M nologging;
alter tablespace ol_ix_N1 add datafile '/oradev/ol_ix009' size 4095M ;
alter tablespace ol_ix_N1 add datafile '/oradev/ol_ix010' size 4095M ;
alter tablespace ol_ix_N1 add datafile '/oradev/ol_ix011' size 4095M ;
alter tablespace ol_ix_N1 add datafile '/oradev/ol_ix012' size 4095M ;
alter tablespace ol_ix_N1 add datafile '/oradev/ol_ix013' size 4095M ;
alter tablespace ol_ix_N1 add datafile '/oradev/ol_ix014' size 4095M ;
alter tablespace ol_ix_N1 add datafile '/oradev/ol_ix015' size 4095M ;


create tablespace ol_ix_N2 datafile '/oradev/ol_ix016' size 4095M
extent management local
uniform size 4094M nologging;
alter tablespace ol_ix_N2 add datafile '/oradev/ol_ix017' size 4095M ;
alter tablespace ol_ix_N2 add datafile '/oradev/ol_ix018' size 4095M ;
alter tablespace ol_ix_N2 add datafile '/oradev/ol_ix019' size 4095M ;
alter tablespace ol_ix_N2 add datafile '/oradev/ol_ix020' size 4095M ;
alter tablespace ol_ix_N2 add datafile '/oradev/ol_ix021' size 4095M ;
alter tablespace ol_ix_N2 add datafile '/oradev/ol_ix022' size 4095M ;
alter tablespace ol_ix_N2 add datafile '/oradev/ol_ix023' size 4095M ;


create tablespace ol_ix_N3 datafile '/oradev/ol_ix024' size 4095M
extent management local
uniform size 4094M nologging;
alter tablespace ol_ix_N3 add datafile '/oradev/ol_ix025' size 4095M ;
alter tablespace ol_ix_N3 add datafile '/oradev/ol_ix026' size 4095M ;
alter tablespace ol_ix_N3 add datafile '/oradev/ol_ix027' size 4095M ;
alter tablespace ol_ix_N3 add datafile '/oradev/ol_ix028' size 4095M ;
alter tablespace ol_ix_N3 add datafile '/oradev/ol_ix029' size 4095M ;
alter tablespace ol_ix_N3 add datafile '/oradev/ol_ix030' size 4095M ;
alter tablespace ol_ix_N3 add datafile '/oradev/ol_ix031' size 4095M ;


create tablespace ord_N0 datafile '/oradev/ord000' size 1500M
extent management local
uniform size 1499M nologging;
alter tablespace ord_N0 add datafile '/oradev/ord001' size 1500M ;
alter tablespace ord_N0 add datafile '/oradev/ord002' size 1500M ;


create tablespace ord_N1 datafile '/oradev/ord003' size 1500M
extent management local
uniform size 1499M nologging;
alter tablespace ord_N1 add datafile '/oradev/ord004' size 1500M ;
alter tablespace ord_N1 add datafile '/oradev/ord005' size 1500M ;


create tablespace ord_N2 datafile '/oradev/ord006' size 1500M
extent management local
uniform size 1499M nologging;
alter tablespace ord_N2 add datafile '/oradev/ord007' size 1500M ;
alter tablespace ord_N2 add datafile '/oradev/ord008' size 1500M ;


create tablespace ord_N3 datafile '/oradev/ord009' size 1500M
extent management local
uniform size 1499M nologging;
alter tablespace ord_N3 add datafile '/oradev/ord010' size 1500M ;
alter tablespace ord_N3 add datafile '/oradev/ord011' size 1500M ;


create tablespace oline_N0 datafile '/oradev/oline000' size 4720M
extent management local
uniform size 4719M nologging;
alter tablespace oline_N0 add datafile '/oradev/oline001' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline002' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline003' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline004' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline005' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline006' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline007' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline008' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline009' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline010' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline011' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline012' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline013' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline014' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline015' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline016' size 4720M ;
alter tablespace oline_N0 add datafile '/oradev/oline017' size 4720M ;


create tablespace oline_N1 datafile '/oradev/oline018' size 4720M
extent management local
uniform size 4719M nologging;
alter tablespace oline_N1 add datafile '/oradev/oline019' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline020' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline021' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline022' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline023' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline024' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline025' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline026' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline027' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline028' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline029' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline030' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline031' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline032' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline033' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline034' size 4720M ;
alter tablespace oline_N1 add datafile '/oradev/oline035' size 4720M ;


create tablespace oline_N2 datafile '/oradev/oline036' size 4720M
extent management local
uniform size 4719M nologging;
alter tablespace oline_N2 add datafile '/oradev/oline037' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline038' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline039' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline040' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline041' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline042' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline043' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline044' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline045' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline046' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline047' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline048' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline049' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline050' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline051' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline052' size 4720M ;
alter tablespace oline_N2 add datafile '/oradev/oline053' size 4720M ;


create tablespace oline_N3 datafile '/oradev/oline054' size 4720M
extent management local
uniform size 4719M nologging;
alter tablespace oline_N3 add datafile '/oradev/oline055' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline056' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline057' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline058' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline059' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline060' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline061' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline062' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline063' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline064' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline065' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline066' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline067' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline068' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline069' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline070' size 4720M ;
alter tablespace oline_N3 add datafile '/oradev/oline071' size 4720M ;


create tablespace ord_alt_N0 datafile '/oradev/ord_alt000' size 1500M
extent management local
uniform size 1499M nologging;
alter tablespace ord_alt_N0 add datafile '/oradev/ord_alt001' size 1500M ;
alter tablespace ord_alt_N0 add datafile '/oradev/ord_alt002' size 1500M ;

create tablespace ord_alt_N1 datafile '/oradev/ord_alt003' size 1500M
extent management local
uniform size 1499M nologging;
alter tablespace ord_alt_N1 add datafile '/oradev/ord_alt004' size 1500M ;
alter tablespace ord_alt_N1 add datafile '/oradev/ord_alt005' size 1500M ;

create tablespace ord_alt_N2 datafile '/oradev/ord_alt006' size 1500M
extent management local
uniform size 1499M nologging;
alter tablespace ord_alt_N2 add datafile '/oradev/ord_alt007' size 1500M ;
alter tablespace ord_alt_N2 add datafile '/oradev/ord_alt008' size 1500M ;

create tablespace ord_alt_N3 datafile '/oradev/ord_alt009' size 1500M
extent management local
uniform size 1499M nologging;
alter tablespace ord_alt_N3 add datafile '/oradev/ord_alt010' size 1500M ;
alter tablespace ord_alt_N3 add datafile '/oradev/ord_alt011' size 1500M ;

create tablespace c_ix datafile '/oradev/c_ix000' size 1604M
       extent management local
       uniform size 400M nologging;
alter tablespace c_ix add datafile '/oradev/c_ix001' size 1604M ;
alter tablespace c_ix add datafile '/oradev/c_ix002' size 1604M ;
alter tablespace c_ix add datafile '/oradev/c_ix003' size 1604M ;
alter tablespace c_ix add datafile '/oradev/c_ix004' size 1604M ;
alter tablespace c_ix add datafile '/oradev/c_ix005' size 1604M ;


create tablespace s_ix datafile '/oradev/s_ix000' size 2004M
extent management local
uniform size 50M nologging;
alter tablespace s_ix add datafile '/oradev/s_ix001' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix002' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix003' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix004' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix005' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix006' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix007' size 2004M ;
```

```
alter tablespace s_ix add datafile '/oradev/s_ix008' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix009' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix010' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix011' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix012' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix013' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix014' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix015' size 2004M ;
alter tablespace s_ix add datafile '/oradev/s_ix016' size 2004M ;



create temporary tablespace temp tempfile
'/oradev/temp000' size 2000M reuse,
'/oradev/temp001' size 2000M reuse,
'/oradev/temp002' size 2000M reuse,
'/oradev/temp003' size 2000M reuse,
'/oradev/temp004' size 2000M reuse,
'/oradev/temp005' size 2000M reuse,
'/oradev/temp006' size 2000M reuse,
'/oradev/temp007' size 2000M reuse,
'/oradev/temp008' size 2000M reuse,
'/oradev/temp009' size 2000M reuse,
'/oradev/temp010' size 2000M reuse,
'/oradev/temp011' size 2000M reuse,
'/oradev/temp012' size 2000M reuse,
'/oradev/temp013' size 2000M reuse,
'/oradev/temp014' size 2000M reuse,
'/oradev/temp015' size 2000M reuse,
'/oradev/temp016' size 2000M reuse,
'/oradev/temp017' size 2000M reuse,
'/oradev/temp018' size 2000M reuse,
'/oradev/temp019' size 2000M reuse
extent management local uniform size 100M;


==> rbsegs.ddl <==

/bin/nawk 'BEGIN {
print "connect internal"
# use an rbs from the build
print "alter rollback segment r01 online;"
print "set transaction use rollback segment r01;"
# create 4x300, online my instances segs only
for (i = 0; i < 1200; i++) {
    printf("create rollback segment t%d tablespace roll_n%d ;\n", i, int(i/300))
    if (i<300) print "alter rollback segment t" i " online;"
}

print "set transaction use rollback segment t0;"
print "alter rollback segment r01 offline;"
for (i=1; i<= 9; i++) {
print "drop rollback segment  r0" i ";"
}
for (i=1; i<= 9; i++) {
print "create rollback segment r0" i " tablespace system;"
}
}' | svrmgrl


==> clusters.ddl <==
connect dbbench/dbbench

create cluster icluster (
i_id            number (6,0)
)
singletable
hashkeys        100000
hash is         i_id
size            120
initrans        3
pctfree         0
tablespace      item;

create cluster ccluster (
    c_id         number(5,0),
    c_d_id       number(2,0),
    c_w_id       number(5,0)
)
singletable
hashkeys        330000020
hash is         (c_w_id * 30000 + c_id * 10 + c_d_id - 33001)
size            850
initrans        3
pctfree         0
tablespace      cust;

create cluster scluster (
   s_i_id       number(6,0),
   s_w_id       number(5,0)
)
singletable
hashkeys        1100000000
hash is         ( (s_i_id - 1) * 2750 + mod((s_w_id - 1), 2750) +
 trunc ((s_w_id - 1) / 2750) * 275000000)
size            350
initrans        3
pctfree         0
tablespace      stk;


==> tables.ddl <==

connect dbbench/dbbench

create table district (
        d_id            number(2,0),
        d_w_id          number(5,0),
        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)
       )
        partition by range (d_w_id) (
  partition p1 values less than (2751)     tablespace wd_n0,
         partition p2 values less than (5501)     tablespace wd_n1,
         partition p3 values less than (8251)     tablespace wd_n2,
         partition p4 values less than (maxvalue) tablespace wd_n3
        )
initrans4
pctfree95
pctused4;


create table history (
        h_c_id          number,
        h_c_d_id        number,
        h_c_w_id        number,
        h_d_id          number,
        h_w_id          number,
        h_date          date,
        h_amount        number(6),
        h_data          varchar2(24)
        )
        partition by range (h_w_id)
        (
        partition p1 values less than (2751)     tablespace hist_n0,
        partition p2 values less than (5501)     tablespace hist_n1,
        partition p3 values less than (8251)     tablespace hist_n2,
        partition p4 values less than (maxvalue) tablespace hist_n3
        )
        initrans 3
        pctfree  1
pctused99
        storage (freelist groups 43 freelists 13
 buffer_pool recycle);



alter tablespace newo_n0 logging;
alter tablespace newo_n1 logging;
alter tablespace newo_n2 logging;
alter tablespace newo_n3 logging;
       create table new_order (
                no_w_id         number,
                no_d_id         number,
                no_o_id         number,
constraint inord primary key (no_w_id, no_d_id, no_o_id)
        )
organization index
partition by range (no_w_id)
        (
        partition p1 values less than (2751)     tablespace newo_n0,
        partition p2 values less than (5501)     tablespace newo_n1,
        partition p3 values less than (8251)     tablespace newo_n2,
        partition p4 values less than (maxvalue) tablespace newo_n3
        )
        initrans 4
        pctfree  5
storage ( freelist groups 13 freelists 22);


alter tablespace oline_n0 logging;
alter tablespace oline_n1 logging;
alter tablespace oline_n2 logging;
alter tablespace oline_n3 logging;
       create table order_line (
        ol_w_id         number,
        ol_d_id         number,
        ol_o_id         number,
        ol_number       number,
        ol_i_id         number,
        ol_delivery_d   date,
        ol_amount       number(6),
        ol_supply_w_id  number,
        ol_quantity     number,
        ol_dist_info    char(24),
constraint iordl primary key (ol_w_id, ol_d_id, ol_o_id, ol_number)
        )
organization index
  partition by range (ol_w_id)
        (
         partition p1 values less than (2751)     tablespace oline_n0,
         partition p2 values less than (5501)     tablespace oline_n1,
         partition p3 values less than (8251)     tablespace oline_n2,
         partition p4 values less than (maxvalue) tablespace oline_n3
        )
        initrans 4
        pctfree  5
        storage ( freelist groups 43 freelists 13);


       create table orders (
        o_id            number,
        o_w_id          number,
        o_d_id          number,
        o_c_id          number,
        o_carrier_id    number,
        o_ol_cnt        number,
        o_all_local     number,
        o_entry_d       date
        )
        partition by range (o_w_id)
        (
         partition p1 values less than (2751)     tablespace ord_n0,
         partition p2 values less than (5501)     tablespace ord_n1,
         partition p3 values less than (8251)     tablespace ord_n2,
         partition p4 values less than (maxvalue) tablespace ord_n3
        )
        initrans 3
        pctfree  5
pctused 95
```

```
        storage ( freelist groups 13 freelists 22);


        create table warehouse (
                w_id            number(5,0),
                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)
        )
        partition by range (w_id)
        (
         partition p1 values less than (2751)     tablespace wd_n0,
         partition p2 values less than (5501)     tablespace wd_n1,
         partition p3 values less than (8251)     tablespace wd_n2,
         partition p4 values less than (maxvalue) tablespace wd_n3
        )
        initrans                4
        pctfree                 95
        pctused                 4;



        create table item (
        i_id            number(6,0),
        i_name          varchar2(24),
        i_price         number(5,0),
        i_data          varchar2(50),
        i_im_id         number
        )
        cluster icluster(i_id);

create table customer (
        c_id            number(5,0),
        c_d_id          number(2,0),
        c_w_id          number(5,0),
        c_discount      number(4),
        c_credit        char(2),
        c_last          varchar2(16),
        c_first         varchar2(16),
        c_credit_lim    number(12),
        c_balance       number(12),
        c_ytd_payment   number(12),
        c_payment_cnt   number(4),
        c_delivery_cnt  number(4),
        c_street_1      varchar2(20),
        c_street_2      varchar2(20),
        c_city          varchar2(20),
        c_state         char(2),
        c_zip           char(9),
        c_phone         char(16),
        c_since         date,
        c_middle        char(2),
        c_data          varchar2(500)
        )
        cluster ccluster (c_id, c_d_id, c_w_id);

        create table stock (
        s_i_id          number(6,0),
        s_w_id          number(5,0),
        s_quantity      number(6,0),
        s_ytd           number(10,0),
        s_order_cnt     number(6,0),
        s_remote_cnt    number(6,0),
        s_data          varchar2(50),
        s_dist_01       char(24),
        s_dist_02       char(24),
        s_dist_03       char(24),
        s_dist_04       char(24),
        s_dist_05       char(24),
        s_dist_06       char(24),
        s_dist_07       char(24),
        s_dist_08       char(24),
        s_dist_09       char(24),
        s_dist_10       char(24)
        )
        cluster scluster (s_i_id, s_w_id);

==> indexes.ddl <==
create unique index clast_idx on customer(c_last, c_d_id, c_w_id, c_first)
        tablespace cl_ix
        nologging
        initrans 3
        parallel 24
        pctfree  1;


create unique index d_idx on district(d_w_id, d_id)
        nologging local
       ( partition p1 tablespace wd_N0,
         partition p2 tablespace wd_N1,
         partition p3 tablespace wd_N2,
         partition p4 tablespace wd_N3
       )
        initrans 3
        pctfree  1
        parallel 24;



create unique index w_idx on warehouse (w_id)
        nologging local
       ( partition p1 tablespace wd_N0,
         partition p2 tablespace wd_N1,
         partition p3 tablespace wd_N2,
         partition p4 tablespace wd_N3
       )
        initrans 3
        pctfree 1
        parallel 24;


create unique index i_idx on item (i_id)
tablespace wd_N0
        initrans 3
        pctfree 1
        parallel 24;


create unique index icustomer on customer(c_w_id, c_d_id, c_id)
        tablespace c_ix
        initrans 3
        parallel 12
        pctfree  1
        storage (freelist groups 13 freelists 22);

create unique index istock on stock(s_i_id, s_w_id)
        tablespace s_ix
        initrans 3
        parallel 20
        pctfree  1
        storage (freelist groups 13 freelists 22);



==> orders2.ddl <==
connect dbbench/dbbench

       create table orders2 (
        o_id            ,
        o_w_id          ,
        o_d_id          ,
        o_c_id          ,
        o_carrier_id    ,
        o_ol_cnt        ,
        o_all_local     ,
        o_entry_d
        )
        partition by range (o_w_id)
        (
         partition p1 values less than (2751)     tablespace ord_alt_n0,
         partition p2 values less than (5501)     tablespace ord_alt_n1,
         partition p3 values less than (8251)     tablespace ord_alt_n2,
         partition p4 values less than (maxvalue) tablespace ord_alt_n3
        )
        initrans 3
        pctfree  5
pctused 95
        storage ( freelist groups 41 freelists 22)
        parallel (degree 25)
        as
        select * from orders;

drop table orders;
create synonym orders for orders2;
grant all on orders to public;

create unique index o_idx1 on orders(o_w_id, o_d_id, o_id)
nologging local
       ( partition p1 tablespace ord_ix_N0,
         partition p2 tablespace ord_ix_N1,
         partition p3 tablespace ord_ix_N2,
         partition p4 tablespace ord_ix_N3
       )
        initrans 3
storage ( freelists 13 freelist groups 43 )
pctfree 1
parallel 24;


create unique index o_idx2 on orders(o_c_id, o_d_id, o_w_id, o_id)
nologging local
       ( partition p1 tablespace ord2_ix_N0,
         partition p2 tablespace ord2_ix_N1,
         partition p3 tablespace ord2_ix_N2,
         partition p4 tablespace ord2_ix_N3
       )
        initrans 3
storage ( freelists 13 freelist groups 43 )
parallel 24
        pctfree  25;

connect system/manager
drop tablespace ord_n0;
drop tablespace ord_n1;
drop tablespace ord_n2;
drop tablespace ord_n3;
alter tablespace ord_alt_N0 add datafile '/oradev/ord001' size 1500M ;
alter tablespace ord_alt_N0 add datafile '/oradev/ord002' size 1500M ;
alter tablespace ord_alt_N0 add datafile '/oradev/ord000' size 1500M ;
alter tablespace ord_alt_N1 add datafile '/oradev/ord004' size 1500M ;
alter tablespace ord_alt_N1 add datafile '/oradev/ord005' size 1500M ;
alter tablespace ord_alt_N1 add datafile '/oradev/ord003' size 1500M ;
alter tablespace ord_alt_N2 add datafile '/oradev/ord007' size 1500M ;
alter tablespace ord_alt_N2 add datafile '/oradev/ord008' size 1500M ;
alter tablespace ord_alt_N2 add datafile '/oradev/ord006' size 1500M ;
alter tablespace ord_alt_N3 add datafile '/oradev/ord010' size 1500M ;
alter tablespace ord_alt_N3 add datafile '/oradev/ord011' size 1500M ;
alter tablespace ord_alt_N3 add datafile '/oradev/ord009' size 1500M ;


==> synonyms.ddl <==
create synonym ware for dbbench.warehouse;
create synonym dist for dbbench.district;
create synonym cust for dbbench.customer;
create synonym hist for dbbench.history;
create synonym nord for dbbench.new_order;
create synonym ordr for dbbench.orders;
create synonym ordl for dbbench.order_line;
create synonym item for dbbench.item;
create synonym stok for dbbench.stock;

==> analyze.ddl <==
```

```
tspaces=`cat<<end
SYSTEM
ROLL_N0
ROLL_N1
ROLL_N2
ROLL_N3
CUST
STK
CL_IX
WD_N0
WD_N1
WD_N2
WD_N3
ITEM
NEWO_N0
NEWO_N1
NEWO_N2
NEWO_N3
NEWO_IX_N0
NEWO_IX_N1
NEWO_IX_N2
NEWO_IX_N3
ORD_N0
ORD_N1
ORD_N2
ORD_N3
ORD_IX_N0
ORD_IX_N1
ORD_IX_N2
ORD_IX_N3
ORD2_IX_N0
ORD2_IX_N1
ORD2_IX_N2
ORD2_IX_N3
HIST_N0
HIST_N1
HIST_N2
HIST_N3
OL_IX_N0
OL_IX_N1
OL_IX_N2
OL_IX_N3
OLINE_N0
OLINE_N1
OLINE_N2
OLINE_N3
TEMP
C_IX
S_IX
ORD_ALT_N0
ORD_ALT_N1
ORD_ALT_N2
ORD_ALT_N3
end `

tables=`cat <<end
CUSTOMER
DISTRICT
HISTORY
ITEM
NEW_ORDER
ORDERS2
ORDER_LINE
STOCK
WAREHOUSE
end `

indexes=`cat<<end
CLAST_IDX
D_IDX
ICUSTOMER
INORD
IORDL
ISTOCK
I_IDX
O_IDX1
O_IDX2
W_IDX
end `

echo  $tables | xargs -n 1 echo | nawk '
BEGIN{ print "connect internal;" }
{ print "alter table dbbench." $1 " enable table lock;" }
' | svrmgrl



echo  $tspaces | xargs -n 1 echo | nawk '
BEGIN   { print "connect internal;" }
        { print "alter tablespace " $1 " logging;" }
{ print "exit" }
' | svrmgrl

sqlplus dbbench/dbbench<<EOT
set timing on;
analyze table warehouse compute statistics;
analyze table district compute statistics;
analyze table customer estimate statistics sample 5000 rows;
%% analyze table %%orders estimate statistics sample 5000 rows;
% analyze table new_order estimate statistics;
% analyze table order_line estimate statistics sample 5000 rows;
analyze table history estimate statistics;
analyze table item compute statistics;
analyze table stock estimate statistics sample 5000 rows;
analyze index w_idx compute statistics;
analyze index d_idx compute statistics;
analyze index clast_idx estimate statistics sample 2 percent;
% analyze index n_idx compute statistics;
% analyze index ol_idx estimate statistics;
analyze table orders2 estimate statistics sample 5000 rows;
analyze index o_idx1 estimate statistics sample 1 percent;
analyze index o_idx2 estimate statistics sample 1 percent;

exit;
EOT


echo  $indexes | xargs -n 1 echo | nawk '
BEGIN   { print "connect internal;" }
        { print "alter index dbbench." $1 " noparallel;" }
' | svrmgrl

echo  $tables | xargs -n 1 echo | nawk '
BEGIN{ print "connect internal;" }
{ print "alter table dbbench." $1 " noparallel;" }
{ print "alter table dbbench." $1 " disable table lock;" }
' | svrmgrl


==> views.ddl <==
create or replace view wh_cust
(w_id, w_tax, c_id, c_d_id, c_w_id, c_discount, c_last, c_credit)
as select w.w_id, w.w_tax,
          c.c_id, c.c_d_id, c.c_w_id, c.c_discount, c.c_last, c.c_credit
   from customer c, warehouse w
   where w.w_id = c.c_w_id
/

create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
   from district d, warehouse w
   where w.w_id = d.d_w_id
/

create  or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
 select i.i_id, s_w_id, i.i_price, i.i_name, i.i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
  from stock s, item i
 where i.i_id = s.s_i_id
/

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;
/



-- The initnew package for storing variables used in the
-- New Order anonymous block

CREATE OR REPLACE PACKAGE initnew
AS
 TYPE intarray IS TABLE OF INTEGER index by binary_integer;
 TYPE distarray IS TABLE OF VARCHAR(24) index by binary_integer;
 nulldate       DATE;
 s_distdistarray;
 idx1arrintarray;
 s_remoteintarray;
 PROCEDURE new_init(idxarr intarray);
END initnew;
/
show errors;

CREATE OR REPLACE PACKAGE BODY initnew AS
  PROCEDURE new_init (idxarr  intarray)
  IS
  BEGIN
        -- initialize null date
    nulldate := TO_DATE('01-01-1811', 'MM-DD-YYYY');
     idx1arr := idxarr;
  END new_init;
END initnew;
/
```

build_dir/loader/tpcc.h

```
/*
 * $Header: tpcc.h 7030100.1 95/07/19 15:10:55 plai Generic<base> $ Copyr (c) 1993
Oracle
 */
/*=============================================================+
 |         Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
 |                    OPEN SYSTEMS PERFORMANCE GROUP                |
 |                         All Rights Reserved                      |
 +=============================================================+
 | FILENAME
 |    tpcc.h
 | DESCRIPTION
 |    Include file for TPC-C benchmark programs.
 +=============================================================*/
```

```
#ifndef TPCC_H
#define TPCC_H

#ifndef FALSE
# define FALSE 0
#endif

#ifndef TRUE
# define TRUE 1
#endif

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>
/*
#ifdef __STDC__
#include "ociapr.h"
#else
#include "ocikpr.h"
#endif
*/

typedef struct cda_def csrdef;
typedef struct cda_def ldadef;


/* TPC-C transaction functions */

extern int TPCinit ();
extern int TPCnew ();
extern int TPCpay ();
extern int TPCord ();
extern int TPCdel ();
extern int TPCsto ();
extern int TPCexit ();
extern int TPCdumpinit ();
extern int TPCdumpnew ();
extern int TPCdumppay ();
extern int TPCdumpord ();
extern int TPCdumpdel ();
extern int TPCdumpsto ();
extern int TPCdumpexit ();


/* Error codes */

#define RECOVERR -10
#define IRRECERR -20
#define NOERR    111
#define DEL_ERROR -666
#define DEL_DATE_LEN 7
#define NDISTS 10
#define NITEMS 15
#define SQL_BUF_SIZE 8192

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



#define DELRT 80.0

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

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

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

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


extern errrpt ();
extern int ocierror(char *fname, int lineno,OCIError *errhp, sword status);
extern int sqlfile(char *fname, text *linebuf);

extern FILE *lfp;
extern FILE *fopen ();
extern int proc_no;
extern int doid[];

extern int execstatus;
extern int errcode;

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

/* for stock-level transaction */

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

/* for delivery transaction */

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

/* for order-status transaction */

extern int bylastname;
extern char c_last[17];
extern char c_first[17];
extern char c_middle[3];
extern double c_balance;
extern int o_id;
extern text o_entry_d[20];
extern int o_carrier_id;
extern int o_ol_cnt;
extern int ol_supply_w_id[15];
extern int ol_i_id[15];
extern int ol_quantity[15];
extern int ol_amount[15];
ub4    ol_del_len[15];
/*
extern text ol_delivery_d[15][11];
*/
extern text ol_delivery_d[15][64];

/* for payment transaction */

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

/* for new order transaction */

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

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

#ifndef DISCARD
# define DISCARD (void)
#endif

#ifndef sword
# define sword int
#endif

#define VER7            2

#define NA              -1      /* ANSI SQL NULL */
#define NLT             1       /* length for string null terminator */
#define DEADLOCK        60      /* ORA-00060: deadlock */
#define NO_DATA_FOUND   1403    /* ORA-01403: no data found */
#define NOT_SERIALIZABLE  8177  /* ORA-08177: transaction not serializable */
#define SNAPSHOT_TOO_OLD  1555  /* ORA-01555: snapshot too old */
```

```
#ifndef NULLP
# define NULLP (void *)NULL
#endif /* NULLP */

#define ADR(object) ((ub1 *)&(object))
#define SIZ(object) ((sword)sizeof(object))

typedef char date[24+NLT];
typedef char varchar2;

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

#define OCIERROR(errp,function)\
        ocierror(__FILE__,__LINE__,(errp),(function));

#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
       ocierror(__FILE__,__LINE__,(errp), \
            OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
       ocierror(__FILE__,__LINE__, (errp), \
            OCIBindByName((stmp), &(bndp), (errp), \
     (text *)(sqlvar), strlen((sqlvar)),\
     (progv), (progvl), (ftype),0,0,0,0,0,OCI_DEFAULT));

#define OCIBNDRA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
       ocierror(__FILE__,__LINE__,(errp), \
            OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
       ocierror(__FILE__,__LINE__,(errp), \
            OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
     (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));

#define OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,cbf_nodata,cbf_data) \
       ocierror(__FILE__,__LINE__,(errp), \
            OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
       ocierror(__FILE__,__LINE__,(errp), \
            OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar), \
                          strlen((sqlvar)),0,(progvl),(ftype), \
                          indp,0,0,0,0,OCI_DATA_AT_EXEC)); \
       ocierror(__FILE__,__LINE__,(errp), \
            OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)));



#define OCIBNDR(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
       ocierror(__FILE__,__LINE__,(errp), \
          OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
       ocierror(__FILE__,__LINE__,(errp), \
          OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
     (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));

#define OCIBNDRAA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode,ms,cu) \
       ocierror(__FILE__,__LINE__, (errp), \
         OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));\
       ocierror(__FILE__,__LINE__,(errp),\
         OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
      (progv),(progvl),(ftype),(indp),(alen),(arcode),(ms),(cu),OCI_DEFAULT));

#define OCIDEFINE(stmp,dfnp,errp,pos,progv,progvl,ftype)\
       OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),(ftype),\
      0,0,0,OCI_DEFAULT);


#define OCIDEF(stmp,dfnp,errp,pos,progv,progvl,ftype) \
       OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
      (dvoid**)0);\
       OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),\
                      (ftype),NULL,NULL,NULL,OCI_DEFAULT); \


#define OCIDFNRA(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,alen,arcode) \
       OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
      (dvoid**)0);\
       OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
      (progvl),(ftype),(indp),(alen),\
      (arcode),OCI_DEFAULT);

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



/* New order */

struct newinstruct {
   int w_id;
   int d_id;
   int c_id;
   int ol_i_id[15];
   int ol_supply_w_id[15];
   int ol_quantity[15];
};

struct newoutstruct {
   int terror;
   int o_id;
   int o_ol_cnt;
   char c_last[17];
   char c_credit[3];
   float c_discount;
   float w_tax;
   float d_tax;
   char o_entry_d[20];
   float total_amount;
   char i_name[15][25];
   int s_quantity[15];
   char brand_generic[15];
   float i_price[15];
   float ol_amount[15];
   char status[26];
   int retry;
};

struct newstruct {
   struct newinstruct newin;
   struct newoutstruct newout;
};


/* Payment */

struct payinstruct {
   int w_id;
   int d_id;
   int c_w_id;
   int c_d_id;
   int c_id;
   int bylastname;
   int h_amount;
   char c_last[17];
};

struct payoutstruct {
   int terror;
   char w_street_1[21];
   char w_street_2[21];
   char w_city[21];
   char w_state[3];
   char w_zip[10];
   char d_street_1[21];
   char d_street_2[21];
   char d_city[21];
   char d_state[3];
   char d_zip[10];
   int  c_id;
   char c_first[17];
   char c_middle[3];
   char c_last[17];
   char c_street_1[21];
   char c_street_2[21];
   char c_city[21];
   char c_state[3];
   char c_zip[10];
   char c_phone[17];
   char c_since[11];
   char c_credit[3];
   double c_credit_lim;
   float c_discount;
   double c_balance;
   char c_data[201];
   char h_date[20];
   int retry;
};

struct paystruct {
   struct payinstruct payin;
   struct payoutstruct payout;
};


/* Order status */

struct ordinstruct {
   int w_id;
   int d_id;
   int c_id;
   int bylastname;
   char c_last[17];
};

struct ordoutstruct {
   int terror;
   int c_id;
   char c_last[17];
   char c_first[17];
   char c_middle[3];
   double c_balance;
   int o_id;
   char o_entry_d[20];
   int o_carrier_id;
   int o_ol_cnt;
   int ol_supply_w_id[15];
   int ol_i_id[15];
   int ol_quantity[15];
   float ol_amount[15];
/*
   char ol_delivery_d[15][11];
*/
   char ol_delivery_d[15][64];
   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
```

build_dir/loader/tpccload.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: tpccload.c 7030100.1 96/05/13 16:20:36 plai Generic<base> $ Copyr (c)
1993 Oracle";
#endif /* RCSID */

/*+=================================================================+
 |        Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
 |                    OPEN SYSTEMS PERFORMANCE GROUP                |
 |                         All Rights Reserved                      |
 +=================================================================+
 | FILENAME
 |    tpccload.c
 | DESCRIPTION
 |    Load or generate TPC-C database tables.
 |    Usage: tpccload -M <# of warehouses> [options]
 |              options: -A  load all tables
 |                       -w  load warehouse table
 |                       -d  load district table
 |                       -c  load customer table
 |                       -i  load item table
 |                       -s  load stock table (cluster around s_w_id)
 |                       -S  load stock table (cluster around s_i_id)
 |                       -h  load history table
 |                       -n  load new-order table
 |                       -o <oline file> load order and order-line table
 |                       -b <ware#>  beginning warehouse number
 |                       -e <ware#>  ending warehouse number
 |                       -j <item#>  beginning item number (with -S)
 |                       -k <item#>  ending item number (with -S)
 |                       -g  generate rows to standard output
 +=================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include "tpcc.h"

#define DISTARR 10/* district insert array size*/
#define CUSTARR 100/* customer insert array size*/
#define STOCARR 100/* stock insert array size*/
#define ITEMARR 100/* item insert array size*/
#define HISTARR 100             /* history insert array size    */
#define ORDEARR 100             /* order insert array size      */
#define NEWOARR 100             /* new order insert array size  */

#define DISTFAC 10/* max. disctrict id*/
#define CUSTFAC 3000/* max. customer id*/
#define STOCFAC 100000/* max. stock id*/
#define ITEMFAC 100000/* max. item id*/
#define HISTFAC 30000           /* history / warehouse  */
#define ORDEFAC 3000            /* order / district     */
#define NEWOFAC 900             /* new order / district */

#define C       0               /* constant in non-uniform dist. eqt. */
#define CNUM1   1               /* first constant in non-uniform dist. eqt. */
#define CNUM2   2               /* second constant in non-uniform dist. eqt. */
#define CNUM3   3               /* third constant in non-uniform dist. eqt. */

#define SEED    2               /* seed for random functions */

#define SQLTXTW "INSERT INTO warehouse (w_id, w_ytd, w_tax, w_name, w_street_1,
w_street_2, w_city, w_state, w_zip) VALUES (:w_id, 30000000, :w_tax, :w_name,
:w_street_1, \
   :w_street_2, :w_city, :w_state, :w_zip)"

#define SQLTXTD "INSERT INTO district (d_id, d_w_id, d_ytd, d_tax, d_next_o_id,
d_name, d_street_1, d_street_2, d_city, d_state, d_zip) VALUES (:d_id,
:d_w_id,3000000, :d_tax, \
     3001, :d_name, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip)"

#define SQLTXTC "INSERT INTO customer (C_ID, C_D_ID, C_W_ID, C_FIRST, C_MIDDLE,
C_LAST, C_STREET_1, C_STREET_2, C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT,
C_CREDIT_LIM, C_DISCOUNT, C_BALANCE, C_YTD_PAYMENT, C_PAYMENT_CNT, C_DELIVERY_CNT,
C_DATA) VALUES (:c_id, :c_d_id, :c_w_id, \
   :c_first, 'OE', :c_last, :c_street_1, :c_street_2, :c_city, :c_state, \
   :c_zip, :c_phone, SYSDATE, :c_credit, 5000000, :c_discount, -1000, 1000, 1, \
   0, :c_data)"

#define SQLTXTH "INSERT INTO history (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
h_date, h_amount, h_data) VALUES (:h_c_id, :h_c_d_id, :h_c_w_id, \
   :h_d_id, :h_w_id, SYSDATE, 1000, :h_data)"

#define SQLTXTS "INSERT INTO stock (s_i_id, s_w_id, s_quantity,s_dist_01, s_dist_02,
s_dist_03, s_dist_04, s_dist_05 , s_dist_06, s_dist_07, s_dist_08, s_dist_09,
s_dist_10, s_ytd, s_order_cnt, s_remote_cnt, s_data) \
VALUES (:s_i_id, :s_w_id, :s_quantity, \
   :s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05, :s_dist_06, \
   :s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10, 0, 0, 0, :s_data)" \

#define SQLTXTI "INSERT INTO item (I_ID,I_IM_ID,I_NAME,I_PRICE,I_DATA) VALUES (:i_id,
:i_im_id, :i_name, :i_price, \
   :i_data)"

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

#define SQLTXTO2 "INSERT INTO orders (O_ID,
O_D_ID,O_W_ID,O_C_ID,O_ENTRY_D,O_CARRIER_ID,O_OL_CNT,O_ALL_LOCAL) \
   VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
   SYSDATE, 11, :o_ol_cnt, 1)"

#define SQLTXTOL1 "INSERT INTO order_line (OL_O_ID, OL_D_ID, OL_W_ID, OL_NUMBER,
OL_DELIVERY_D, OL_I_ID, OL_SUPPLY_W_ID, OL_QUANTITY, OL_AMOUNT, OL_DIST_INFO) \
   VALUES (:ol_o_id, :ol_d_id, \
   :ol_w_id, :ol_number, SYSDATE, :ol_i_id, :ol_supply_w_id, 5, 0, \
   :ol_dist_info)"

#define SQLTXTOL2 "INSERT INTO order_line (OL_O_ID, OL_D_ID, OL_W_ID, OL_NUMBER,
OL_DELIVERY_D, OL_I_ID, OL_SUPPLY_W_ID, OL_QUANTITY, OL_AMOUNT, OL_DIST_INFO) \
   VALUES (:ol_o_id, :ol_d_id, \
   :ol_w_id, :ol_number, to_date('01-Jan-1811'), :ol_i_id, :ol_supply_w_id, 5,
:ol_amount, \
   :ol_dist_info)"

#define SQLTXTNO "INSERT INTO new_order (no_o_id, no_d_id, no_w_id) VALUES (:no_o_id,
:no_d_id, :no_w_id)"

ldadef tpclda;
csrdef curw, curd, curc, curh, curs, curi, curo1, curo2, curol1, curol2, curno;
unsigned long tpchda[256];

static char *lastname[] = {
   "BAR",
   "OUGHT",
   "ABLE",
   "PRI",
   "PRES",
   "ESE",
   "ANTI",
   "CALLY",
   "ATION",
   "EING"
};

char num9[10];
char num16[17];
char str2[3];
char str24[15][25];
int randperm3000[3000];



myusage()

{

   fprintf (stderr, "\n");
   fprintf (stderr, "Usage:\ttpccload -M <multiplier> [options]\n");
   fprintf (stderr, "options:\n");
   fprintf (stderr, "\t-A :\tload all tables\n");
   fprintf (stderr, "\t-w :\tload warehouse table\n");
   fprintf (stderr, "\t-d :\tload district table\n");
   fprintf (stderr, "\t-c :\tload customer table\n");
   fprintf (stderr, "\t-i :\tload item table\n");
   fprintf (stderr, "\t-s :\tload stock table (cluster around s_w_id)\n");
   fprintf (stderr, "\t-S :\tload stock table (cluster around s_i_id)\n");
   fprintf (stderr, "\t-h :\tload history table\n");
   fprintf (stderr, "\t-n :\tload new-order table\n");
   fprintf (stderr, "\t-o <oline file> :\tload order and order-line table\n");
   fprintf (stderr, "\t-b <ware#> :\tbeginning warehouse number\n");
   fprintf (stderr, "\t-e <ware#> :\tending warehouse number\n");
   fprintf (stderr, "\t-j <item#> :\tbeginning item number (with -S)\n");
   fprintf (stderr, "\t-k <item#> :\tending item number (with -S)\n");
   fprintf (stderr, "\t-g :\tgenerate rows to standard output\n");
   fprintf (stderr, "\n");
   exit(1);

}



errrpt (lda, cur)

csrdef *lda;
csrdef *cur;

{

   text msg[2048];

   if (cur->rc) {
      oerhms (lda, cur->rc, msg, 2048);
      fprintf (stderr, "TPC-C load error: %s\n", msg);
   }

}



quit ()
```

```
{

   if (oclose (&curw))
      errrpt (&tpclda, &curw);

   if (oclose (&curd))
      errrpt (&tpclda, &curd);

   if (oclose (&curc))
      errrpt (&tpclda, &curc);

   if (oclose (&curh))
      errrpt (&tpclda, &curh);

   if (oclose (&curs))
      errrpt (&tpclda, &curs);

   if (oclose (&curi))
      errrpt (&tpclda, &curi);

   if (oclose (&curo1))
      errrpt (&tpclda, &curo1);

   if (oclose (&curo2))
      errrpt (&tpclda, &curo2);

   if (oclose (&curol1))
      errrpt (&tpclda, &curol1);

   if (oclose (&curol2))
      errrpt (&tpclda, &curol2);

   if (oclose (&curno))
      errrpt (&tpclda, &curno);

   if (ologof (&tpclda))
      fprintf (stderr, "TPC-C load error: Error in logging off\n");

}


main (argc, argv)

int argc;
char *argv[];

{

   char *uid="dbbench/dbbench";
   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 (SEED);
   srand48 (SEED);
   initperm ();


/*-------------------------------------------------------------------+
 | Load the WAREHOUSE table.|
 +-------------------------------------------------------------------*/

   if (do_A || do_w) {
      nrows = eware - bware + 1;

      fprintf (stderr, "Loading/generating warehouse: w%d - w%d (%d rows)\n",
               bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      for (loop = bware; loop <= eware; loop++) {

         w_tax = (rand () % 2001);
         randstr (w_name, 6, 10);
         randstr (w_street_1, 10, 20);
         randstr (w_street_2, 10, 20);
         randstr (w_city, 10, 20);
         randstr (str2, 2, 2);
         randnum (num9, 9);
         num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

         if (gen) {
            printf ("%d 30000000 %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 %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 %s %s 0 0 0 %s\n",
                       sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
                       str24[3], str24[4], str24[5], str24[6], str24[7],
                       str24[8], str24[9], s_data[i]);
            }
            else {
               s_i_id[i] = sid;
               s_w_id[i] = swid;
               strncpy (s_dist_01[i], str24[0], 24);
               strncpy (s_dist_02[i], str24[1], 24);
               strncpy (s_dist_03[i], str24[2], 24);
               strncpy (s_dist_04[i], str24[3], 24);
               strncpy (s_dist_05[i], str24[4], 24);
               strncpy (s_dist_06[i], str24[5], 24);
               strncpy (s_dist_07[i], str24[6], 24);
               strncpy (s_dist_08[i], str24[7], 24);
               strncpy (s_dist_09[i], str24[8], 24);
               strncpy (s_dist_10[i], str24[9], 24);
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            if (oexn (&curs, STOCARR, 0)) {
               errrpt (&tpclda, &curs);
               orol (&tpclda);
               fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                        s_i_id[0]);
               quit ();
               exit (1);
            }
            else if (ocom (&tpclda)) {
               errrpt (&tpclda, &tpclda);
               orol (&tpclda);
               fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                        s_i_id[0]);
               quit ();
               exit (1);
            }
         }

         if ((++loopcount) % 50)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | Load the STOCK table (cluster around s_i_id).|
 +---------------------------------------------------------------*/

   if (do_S) {

      nrows = (eitem - bitem + 1) * (eware - bware + 1);

      fprintf (stderr, "Loading/generating stock: i%d - i%d, w%d - w%d (%d rows)\n
",
               bitem, eitem, bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      sid = bitem;
      swid = bware - 1;
      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < STOCARR; i++, row++) {
            if (++swid > eware) {          /* cheap mod */
               swid = bware;
               sid++;
            }
            s_quantity[i] = (rand () % 91) + 10;
            randstr (str24[0], 24, 24);
            randstr (str24[1], 24, 24);
            randstr (str24[2], 24, 24);
            randstr (str24[3], 24, 24);
            randstr (str24[4], 24, 24);
            randstr (str24[5], 24, 24);
            randstr (str24[6], 24, 24);
            randstr (str24[7], 24, 24);
            randstr (str24[8], 24, 24);
            randstr (str24[9], 24, 24);
            randdatastr (s_data[i], 26, 50);

            if (gen) {
               printf ("%d %d %d %s %s %s %s %s %s %s %s %s %s 0 0 0 %s\n",
                       sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
                       str24[3], str24[4], str24[5], str24[6], str24[7],
                       str24[8], str24[9], s_data[i]);
            }
            else {
               s_i_id[i] = sid;
               s_w_id[i] = swid;
               strncpy (s_dist_01[i], str24[0], 24);
               strncpy (s_dist_02[i], str24[1], 24);
               strncpy (s_dist_03[i], str24[2], 24);
               strncpy (s_dist_04[i], str24[3], 24);
               strncpy (s_dist_05[i], str24[4], 24);
               strncpy (s_dist_06[i], str24[5], 24);
               strncpy (s_dist_07[i], str24[6], 24);
               strncpy (s_dist_08[i], str24[7], 24);
               strncpy (s_dist_09[i], str24[8], 24);
               strncpy (s_dist_10[i], str24[9], 24);
            }
```

```
        }

        if (gen) {
           fflush (stdout);
        }
        else {
           if (oexn (&curs, STOCARR, 0)) {
              errrpt (&tpclda, &curs);
              orol (&tpclda);
              fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                       s_i_id[0]);
              quit ();
              exit (1);
           }
           else if (ocom (&tpclda)) {
              errrpt (&tpclda, &tpclda);
              orol (&tpclda);
              fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                       s_i_id[0]);
              quit ();
              exit (1);
           }
        }

        if ((++loopcount) % 50)
           fprintf (stderr, ".");
        else
           fprintf (stderr, " %d rows committed\n   ", row);
     }

     end_time = gettime ();
     end_cpu = getcpu ();
     fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
              nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | Load the HISTORY table.|
 +---------------------------------------------------------------*/

   if (do_A || do_h) {
      nrows = (eware - bware + 1) * HISTFAC;

      fprintf (stderr, "Loading/generating history: w%d - w%d (%d rows)\n   ",
               bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      cid = 0;
      cdid = 1;
      cwid = bware;
      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < HISTARR; i++, row++) {
            cid++;
            if (cid > CUSTFAC) {        /* cycle cust id */
               cid = 1;                 /* cheap mod */
               cdid++;                  /* shift district cycle */
               if (cdid > DISTFAC) {
                  cdid = 1;
                  cwid++;               /* shift warehouse cycle */
               }
            }
            h_c_id[i] = cid;
            h_d_id[i] = cdid;
            h_w_id[i] = cwid;
            randstr (h_data[i], 12, 24);
            if (gen) {
               printf ("%d %d %d %d %d %s 1000 %s\n", cid, cdid, cwid, cdid,
                       cwid, sdate, h_data[i]);
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            if (oexn (&curh, HISTARR, 0)) {
               errrpt (&tpclda, &curh);
               orol (&tpclda);
               fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                        h_w_id[0], h_d_id[0], h_c_id[0]);
               quit ();
               exit (1);
            }
            else if (ocom (&tpclda)) {
               errrpt (&tpclda, &tpclda);
               orol (&tpclda);
               fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                        h_w_id[0], h_d_id[0], h_c_id[0]);
               quit ();
               exit (1);
            }
         }

         if ((++loopcount) % 50)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | Load the ORDERS and ORDER-LINE table.|
 +---------------------------------------------------------------*/

   if (do_A || do_o) {
      nrows = (eware - bware + 1) * ORDEFAC * DISTFAC;

      fprintf (stderr, "Loading/generating orders and order-line: w%d - w%d (%d ord,
~%d ordl)\n   ",
               bware, eware, nrows, nrows * 10);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      cid = 0;
      cdid = 1;
      cwid = bware;
      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < ORDEARR; i++, row++) {
            cid++;
            if (cid > ORDEFAC) {        /* cycle cust id */
               cid = 1;                 /* cheap mod */
               cdid++;                  /* shift district cycle */
               if (cdid > DISTFAC) {
                  cdid = 1;
                  cwid++;               /* shift warehouse cycle */
               }
            }
            o_carrier_id[i] = rand () % 10 + 1;
            o_ol_cnt[i] = olcnt = rand () % 11 + 5;

            if (gen) {
               if (cid < 2101) {
                  printf ("%d %d %d %d %s %d %d 1\n", cid, cdid, cwid,
                          randperm3000[cid - 1], sdate,o_carrier_id[i],
                          o_ol_cnt[i]);
               }
               else {
             /* set carrierid to 11 instead of null */
                  printf ("%d %d %d %d %s 11 %d 1\n", cid, cdid, cwid,
                          randperm3000[cid - 1], sdate, o_ol_cnt[i]);
               }
            }
            else {
               o_id[i] = cid;
               o_d_id[i] = cdid;
               o_w_id[i] = cwid;
               o_c_id[i] = randperm3000[cid - 1];
            }

            for (j = 0; j < o_ol_cnt[i]; j++) {
               ol_i_id[j] = sid = lrand48 () % 100000 + 1;
               if (cid < 2101)
                  ol_amount[j] = 0;
               else
                  ol_amount[j] = (lrand48 () % 999999 + 1) ;
               randstr (str24[j], 24, 24);

               if (gen) {
                  if (cid < 2101) {
                     fprintf (olfp, "%d %d %d %d %s %d %d 5 %ld %s\n", cid,
                              cdid, cwid, j + 1, sdate, ol_i_id[j], cwid,
                              ol_amount[j], str24[j]);
                  }
                  else {
                   /* Insert a default date instead of null date */
                     fprintf (olfp, "%d %d %d %d 01-Jan-1811 %d %d 5 %ld %s\n", cid,
                              cdid, cwid, j + 1, ol_i_id[j], cwid,
                              ol_amount[j], str24[j]);
                  }
               }
               else {
                  ol_o_id[j] = cid;
                  ol_d_id[j] = cdid;
                  ol_w_id[j] = cwid;
                  ol_number[j] = j + 1;
                  ol_supply_w_id[j] = cwid;
                  strncpy (ol_dist_info[j], str24[j], 24);
               }
            }

            if (gen) {
               fflush (olfp);
            }
            else {
               if (cid < 2101) {
                  if (oexn (&curol1, olcnt, 0)) {
                     errrpt (&tpclda, &curol1);
                     orol (&tpclda);
                     fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                              cwid, cdid, cid);
                     quit ();
                     exit (1);
                  }
                  else if (ocom (&tpclda)) {
                     errrpt (&tpclda, &tpclda);
                     orol (&tpclda);
                     fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                            cwid, cdid, cid);
                     quit ();
                     exit (1);
                  }
               }
               else {
                  if (oexn (&curol2, olcnt, 0)) {
                     errrpt (&tpclda, &curol2);
                     orol (&tpclda);
                     fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                              cwid, cdid, cid);
                     quit ();
                     exit (1);
                  }
                  else if (ocom (&tpclda)) {
                     errrpt (&tpclda, &tpclda);
                     orol (&tpclda);
```

```
                    fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                             cwid, cdid, cid);
                    quit ();
                    exit (1);
                 }
              }
           }
        }

        if (gen) {
           fflush (stdout);
        }
        else {
           if (cid < 2101) {
              if (oexn (&curo1, ORDEARR, 0)) {
                 errrpt (&tpclda, &curo1);
                 orol (&tpclda);
                 fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
                          cwid, cdid, cid);
                 quit ();
                 exit (1);
              }
              else if (ocom (&tpclda)) {
                 errrpt (&tpclda, &tpclda);
                 orol (&tpclda);
                 fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
                          cwid, cdid, cid);
                 quit ();
                 exit (1);
              }
           }
           else {
              if (oexn (&curo2, ORDEARR, 0)) {
                 errrpt (&tpclda, &curo2);
                 orol (&tpclda);
                 fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
                          cwid, cdid, cid);
                 quit ();
                 exit (1);
              }
              else if (ocom (&tpclda)) {
                 errrpt (&tpclda, &tpclda);
                 orol (&tpclda);
                 fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
                          cwid, cdid, cid);
                 quit ();
                 exit (1);
              }
           }
        }

        if ((++loopcount) % 50)
           fprintf (stderr, ".");
        else
           fprintf (stderr, " %d orders committed\n   ", row);
     }

     end_time = gettime ();
     end_cpu = getcpu ();
     fprintf (stderr, "Done.  %d orders loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
              nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | Load the NEW-ORDER table.|
 +---------------------------------------------------------------*/

   if (do_A || do_n) {
      nrows = (eware - bware + 1) * NEWOFAC * DISTFAC;

      fprintf (stderr, "Loading/generating new-order: w%d - w%d (%d rows)\n   ",
               bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      cid = 0;
      cdid = 1;
      cwid = bware;
      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < NEWOARR; i++, row++) {
            cid++;
            if (cid > NEWOFAC) {
               cid = 1;
               cdid++;
               if (cdid > DISTFAC) {
                  cdid = 1;
                  cwid++;
               }
            }

            if (gen) {
               printf ("%d %d %d\n", cid + 2100, cdid, cwid);
            }
            else {
               no_o_id[i] = cid + 2100;
               no_d_id[i] = cdid;
               no_w_id[i] = cwid;
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            if (oexn (&curno, NEWOARR, 0)) {
               errrpt (&tpclda, &curno);
               orol (&tpclda);
               fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
                        cwid, cdid, cid + 2100);
               quit ();
               exit (1);
            }
            else if (ocom (&tpclda)) {
               errrpt (&tpclda, &tpclda);
               orol (&tpclda);
               fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
                        cwid, cdid, cid + 2100);
               quit ();
               exit (1);
            }
         }

         if ((++loopcount) % 45)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | clean up and exit.|
 +---------------------------------------------------------------*/

   if (olfp)
      fclose (olfp);
   if (!gen)
      quit ();
   exit (0);

}



initperm ()

{

   int i;
   int pos;
   int temp;

   /* init randperm3000 */

   for (i = 0; i < 3000; i++)
      randperm3000[i] = i + 1;
   for (i = 3000; i > 0; i--) {
      pos = rand () % i;
      temp = randperm3000[i - 1];
      randperm3000[i - 1] = randperm3000[pos];
      randperm3000[pos] = temp;
   }

}



randstr (str, x, y)

char *str;
int x;
int y;

{

   int i, j;
   int len;

   len = (rand () % (y - x + 1)) + x;
   for (i = 0; i < len; i++) {
      j = rand () % 62;
      if (j < 26)
         str[i] = (char) (j + 'a');
      else if (j < 52)
         str[i] = (char) (j - 26 + 'A');
      else
         str[i] = (char) (j - 52 + '0');
   }
   str[len] = '\0';

}



randdatastr (str, x, y)

char *str;
int x;
int y;

{

   int i, j;
   int len;
   int pos;

   len = (rand () % (y - x + 1)) + x;
   for (i = 0; i < len; i++) {
      j = rand () % 62;
      if (j < 26)
         str[i] = (char) (j + 'a');
      else if (j < 52)
         str[i] = (char) (j - 26 + 'A');
      else
         str[i] = (char) (j - 52 + '0');
   }
   str[len] = '\0';
   if ((rand () % 10) == 0) {
      pos = (rand () % (len - 8));
```

```
      str[pos] = 'O';
      str[pos + 1] = 'R';
      str[pos + 2] = 'I';
      str[pos + 3] = 'G';
      str[pos + 4] = 'I';
      str[pos + 5] = 'N';
      str[pos + 6] = 'A';
      str[pos + 7] = 'L';
   }

}



randnum (str, len)

char *str;
int len;

{

   int i;

   for (i = 0; i < len; i++)
      str[i] = (char) (rand () % 10 + '0');
   str[len] = '\0';

}



randlastname (str, id)

char *str;
int id;

{

   id = id % 1000;
   strcpy (str, lastname[id / 100]);
   strcat (str, lastname[(id / 10) % 10]);
   strcat (str, lastname[id % 10]);

}



NURand (A, x, y, cnum)

int A, x, y, cnum;

{

   int a, b;

   a = lrand48 () % (A + 1);
   b = (lrand48 () % (y - x + 1)) + x;
   return ((((a | b) + cnum) % (y - x + 1)) + x);

}



sysdate (sdate)

char *sdate;

{

   time_t tp;
   struct tm *tmptr;

   time (&tp);
   tmptr = localtime (&tp);
   strftime (sdate, 29, "%d-%b-%Y", tmptr);

}
```

Appendix C. Database Layout

This Appendix contains logical and physical assignment of data to disks.

Three types of Sun A5x00 Fiber Channel disk enclosures were used for the tested system - A5000, A5100 and A5200. These were organized into stacks of 3 or 4 enclosures, resulting in either a stack of 56 disks or a stack of 66 disks. The A5x00 enclosures provide data paths for 2 FC-AL (Fiber Channel Arbitrated Loop) loops providing access to both A & B ports of the disks which are all dual-ported. As the A5x00 enclosures provide 4 ports (loop A in+out, loop B in+out), the enclosures were daisy chained within a stack and one node of the cluster was connected at each end of both loops as shown in the figure below.. No hubs of switches were used for connecting the storage:

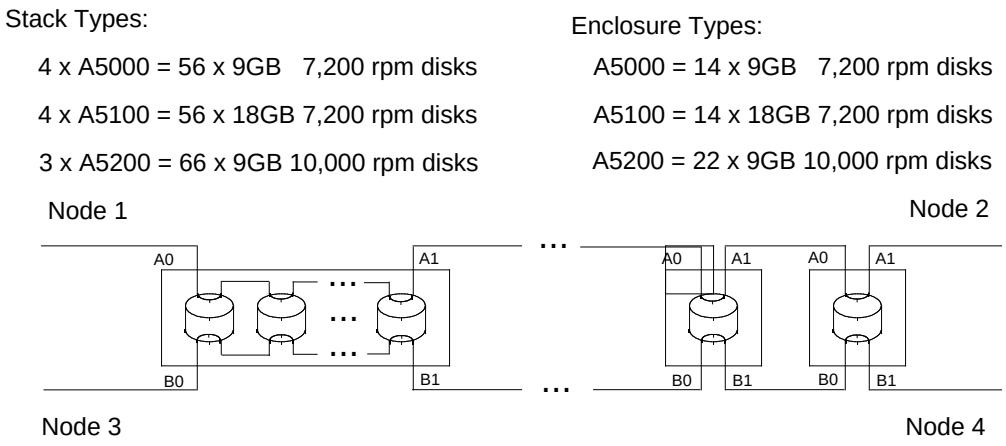


Figure 1 A loop of StorEdge Arrays

Oracle redo logs were assigned to two stacks of 4 * A5100s, mirrored from one stack to the other and each node logs were striped over disjoint sets of disks. Oracle datafiles were assigned to 12 stacks of 4*A500s and 9 stacks of 3 * A5200s. The datafiles logically mapped onto a single stack and were striped evenly across the disks.

Table 6 on page 104 in provides the logical assignment of files to array stacks. Table 7 on page 106 in provides the physical assignment of datafiles to disk stacks as tested.Table 8 on page 107 in provides the proposed logical and physical assignment of datafiles for the priced system.

TABLE 6. Tested System, Logical File Assignment

Stack	Stack Type	No. of Disks	Disk Size	Contents
log	4xA5100	56	18GB	Log devices (8hr)
log mirror	4xA5100	56	18GB	Log mirrors (8hr)
cust-0	3xA5200	65	9GB	sys000 roll002-3 cl_ix000 cl_ix002-3 cl_ix005 cust003-4 cust006-7 cust010 cust012 cust014-6 cust020 cust022 cust025 cust027 cust032 cust036-7 cust039 cust042 cust045 cust046 cust048 cust050-2 cust054-5 cust057-8 cust060 cust062 cust065 cust067 cust070-1

TABLE 6. Tested System, Logical File Assignment

Stack	Stack Type	No. of Disks	Disk Size	Contents
cust-1	3xA5200	65	9GB	ctrl000 roll000-1 cl_ix001 cl_ix004 cl_ix006-7 cust000-2 cust008-9 cust013 cust018-9 cust021 cust024 cust026 cust028-31 cust033-35 cust038 cust040-1 cust043-4 cust049 cust053 cust056 cust059 cust061 cust063-4 cust066 cust068-9
stk-0	3xA5200	65	9GB	c_ix000-1 item000 s_ix003-4 stk029 stk041 stk044-5 stk047 stk052 stk055 stk060 stk075 stk086 stk099 stk104
stk-1	3xA5200	65	9GB	c_ix002 item001 s_ix002 s_ix005 s_ix016 stk006 stk017-8 stk031 stk036 stk039 stk050 stk053 stk069-70 stk077 stk092
stk-2	3xA5200	65	9GB	c_ix003 item002 s_ix001 s_ix006 s_ix015 stk003 stk012 stk023 stk034 stk042 stk046 stk049 stk054 stk057 stk059 stk081 stk083
stk-3	3xA5200	65	9GB	c_ix004 item003 s_ix000 s_ix007 s_ix014 stk011 stk015 stk016 stk019 stk038 stk048 stk056 stk085 stk089-91 stk097
stk-4	3xA5200	65	9GB	c_ix005 cust047 s_ix008 s_ix013 stk004-5 stk014 stk020 stk024-5 stk030 stk051 stk078 stk095 stk098 stk102 wd000
stk-5	3xA5200	65	9GB	cust005 cust023 s_ix009 s_ix012 stk000-1 stk032 stk040 stk061-3 stk087 stk093 stk101 stk103 wd001
stk-6	3xA5200	65	9GB	cust011 cust017 s_ix010-1 stk002 stk009 stk013 stk022 stk043 stk068 stk080 stk084 stk096 stk105 wd002-3
general-A0	4xA5000	55	9GB	hist003 hist005 oline000-1 oline011 oline016-7 oline020 ord000 ord2_ix000 ord_alt005 stk008 stk021
general-A1	4xA5000	55	9GB	hist001 hist006 oline004 oline018 oline022 oline026 oline032-3 ord002 ord2_ix002 ord_alt004 stk010 stk037
general-A2	4xA5000	55	9GB	hist000 oline002 oline006 oline010 oline021 oline023 oline030 ord003 ord2_ix001 ord_alt002 ord_ix001 stk007 stk026
general-A3	4xA5000	55	9GB	hist004 oline009 oline015 oline019 oline024 oline031 oline035 ord004 ord2_ix003 ord_alt003 ord_ix003 stk033 stk066
general-A4	4xA5000	55	9GB	hist002 newo001 oline005 oline007 oline012 oline014 oline027-8 ord005 ord_alt001 ord_ix000 stk035 stk065 stk073
general-A5	4xA5000	55	9GB	hist007 newo000 oline003 oline008 oline013 oline025 oline029 oline034 ord001 ord_alt000 ord_ix002 stk071-2 stk107
general-B0	4xA5000	55	9GB	hist010-1 oline037 oline049 oline061-3 oline070 ord2_ix004 ord_alt008 ord_alt010 stk082 stk100
general-B1	4xA5000	55	9GB	hist013 oline040-1 oline043 oline046 oline054 oline069 ord006 ord2_ix006 ord_alt011 ord_ix007 stk027 stk058
general-B2	4xA5000	55	9GB	hist014 oline044 oline047-8 oline052 oline056 oline068 ord007 ord2_ix005 ord_alt009 ord_ix005 stk064 stk067
general-B3	4xA5000	55	9GB	hist008 oline042 oline051 oline057-60 ord008 ord2_ix007 ord_alt007 ord_ix006 stk028 stk074
general-B4	4xA5000	55	9GB	hist012 oline038-9 oline045 oline053 oline064 oline071 ord009 ord_alt006 ord_ix004 stk079 stk088 stk106

TABLE 6. Tested System, Logical File Assignment

Stack	Stack Type	No. of Disks	Disk Size	Contents
general-B5	4xA5000	55	9GB	hist009 hist015 newo002 newo003 oline036 oline050 oline055 oline065 oline066 oline067 ord010 ord011 stk076 stk094
-	-	-	-	-
		1357		

TABLE 7. Tested System, Physical File Assignment

Stack	Stack Type	No. of Disks	Disk Size	Contents
log	4xA5100	56	18GB	Log devices (8hr)
log mirror	4xA5100	56	18GB	Log mirrors (8hr)
cust-0	3xA5200	65	9GB	sys000 roll002-3 cl_ix000 cl_ix002-3 cl_ix005 cust003-4 cust006-7 cust010 cust012 cust014-6 cust020 cust022 cust025 cust027
				cust032 cust036-7 cust039 cust042 cust045 cust046 cust048 cust050-2 cust054-5 cust057-8 cust060 cust062 cust065 cust067 cust070-1
cust-1	3xA5200	65	9GB	ctrl000 roll000-1 cl_ix001 cl_ix004 cl_ix006-7 cust000-2 cust008-9 cust013 cust018-9 cust021 cust024 cust026 cust028-31
				cust033-35 cust038 cust040-1 cust043-4 cust049 cust053 cust056 cust059 cust061 cust063-4 cust066 cust068-9
stk-0	3xA5200	65	9GB	c_ix000-1 item000 s_ix003-4 stk029 stk041 stk044-5 stk047 stk052 stk055 stk060 stk075 stk086 stk099 stk104
stk-1	3xA5200	65	9GB	c_ix002 item001 s_ix002 s_ix005 s_ix016 stk006 stk017-8 stk031 stk036 stk039 stk050 stk053 stk069-70 stk077 stk092
stk-2	3xA5200	65	9GB	c_ix003 item002 s_ix001 s_ix006 s_ix015 stk003 stk012 stk023 stk034 stk042 stk046 stk049 stk054 stk057 stk059 stk081 stk083
stk-3	3xA5200	65	9GB	c_ix004 item003 s_ix000 s_ix007 s_ix014 stk011 stk015 stk016 stk019 stk038 stk048 stk056 stk085 stk089-91 stk097
stk-4	3xA5200	65	9GB	c_ix005 cust047 s_ix008 s_ix013 stk004-5 stk014 stk020 stk024-5 stk030 stk051 stk078 stk095 stk098 stk102 wd000
stk-5	3xA5200	65	9GB	cust005 cust023 s_ix009 s_ix012 stk000-1 stk032 stk040 stk061-3 stk087 stk093 stk101 stk103 wd001
stk-6	3xA5200	65	9GB	cust011 cust017 s_ix010-1 stk002 stk009 stk013 stk022 stk043 stk068 stk080 stk084 stk096 stk105 wd002-3
general-A0	4xA5000	55	9GB	hist003 hist005 oline000-1 oline011 oline016-7 oline020 ord000 ord2_ix000 ord_alt005 stk008 stk021
general-A1	4xA5000	54	9GB	hist001 hist006 oline004 oline018 oline022 oline026 oline032-3 ord002 ord2_ix002 ord_alt004 stk010 stk037
general-A2	4xA5000	55	9GB	hist000 oline002 oline006 oline010 oline021 oline023 oline030 ord003 ord2_ix001 ord_alt002 ord_ix001 stk007 stk026

TABLE 7. Tested System, Physical File Assignment

Stack	Stack Type	No. of Disks	Disk Size	Contents
general-A3	4xA5000	55	9GB	hist004 oline009 oline015 oline019 oline024 oline031 oline035 ord004 ord2_ix003 ord_alt003 ord_ix003 stk033 stk066
general-A4	4xA5000	54	9GB	hist002 newo001 oline005 oline007 oline012 oline014 oline027-8 ord005 ord_alt001 ord_ix000 stk035 stk065 stk073
general-A5	4xA5000	54	9GB	hist007 newo000 oline003 oline008 oline013 oline025 oline029 oline034 ord001 ord_alt000 ord_ix002 stk071-2 stk107
general-B0	4xA5000	54	9GB	hist010-1 oline037 oline049 oline061-3 oline070 ord2_ix004 ord_alt008 ord_alt010 stk082 stk100
general-B1	4xA5000	55	9GB	hist013 oline040-1 oline043 oline046 oline054 oline069 ord006 ord2_ix006 ord_alt011 ord_ix007 stk027 stk058
general-B2	4xA5000	53	9GB	hist014 oline044 oline047-8 oline052 oline056 oline068 ord007 ord2_ix005 ord_alt009 ord_ix005 stk064 stk067
general-B3	4xA5000	55	9GB	hist008 oline042 oline051 oline057-60 ord008 ord2_ix007 ord_alt007 ord_ix006 stk028 stk074
general-B4	4xA5000	54	9GB	hist012 oline038-9 oline045 oline053 oline064 oline071 ord009 ord_alt006 ord_ix004 stk079 stk088 stk106
gen.B5 +/-	4xA5000	44	9GB	3: spares; 41: hist009 hist015 newo002 newo003 oline036 oline050 oline055 oline065 oline066 oline067 ord010 ord011 stk076 stk094
general-B5	1xA5000	14	9GB	14: hist009 hist015 newo002 newo003 oline036 oline050 oline055 oline065 oline066 oline067 ord010 ord011 stk076 stk094
spares	4xA5000	4	9GB	4: spares
-	-	-	-	-
		1357		

TABLE 8. Priced System, Logical/Physical File Assignment

Stack	Stack Type	No. of Disks	Disk Size	Contents
log	4xA5100	56	18GB	Log devices (8hr)
log mirror	4xA5100	56	18GB	Log mirrors (8hr)
cust-0	3xA5200	66	9GB	growth + sys000 roll002-3 cl_ix000 cl_ix002-3 cl_ix005 cust003-4 cust006-7 cust010 cust012 cust014-6 cust020 cust022 cust025 cust027 cust032 cust036-7 cust039 cust042 cust045 cust046 cust048 cust050-2 cust054-5 cust057-8 cust060 cust062 cust065 cust067 cust070-1
cust-1	3xA5200	66	9GB	growth + ctrl000 roll000-1 cl_ix001 cl_ix004 cl_ix006-7 cust000-2 cust008-9 cust013 cust018-9 cust021 cust024 cust026 cust028-31 cust033-35 cust038 cust040-1 cust043-4 cust049 cust053 cust056 cust059 cust061 cust063-4 cust066 cust068-9
stk-0	3xA5200	66	9GB	growth + c_ix000-1 item000 s_ix003-4 stk029 stk041 stk044-5 stk047 stk052 stk055 stk060 stk075 stk086 stk099 stk104

TABLE 8. Priced System, Logical/Physical File Assignment

Stack	Stack Type	No. of Disks	Disk Size	Contents
stk-1	3xA5200	66	9GB	growth + c_ix002 item001 s_ix002 s_ix005 s_ix016 stk006 stk017-8 stk031 stk036 stk039 stk050 stk053 stk069-70 stk077 stk092
stk-2	3xA5200	66	9GB	growth + c_ix003 item002 s_ix001 s_ix006 s_ix015 stk003 stk012 stk023 stk034 stk042 stk046 stk049 stk054 stk057 stk059 stk081 stk083
stk-3	3xA5200	66	9GB	growth + c_ix004 item003 s_ix000 s_ix007 s_ix014 stk011 stk015 stk016 stk019 stk038 stk048 stk056 stk085 stk089-91 stk097
stk-4	3xA5200	66	9GB	growth + c_ix005 cust047 s_ix008 s_ix013 stk004-5 stk014 stk020 stk024-5 stk030 stk051 stk078 stk095 stk098 stk102 wd000
stk-5	3xA5200	66	9GB	growth + cust005 cust023 s_ix009 s_ix012 stk000-1 stk032 stk040 stk061-3 stk087 stk093 stk101 stk103 wd001
stk-6	3xA5200	66	9GB	growth + cust011 cust017 s_ix010-1 stk002 stk009 stk013 stk022 stk043 stk068 stk080 stk084 stk096 stk105 wd002-3
general-A0	3xA5200	66	9GB	growth + 0.6*(tested general A0+A1)
general-A1	2xA5200	44	9GB	growth + 0.4*(tested general A0+A1)
general-A2	3xA5200	66	9GB	0.6*(tested general A2+A3)
general-A3	2xA5200	44	9GB	0.4*(tested general A2+A3)
general-A4	3xA5200	66	9GB	growth + 0.6*(tested general A4+A5)
general-A5	2xA5200	44	9GB	growth + 0.4*(tested general A4+A5)
general-B0	3xA5200	66	9GB	growth + 0.6*(tested general B0+B1)
general-B1	2xA5200	44	9GB	growth + 0.4*(tested general B0+B1)
general-B2	3xA5200	66	9GB	growth + 0.6*(tested general B2+B3)
general-B3	2xA5200	44	9GB	growth + 0.4*(tested general B2+B3)
general-B4	3xA5200	66	9GB	growth + tested general-B4
gen.B5 +/-	2xA5200	44	9GB	tested gen.B5 +/-
general-B5	1xA5200	22	9GB	growth + tested general-B5
spares	2xA5200	44	9GB	growth + tested spares
-	-	-	-	-
		1432		

Appendix D. Tunable Parameters

This Appendix contains the configuration information for the operating system, the RDBMS and Tuxedo.

Operating System Configuration Values

The Solaris kernel configuration parameters set in the file /etc/system are given below.

Solaris Configuration File for Sun Enterprise 6500 Cluster

```
set ssd:ssd_io_time=30
set ssd:ssd_retry_count=15
* vxvm_START
forceload: drv/vxio
forceload: drv/vxspec
* vxvm_END
set shmsys:shminfo_shmmax=0xffffffff
set shmsys:shminfo_shmseg=32
set maxphys=1048576
set vxio:vol_maxio=2048
set vxio:voliomem_base_memory=1048576
set vxio:voliomem_kvmap_size=16777216
set vxio:voliomem_max_memory=15728640
set vxio:vol_default_iodelay=0
set vxio:vol_maxkiocount=8192
set vxio:vol_maxparallelio=1024
set enable_grp_ism=1
set semsys:seminfo_semmap=4096
set semsys:seminfo_semmni=4096
set semsys:seminfo_semmns=4096
set semsys:seminfo_semmnu=4096
set semsys:seminfo_semume=4096
set semsys:seminfo_semmsl=4096
set semsys:seminfo_semopm=4096
set semsys:seminfo_semvmx = 32767
set semsys:seminfo_semaem = 16384
```

Solaris configuration file for the client systems

```
set pt_cnt=4096
set shmsys:shminfo_shmmax=0xffffffff
set shmsys:shminfo_shmseg=600
set shmsys:shminfo_shmmni=10
set msgsys:msginfo_msgmni=4096
set msgsys:msginfo_msgmax=2048
set msgsys:msginfo_msgmnb=800000
set msgsys:msginfo_msgmap=200000
set msgsys:msginfo_msgseg=10000
set msgsys:msginfo_msgssz=2048
set msgsys:msginfo_msgtql=5000
set semsys:seminfo_semmns=5000
set semsys:seminfo_semmni=5000
set semsys:seminfo_semmsl=5000
set semsys:seminfo_semmap=5000
set semsys:seminfo_semume=1
set semsys:seminfo_semmnu=5000
*set tune_t_fsflushr = 50
set autoup = 300
```

Common Oracle initialization File

```
_bump_highwater_mark_count          = 100
_cr_server                          = false
_db_aging_stay_count                = 1
_db_block_hash_buckets              = 466667
_db_file_noncontig_mblock_read_count = 1
_log_simultaneous_copies            = 128
_spin_count                         = 8000
buffer_pool_recycle                 = (buffers:30000, lru_latches:4)
compatible                          = 8.1.4
control_files                       = (/oradev/ctrl1000)
cursor_space_for_time               = TRUE
db_block_buffers                    = 1250000
db_block_checking                   = false
db_block_lru_latches                = 21
db_block_max_dirty_target           = 600000
db_block_size                       = 2048
db_file_multiblock_read_count       = 1
db_files                            = 450
db_name                             = tpcc
db_writer_processes                 = 3
distributed_transactions            = 0
dml_locks                           = 500
enqueue_resources                   = 6000
fast_start_io_target                = 0
gc_defer_time                       = 0
gc_releasable_locks                 = 600000
hash_join_enabled                   = FALSE
java_pool_size                      = "1000K"
log_archive_start                   = FALSE
log_buffer                          = 4194304
log_checkpoint_interval             = 1000000000
log_checkpoints_to_alert            = TRUE
max_dump_file_size                  = unlimited
max_rollback_segments               = 420
open_cursors                        = 200
oracle_trace_collection_size        = 51200000
parallel_max_servers                = 100
parallel_server                     = true
processes                           = 1000
recovery_parallelism                = 50
replication_dependency_tracking     = FALSE
sessions                            = 1000
shared_pool_reserved_size           = 1024000
shared_pool_size                    = 280000000
sort_area_size                      = 524288
timed_statistics                    = false
transactions                        = 1000
transactions_per_rollback_segment   = 1
gc_files_to_locks="
1,359=1000each:\
6-8,15-30,32-47,49-64,66-77=5152!480each:\
10-14=1each:\
31=4859!509each:\
9,48=6776!365each:\
65=6137!403each:\
86,88,90,92,94,96,98,100,102,104,106,108,110,111,113,115,117,119=3000each:\
121,123-132,134,135,137,139,141,143,145,147,149=3000each:\
151,153,155,159,161,163,165,167,169,171,173,175,177,179,181=3000each:\
87,89,91,93,95,97,99,101,103,105,107,112,114,116,118,120,122=5000each:\
136,138,140,142,144,146,148,150,152,154,156,158,160,162,164,166,168=5000each:\
170,172,174,176,178,180=5000each:\
78-84,190-193=1each:\
109=37888!60each:\
133=20667!110each:\
85=10000each:\
157=20667!110each:\
182-189=1each:\
194-197=1each:\
198-201=1each:\
202-205=1each:\
206-209=1each:\
210-215,217-219,221,224-230,232,233,235,237,287-292,295,298-305,307,309-310,312-326,333-337,339-342,344-352=1each:\
216,222,234,286,293,296,308,311,327,329-331=1each:\
220,223,231,236,294,297,306,328,332,338,343,353-357=1each:\
238-253=1each:\
254-285=1each:\
358,360-393=1each"
```

Oracle initialization file for Node 1

```
ifile= /dbbench/vendors/oracle/TPCSO/scripts/p_common.ora
instance_number= 1
LM_PROCS= 1500
LM_LOCKS= 1600000
LM_RESS= 900000
thread= 1
rollback_segments
=(t0,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,t265,t266,t267,t268,t269,t270,t271,t272,t273,t274,t275,t276,t277,t278,t279,t280,t281,t282,t283,t284,t285,t286,t287,t288,t289,t290,t291,t292,t293,t294,t295,t296,t297,t298,t299)
```

Oracle initialization file for Node 2

```
ifile= /dbbench/vendors/oracle/TPCSO/scripts/p_common.ora
instance_number= 2
LM_PROCS        = 1500
LM_LOCKS        = 1600000
LM_RESS         = 900000
thread= 2
rollback_segments
=(t300,t301,t302,t303,t304,t305,t306,t307,t308,t309,t310,t311,t312,t313,t314,t315,t316,t317,t318,t319,t320,t321,t322,t323,t324,t325,t326,t327,t328,t329,t330,t331,t332,t333,t334,t335,t336,t337,t338,t339,t340,t341,t342,t343,t344,t345,t346,t347,t348,t349,t350,t351,t352,t353,t354,t355,t356,t357,t358,t359,t360,t361,t362,t363,t364,t365,t366,t367,t368,t369,t370,t371,t372,t373,t374,t375,t376,t377,t378,t379,t380,t381,t382,t383,t384,t385,t386,t387,t388,t389,t390,t391,t392,t393,t394,t395,t396,t397,t398,t399,t400,\
t401,t402,t403,t404,t405,t406,t407,t408,t409,t410,t411,t412,t413,t414,t415,t416,t417,t418,t419,t420,t421,t422,t423,t424,t425,t426,t427,t428,t429,t430,t431,t432,t433,t434,t435,t436,t437,t438,t439,t440,t441,t442,t443,t444,t445,t446,t447,t448,t449,t450,t451,t452,t453,t454,t455,t456,t457,t458,t459,t460,t461,t462,t463,t464,t465,t466,t467,t468,t469,t470,t471,t472,t473,t474,t475,t476,t477,t478,t479,t480,t481,t482,t483,t484,t485,t486,t487,t488,t489,t490,t491,t492,t493,t494,t495,t496,t497,t498,t499,t500, \
t501,t502,t503,t504,t505,t506,t507,t508,t509,t510,t511,t512,t513,t514,t515,t516,t517,t518,t519,t520,t521,t522,t523,t524,t525,t526,t527,t528,t529,t530,t531,t532,t533,t534,t535,t536,t537,t538,t539,t540,t541,t542,t543,t544,t545,t546,t547,t548,t549,t550,t551,t552,t553,t554,t555,t556,t557,t558,t559,t560,t561,t562,t563,t564,t565,t566,t567,t568,t569,t570,t571,t572,t573,t574,t575,t576,t577,t578,t579,t580,t581,t582,t583,t584,t585,t586,t587,t588,t589,t590,t591,t592,t593,t594,t595,t596,t597,t598,t599)
```

Oracle initialization file for Node 3

```
ifile= /dbbench/vendors/oracle/TPCSO/scripts/p_common.ora
instance_number= 3
```

```
LM_PROCS        = 1500
LM_LOCKS        = 1600000
LM_RESS         = 900000
thread= 3
rollback_segments
=(t600,t601,t602,t603,t604,t605,t606,t607,t608,t609,t610,t611,t612,t613,t614,t615,t61
6,t617,t618,t619,t620,t621,t622,t623,t624,t625,t626,t627,t628,t629,t630,t631,t632,t63
3,t634,t635,t636,t637,t638,t639,t640,t641,t642,t643,t644,t645,t646,t647,t648,t649,t65
0,t651,t652,t653,t654,t655,t656,t657,t658,t659,t660,t661,t662,t663,t664,t665,t666,t66
7,t668,t669,t670,t671,t672,t673,t674,t675,t676,t677,t678,t679,t680,t681,t682,t683,t68
4,t685,t686,t687,t688,t689,t690,t691,t692,t693,t694,t695,t696,t697,t698,t699,t700,\
t701,t702,t703,t704,t705,t706,t707,t708,t709,t710,t711,t712,t713,t714,t715,t716,t717,
t718,t719,t720,t721,t722,t723,t724,t725,t726,t727,t728,t729,t730,t731,t732,t733,t734,
t735,t736,t737,t738,t739,t740,t741,t742,t743,t744,t745,t746,t747,t748,t749,t750,t751,
t752,t753,t754,t755,t756,t757,t758,t759,t760,t761,t762,t763,t764,t765,t766,t767,t768,
t769,t770,t771,t772,t773,t774,t775,t776,t777,t778,t779,t780,t781,t782,t783,t784,t785,
t786,t787,t788,t789,t790,t791,t792,t793,t794,t795,t796,t797,t798,t799,t800, \
t801,t802,t803,t804,t805,t806,t807,t808,t809,t810,t811,t812,t813,t814,t815,t816,t817,
t818,t819,t820,t821,t822,t823,t824,t825,t826,t827,t828,t829,t830,t831,t832,t833,t834,
t835,t836,t837,t838,t839,t840,t841,t842,t843,t844,t845,t846,t847,t848,t849,t850,t851,
t852,t853,t854,t855,t856,t857,t858,t859,t860,t861,t862,t863,t864,t865,t866,t867,t868,
t869,t870,t871,t872,t873,t874,t875,t876,t877,t878,t879,t880,t881,t882,t883,t884,t885,
t886,t887,t888,t889,t890,t891,t892,t893,t894,t895,t896,t897,t898,t899)
```

Oracle initialization file forNode 4

```
ifile= /dbbench/vendors/oracle/TPCSO/scripts/p_common.ora
instance_number= 4
LM_PROCS        = 1500
LM_LOCKS        = 1600000
LM_RESS         = 900000
thread= 4
rollback_segments
=(t900,t901,t902,t903,t904,t905,t906,t907,t908,t909,t910,t911,t912,t913,t914,t915,t91
6,t917,t918,t919,t920,t921,t922,t923,t924,t925,t926,t927,t928,t929,t930,t931,t932,t93
3,t934,t935,t936,t937,t938,t939,t940,t941,t942,t943,t944,t945,t946,t947,t948,t949,t95
0,t951,t952,t953,t954,t955,t956,t957,t958,t959,t960,t961,t962,t963,t964,t965,t966,t96
7,t968,t969,t970,t971,t972,t973,t974,t975,t976,t977,t978,t979,t980,t981,t982,t983,t98
4,t985,t986,t987,t988,t989,t990,t991,t992,t993,t994,t995,t996,t997,t998,t999,t1000,\
t1001,t1002,t1003,t1004,t1005,t1006,t1007,t1008,t1009,t1010,t1011,t1012,t1013,t1014,t
1015,t1016,t1017,t1018,t1019,t1020,t1021,t1022,t1023,t1024,t1025,t1026,t1027,t1028,t1
029,t1030,t1031,t1032,t1033,t1034,t1035,t1036,t1037,t1038,t1039,t1040,t1041,t1042,t10
43,t1044,t1045,t1046,t1047,t1048,t1049,t1050,t1051,t1052,t1053,t1054,t1055,t1056,t105
7,t1058,t1059,t1060,t1061,t1062,t1063,t1064,t1065,t1066,t1067,t1068,t1069,t1070,t1071
,t1072,t1073,t1074,t1075,t1076,t1077,t1078,t1079,t1080,t1081,t1082,t1083,t1084,t1085,
t1086,t1087,t1088,t1089,t1090,t1091,t1092,t1093,t1094,t1095,t1096,t1097,t1098,t1099,t
1100, \
t1101,t1102,t1103,t1104,t1105,t1106,t1107,t1108,t1109,t1110,t1111,t1112,t1113,t1114,t
1115,t1116,t1117,t1118,t1119,t1120,t1121,t1122,t1123,t1124,t1125,t1126,t1127,t1128,t1
129,t1130,t1131,t1132,t1133,t1134,t1135,t1136,t1137,t1138,t1139,t1140,t1141,t1142,t11
43,t1144,t1145,t1146,t1147,t1148,t1149,t1150,t1151,t1152,t1153,t1154,t1155,t1156,t115
7,t1158,t1159,t1160,t1161,t1162,t1163,t1164,t1165,t1166,t1167,t1168,t1169,t1170,t1171
,t1172,t1173,t1174,t1175,t1176,t1177,t1178,t1179,t1180,t1181,t1182,t1183,t1184,t1185,
t1186,t1187,t1188,t1189,t1190,t1191,t1192,t1193,t1194,t1195,t1196,t1197,t1198,t1199)
```

Tuxedo initialization file

```
*RESOURCES
IPCKEY 40001
MASTER c1
PERM 0666
MODEL SHM
LDBAL Y
MAXACCESSERS 4000
MAXSERVERS 300
MAXSERVICES 100
SCANUNIT 20
SANITYSCAN 5
BLOCKTIME 180
BBLQUERY 60

*MACHINES
c1 LMID=c1
 ROOTDIR="/export/home/tuxedo"
 APPDIR="/export/home/dbbench/tuxedo"
 TUXCONFIG="/export/home/dbbench/tuxedo/tuxconfig.c1"
# ULOGPFX is prefix of logfile where tuxedo logs its actions
ULOGPFX="/export/home/dbbench/tuxedo/ULOGc1"

*GROUPS
group1 LMID=c1 GRPNO=1

*SERVERS

tpcc_srv_del SRVGRP=group1 SRVID=1 RQADDR=delq1 REPLYQ=N CLOPT="-A  -- 1"
tpcc_srv_del SRVGRP=group1 SRVID=2 RQADDR=delq2 REPLYQ=N CLOPT="-A  -- 2"
tpcc_srv_del SRVGRP=group1 SRVID=3 RQADDR=delq3 REPLYQ=N CLOPT="-A  -- 3"
tpcc_srv_del SRVGRP=group1 SRVID=4 RQADDR=delq4 REPLYQ=N CLOPT="-A  -- 4"
tpcc_srv_del SRVGRP=group1 SRVID=5 RQADDR=delq5 REPLYQ=N CLOPT="-A  -- 5"
tpcc_srv_del SRVGRP=group1 SRVID=6 RQADDR=delq6 REPLYQ=N CLOPT="-A  -- 6"
tpcc_srv_del SRVGRP=group1 SRVID=7 RQADDR=delq7 REPLYQ=N CLOPT="-A  -- 7"
tpcc_srv_del SRVGRP=group1 SRVID=8 RQADDR=delq8 REPLYQ=N CLOPT="-A  -- 8"
tpcc_srv_del SRVGRP=group1 SRVID=9 RQADDR=delq9 REPLYQ=N CLOPT="-A  -- 9"
tpcc_srv_del SRVGRP=group1 SRVID=10 RQADDR=delq10 REPLYQ=N CLOPT="-A  -- 10"
tpcc_srv_del SRVGRP=group1 SRVID=11 RQADDR=delq11 REPLYQ=N CLOPT="-A  -- 11"
tpcc_srv_del SRVGRP=group1 SRVID=12 RQADDR=delq12 REPLYQ=N CLOPT="-A  -- 12"

tpcc_srv_newo SRVGRP=group1 SRVID=13 RQADDR=newoq1 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=14 RQADDR=newoq2 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=15 RQADDR=newoq3 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=16 RQADDR=newoq4 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=17 RQADDR=newoq5 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=18 RQADDR=newoq6 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=19 RQADDR=newoq7 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=20 RQADDR=newoq8 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=21 RQADDR=newoq9 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=22 RQADDR=newoq10 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=23 RQADDR=newoq11 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=24 RQADDR=newoq12 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=25 RQADDR=newoq13 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=26 RQADDR=newoq14 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=27 RQADDR=newoq15 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=28 RQADDR=newoq16 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=29 RQADDR=newoq17 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=30 RQADDR=newoq18 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=31 RQADDR=newoq19 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=32 RQADDR=newoq20 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=33 RQADDR=newoq21 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=34 RQADDR=newoq22 REPLYQ=N
tpcc_srv_newo SRVGRP=group1 SRVID=35 RQADDR=newoq23 REPLYQ=N

tpcc_srv_ords SRVGRP=group1 SRVID=36 RQADDR=ordsq1 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=37 RQADDR=ordsq2 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=38 RQADDR=ordsq3 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=39 RQADDR=ordsq4 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=40 RQADDR=ordsq5 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=41 RQADDR=ordsq6 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=42 RQADDR=ordsq7 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=43 RQADDR=ordsq8 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=44 RQADDR=ordsq9 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=45 RQADDR=ordsq10 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=46 RQADDR=ordsq11 REPLYQ=N
tpcc_srv_ords SRVGRP=group1 SRVID=47 RQADDR=ordsq12 REPLYQ=N

tpcc_srv_paym SRVGRP=group1 SRVID=48 RQADDR=paymq1 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=49 RQADDR=paymq2 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=50 RQADDR=paymq3 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=51 RQADDR=paymq4 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=52 RQADDR=paymq5 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=53 RQADDR=paymq6 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=54 RQADDR=paymq7 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=55 RQADDR=paymq8 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=56 RQADDR=paymq9 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=57 RQADDR=paymq10 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=58 RQADDR=paymq11 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=59 RQADDR=paymq12 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=60 RQADDR=paymq13 REPLYQ=N
tpcc_srv_paym SRVGRP=group1 SRVID=61 RQADDR=paymq14 REPLYQ=N

tpcc_srv_stock SRVGRP=group1 SRVID=62 RQADDR=stockq1 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=63 RQADDR=stockq2 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=64 RQADDR=stockq3 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=65 RQADDR=stockq4 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=66 RQADDR=stockq5 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=67 RQADDR=stockq6 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=68 RQADDR=stockq7 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=69 RQADDR=stockq8 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=70 RQADDR=stockq9 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=71 RQADDR=stockq10 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=72 RQADDR=stockq11 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=73 RQADDR=stockq12 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=74 RQADDR=stockq13 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=75 RQADDR=stockq14 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=76 RQADDR=stockq15 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=77 RQADDR=stockq16 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=78 RQADDR=stockq17 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=79 RQADDR=stockq18 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=80 RQADDR=stockq19 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=81 RQADDR=stockq20 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=82 RQADDR=stockq21 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=83 RQADDR=stockq22 REPLYQ=N
tpcc_srv_stock SRVGRP=group1 SRVID=84 RQADDR=stockq23 REPLYQ=N

*SERVICES
NEWO
PAYM
ORDS
DEL
STOCK
```

Appendix E. Disk Storage

This Appendix contains the 180-day space calculations.

```
These calculations are used to determine the storage requirements
for 8 hours worth of logical logs as well as the 180day space
requirements.

# newords 4,063,842
# newords rolled back 40,801
# commited newords 4,023,041
#  payment 3,914,864
# delivery 368,227

user commits per tpmC 2.0439

Redo blocks written - r+m 213640339
Redo block size 512 bytes
user commits - r+m 14721410
redo blocks written per neworder submitted 29.837582086

rdeo blocks written - ramp + measure 216005595
neworder commits - ramp + measure 7239380
user commits - ramp + measure 14946563
user commits/neworder commit - r+m 2.064619208
```

TPM	135,461.40	**Warehouses**	11,000

SEGMENT	TYPE	TSPACE	BLOCKS	FIVE_PCT	DAILY_GROW	TOTAL
CUSTOMER	TABLE	CUST	165,001,012	8,250,051	0	173,251,063
DISTRICT	TABLE	WD_N?	110,000	5,500	0	115,500
HISTORY	TABLE	HIST_N?	9,398,228	469,911	1,851,778	11,719,917
ICUSTOMER	INDEX	C_IX	4,915,200	245,760	0	5,160,960
CLAST_IDX	INDEX	ICUST2	10,219,520	510,976	0	10,730,496
D_IDX	INDEX	WD_N?	2,048	102	0	2,150
I_IDX	INDEX	WD_N0	10,752	538	0	11,290
INORD	INDEX	NEWO_N?	1,431,552	71,578	0	1,503,130
O_IDX1	INDEX	ORD_IX_N?	5,730,304	286,515	0	6,016,819
O_IDX2	INDEX	ORD2_IX_N?	8,380,416	419,021	0	8,799,437
IORDL	INDEX	'OLINE_N?	135,303,168	6,765,158	26,659,428	168,727,754
ISTOCK	INDEX	S_IX	11,852,800	592,640	0	12,445,440
ITEM	TABLE	ITEM	6,667	333	0	7,000
W_IDX	INDEX	WD_N?	2,048	102	0	2,150
ORDERS2	TABLE	ORD_ALT_N?	6,871,622	343,581	1,353,948	8,569,151
ROLL_SEG	SYS	ROLL	407,552	20,378	0	427,930
STOCK	TABLE	STK	220,000,002	11,000,000	0	231,000,002
SYSTEM	SYS	SYSTEM	102,400	5,120	0	107,520
WAREHOUSE	TABLE	WD_N?	11,016	551	0	11,567
Total			579,756,307	28,987,815	29,865,154	638,609,276

Dynamic space	151,573,018	
Static space	457,171,104	
Free space	29,865,154	
Daily growth	29,865,154	
Daily spread	0	Oracle may be configured such that daily spread is 0
180-day space (blk.)	5,832,898,824	
Block size (bytes)	2,048	
180-day (GB)	11,125.37	
Log block size	512	
Log blocks/N_O txn.	29.8376	Log blocks used per New-Order transactions
8-hour log (GB)	925.10	
8-hour log mirrored(GB)	1850.21	

Storage Type	Disk Type	Disk Capacity (GB)	SUT # disks	Priced # disks	SUT capacity	Priced capacity
Logs - A5100	18GB 7krpm	16.86	112	112	1888.87	1888.87
Data - A5000	9GB 7krpm	8.43	660		5564.77	
Data - A5200	9GB 10krpm	8.43	585	1320	4932.41	11129.55
		Totals:	1357	1432	12386.06	13018.42

Appendix F. Driver Scripts

The following code sections show how the transactions are generated and how statistics are gathered. Each of the transaction functions generates the input data for that transaction, sends it to the client, reads the output form and computes keying, response and think time statistics.

This is the main loop of the RTE

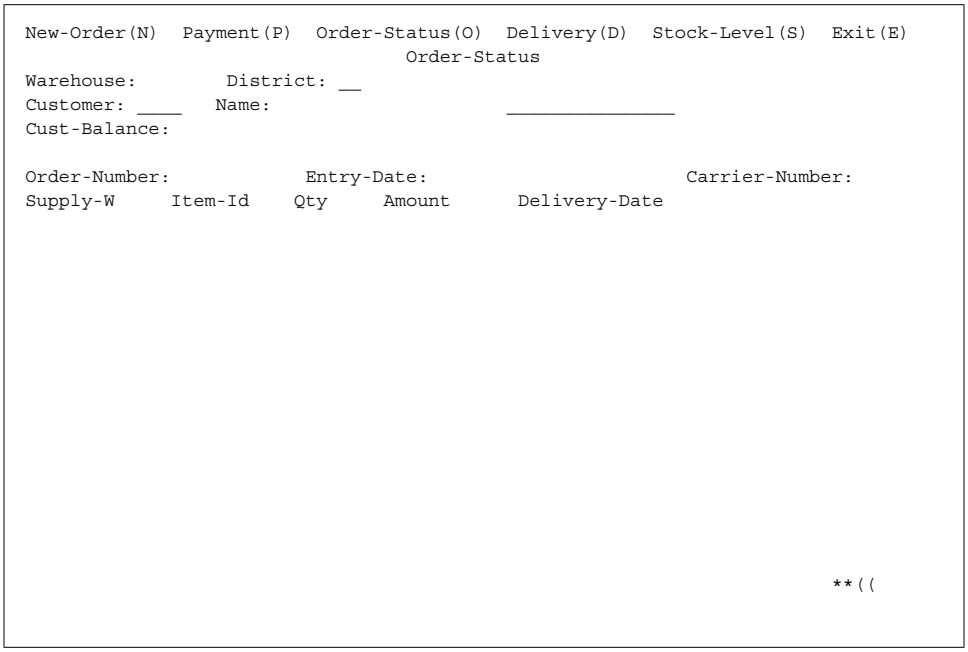
```
/* run for ramp up without capturing the stats */
i=0;
in_ramp = 1;
while (1)
{
tx_type = do_menu();/* Select transaction */
switch (tx_type) {
case NEWORDER:
do_neworder();
break;
case PAYMENT:
do_payment();
break;
case DELIVERY:
do_delivery();
break;
case ORDSTAT:
do_ordstat();
break;
case STOCKLEVEL:
do_stocklevel();
break;
default:
fprintf(stderr, "%s: Slave %d: Internal error. Tx-type = %d\n",
hostname, slave_num, tx_type);
cleanup(-1);
}
end_time = gettime();
if ( end_time >= control->end_rampup &&
end_time < control->end_stdystate )
in_ramp = 0;
else
in_ramp = 1;
if (end_time >= control->end_rampdown)
break;
}
The do_menu function selects the transaction to execute based on the weighted
distribution
algorithm.
int
do_menu()
{
int val, result, menu_start, menu_end, menu_resp;
char ch;
/* Read menu line from client */
/* Choose tx. type*/
/* Now select menu and compute menu response time */
menu_start = gettime();
/* Write menu selection to client */
/* Read input form for this transaction type */
menu_end = gettime();
menu_resp = menu_end - menu_start;
if ( ! in_ramp) {
statsp->menu_resp += menu_resp;
/* Post in histogram bucket */
if ((menu_resp / MENU_BUCKET) < MENU_MAX)
statsp->menu_hist[menu_resp / MENU_BUCKET]++;
else
statsp->menu_hist[MENU_MAX - 1]++;
if (menu_resp > statsp->menu_max)
statsp->menu_max = menu_resp;
}
return(result);
}
/*
 * Function: do_neworder
 * This function executes the neworder transaction
 * It generates all the input fields, sends it to the
 * client over the keying time, measures the response
 * time, reads the results and delays for the think time.
 */
 /* The code for the other transactions is similar */
do_neworder()
{
struct newo_fld no;
struct items_fld *itemp = no.items;
int ol_cnt, rbk, remote = 0, i, x;
char *bufp = fldbuf;
int start_time, end_time, key_time, resp_time, elapse_time, del;
start_time = gettime();
/* Now wait for keying time */
poll (0, 0, NEWO_KEY);
/* Generate all input data */
no.d_id = random(1, 10);
no.c_id = NURand(1023, 1, 3000, CONST_CID);
ol_cnt = random(5, 15);
rbk = random(1, 100);/* trans. to be rolledback */
sprintf(bufp, "%02d%04d", no.d_id, no.c_id);
bufp += strlen(bufp);
/* Generate all the item fields */
for (i=0; i < ol_cnt; i++, itemp++) {
itemp->ol_i_id = NURand(8191, 1, 100000, CONST_IID);
/* If last item and rbk, select unused item */
if (i == ol_cnt - 1 && rbk == 1) {
itemp->ol_i_id = 100001;
}
x = random(1, 100);
if ( x > 1)
itemp->ol_supply_w_id = W_ID;
else {
/* Select a warehouse other than w_id */
do {
x = random(1, control->scale);
} while (x == W_ID);
itemp->ol_supply_w_id = x;
remote++;
}
itemp->ol_quantity = random(1, 10);
sprintf(bufp, "%05d%06d%02d",itemp->ol_supply_w_id,
itemp->ol_i_id, itemp->ol_quantity);
bufp += strlen(bufp);
}
strcpy(bufp, leave_key);
bufp += 2;
/* Compute keying time info */
end_time = gettime();
key_time = end_time - start_time;
start_time = end_time;

/* Now send fields to client */
/* Read output screen from client */
end_time = gettime();
/* Store elapse time info for thruput */
elapse_time = end_time - control->start_time;
/* compute the how long it took to run the tx */
resp_time = end_time - start_time + control->newo_delta;
/* Wait think time */
del = delay(control->newo_think, 5*control->newo_think);
poll(0, 0, del + control->newo_delta);
end_time = gettime();
/* Now post all stats */
if ( ! in_ramp && end_time <= control->end_stdystate) {
statsp->newo_cnt++; /* another one bytes the dust */
if ( rbk == 1 )
statsp->newo_rbkcnt++;
statsp->newo_remote += remote;
statsp->newo_olcnt += ol_cnt;
statsp->newo_key += key_time;
/* Save keying time in histogram bucket */
statsp->newo_resp += (double) resp_time;/* sum up the response time */
/* Save response time in histogram bucket */
statsp->newo_think += (double) del;
/* Save think time in histogram bucket */
}
}
```

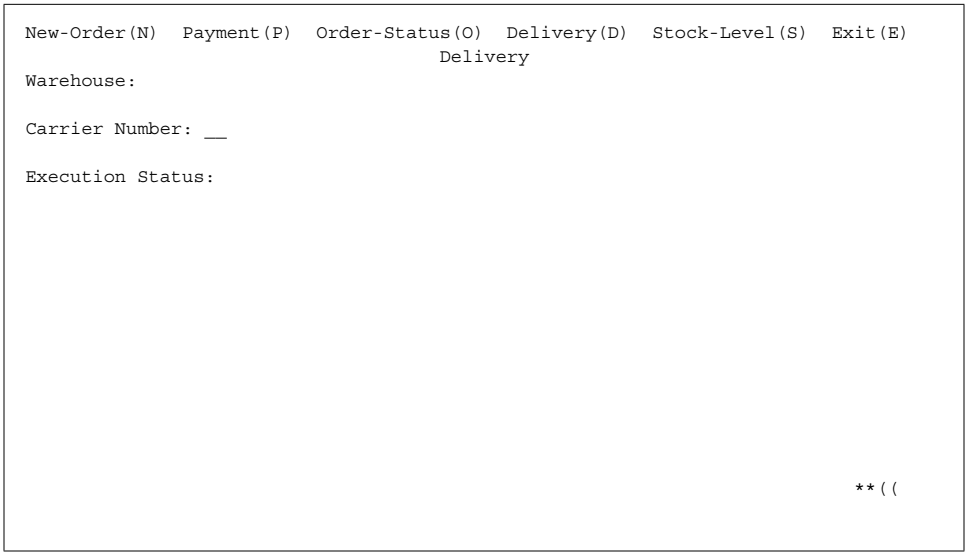
Appendix G. Screen Layout

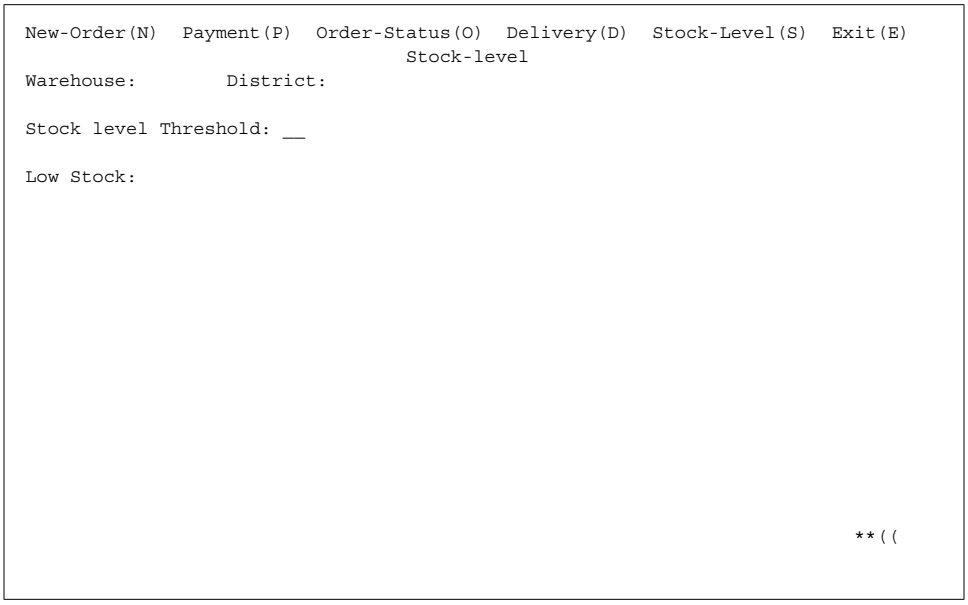
This Appendix contains the screen form layouts for the 5 transactions.

```
New-Order(N)  Payment(P)  Order-Status(O)  Delivery(D)  Stock-Level(S)  Exit(E)
                                  New Order
Warehouse:        District: __                      Date:
Customer: ____   Name:                    Credit:      %Disc:
Order Number:          Number of Lines:          W_tax:         D_tax:

 Supp_W  Item_Id  Item Name                 Qty  Stock  B/G  Price    Amount
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
  ____   ______                            __
Execution Status:                                          Total:
                                                                  **((
```

```
New-Order(N)  Payment(P)  Order-Status(O)  Delivery(D)  Stock-Level(S)  Exit(E)
                                  Payment
Date:

Warehouse:                              District: __




Customer: ____  Cust-Warehouse: ____  Cust-District: __
Name:                          _______________     Since:
                                                    Credit:
                                                    %Disc:
                                                    Phone:

Amount Paid:          _______       New Cust-Balance:
Credit Limit:

Cust-Data:




                                                                  **((
```


New-Order(N) Payment(P) Order-Status(O) Delivery(D) Stock-Level(S) Exit(E)
Order-Status
Warehouse: District: __
Customer: ____ Name: _____
Cust-Balance:
Order-Number: Entry-Date: Carrier-Number:
Supply-W Item-Id Qty Amount Delivery-Date
**((


New-Order(N) Payment(P) Order-Status(O) Delivery(D) Stock-Level(S) Exit(E)
Delivery
Warehouse:
Carrier Number: __
Execution Status:
**((


New-Order(N) Payment(P) Order-Status(O) Delivery(D) Stock-Level(S) Exit(E)
Stock-level
Warehouse: District:
Stock level Threshold: __
Low Stock:
**((

Appendix H. Price Quotes

The following pages contain the pricing quotes for the hardware and software included in this FDR.

BEA Systems, Inc.
2315 North First Street
San Jose, CA 95131
Ph: 408.570.8019
Fax: 408.570.8901

To: Ganesh Ramamurthy

Company: Sun Microsystems **Fax:** 650.786.7353

From: Christina Claure **Date:** 9/14/99

Number of Pages: 1

If you experience any difficulty receiving this fax, please contact 408.570.8000.

Dear Ganesh,

For purposes of your benchmarking activity, we recommend the use of BEA Tuxedo 6.3 available through our Core Functionality Services (CFS) program. I understand the computer that will be used for this benchmark will be a single processor workstation or server, and therefore will be subject to Tier 1 Pricing as follows:

Unlimited User License Fees per Server	***Number of users***	***Dollar Amount***	***Maintenance (5x8) per year***
Tier 1 – PC Servers with 1 or 2 CPUs, entry level RISC Uniprocessor workstation and servers	Unlimited	$3,000	$ 480

This quote is good for 90 days. Please contact me at 408.570.8019 if you have any questions.

Sincerely,

Christina Claure
Global Alliances Manager

September 14, 1999

Ganesh Ramamurthy
Sun Microsystems Inc
910 San Antonio Rd
Palo Alto, CA 94303

Quotation

Quantity	Part No.	Description	Unit Price	Total
10	NX-SOHODH8	8-port 10/100Mbps FAST Ethernet Hub	$129.00	$1,290.00
15136	NX-H9EZ	9-port 10Mbps Ethernet Hub	$30.00	$454,080
4	NX-SW8	8-port 10/100Mbps Ethernet Switch	$219.00	$876.00

Terms and Conditions:
FOB Origin
Quote Valid for 60 days
5 Year Warranty

Sincerely,
Martin Parry
NETLUX

NETLUX **1-800-789-1780**
Phone#626-851-9737
14180 Live Oak Ave., Unit E Fax #626-851-9837

20.09.99
Quote: valid for 90 days

Ganesh Ramamurthy
Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

Dick Crouch
CAT Technology, Inc.
131 C Albright
Los Gatos, CA 95032

Phone: 408-341-1717

ITEM	PART NO.	DESCRIPTION	QTY	UNIT PRICE	AMOUNT
1	E6501	Enterprise 6500 Server Base	4	$86,400.00	$345,600.00
2	2602A	CPU/Memory board	48	$4,320.00	$207,360.00
3	2580A	400MHz/8MB Ultra SPARC II	96	$10,800.00	$1,036,800.00
4	7023A	1GB memory (8*128MB)	32	$6,840.00	$218,880.00
5	954A	PS/300W for Ex000	24	$1,296.00	$31,104.00
6	2612A	Sbus I/O board	16	$4,680.00	$74,880.00
7	6730A	FCAL 100MB/S Sbus Host Adapter	48	$1,944.00	$93,312.00
8	6731a	FCAL GBIC Module	188	$432.00	$81,216.00
9	SG-ARY522A-200GR4	200 GB StorEdge A5200 Array	16	$68,040.00	$1,088,640.00
10	SG-ARY520A-200G	200 GB StorEdge A5200 Array	44	$68,040.00	$2,993,760.00
11	SG-ARY530A-254G	252 GB StorEdge A5100 Array	8	$45,288.00	$362,304.00
12	978A	FC-AL cables	72	$978.00	$70,416.00
13	SG-XDSK010A-9G	9GB disk Unipack	17	$1,053.00	$17,901.00
14	X1312A	Cluster Terminal Concentrator	1	$1,800.00	$1,800.00
15	6283A	12-24GB 4mm DDS-3 Tape Drive	4	$972.00	$3,888.00
16	A22UHC1Z9S-B512CP	Ultra 10 Server Model 333	40	$5,433.00	$217,320.00
17	7039A	512MB Memory for Ultra 10	40	$1,562.00	$62,480.00
18	1034A	PCI QEF Card	40	$1,292.00	$51,680.00
19	X7126	Color Monitor	40	$418.00	$16,720.00
		Total		**TOTAL**	**$6,622,173.00**

9/20/99 6:49:18 PM Page 1