hp server rx5670
using
HP-UX 11.i, v1.6 with Performance Pack, 64-bit
and
Oracle10i Database Standard Edition

TPC Benchmark® C
Full Disclosure Report

First Edition

Submitted for Review
November 13, 2002

First Edition - November 13, 2002

Hewlett-Packard Company believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. Hewlett-Packard Company assumes no responsibility for any errors that may appear in this document.
The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, Hewlett-Packard Company provides no warranty of the pricing information in this document.
Benchmark results are highly dependent upon workload, specific application requirements, and system design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC Benchmark® C should not be used as a substitute for a specific customer application benchmark when critical capacity planning and/or product evaluation decisions are contemplated.
All performance data contained in this report was obtained in a rigorously controlled environment. Results obtained in other operating environments may vary significantly. Hewlett-Packard Company does not warrant or represent that a user can or will achieve similar performance expressed in transactions per minute (tpmC®) or normalized price/performance ($/tpmC®). No warranty of system performance or price/performance is expressed or implied in this report.

Printed in U.S.A., November 13, 2002.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark® C test conducted on the hp server rx5670 in a client/server configuration, using Oracle10i Database Standard Edition and the TUXEDO 8.0 transaction monitor. The operating system used for the benchmark was Hewlett-Packard's HP-UX 11.i, v1.6 with Performance Pack, 64-bit. The application was written in C and compiled using HP C/ANSI C/HP-UX.

TPC Benchmark C Metrics

The standard TPC Benchmark® C metrics, tpmC® (transactions per minute), price per tpmC® (three year capital cost per measured tpmC®), and the availability date are reported as required by the benchmark specification.

Standard and Executive Summary Statements

Page *iii* contains the standard system summary and pages ***iv-vi*** contain the executive summary of the benchmark results for the hp server rx5670.

Auditor

The benchmark configuration, environment and methodology used to produce and validate the test results, and the pricing model used to calculate the price/performance, were audited by Lorna Livingtree for Performance Metrics, Inc. to verify compliance with the relevant TPC specifications.

Standard System Summary

Company Name	**System Name**	**Database Software**	**Operating System Software**
Hewlett-Packard Company	hp server rx5670	Oracle10i Database Standard Edition	HP-UX 11.i, v1.6 with Performance Pack, 64-bit
HP H/W Availability Date - Now S/W Availability Date - May 11, 2003			
Total System Cost	**TPC-C® Throughput**	**Price/Performance**	
Hardware Software 3-year maintenance	Sustained maximum throughput of System running TPC-C® expressed in transactions per minute	Total system cost/tpmC ($1,068,647/80570.81)	
$1,068,647	**80,570.81 tpmC**	**$13.26 per tpmC**	

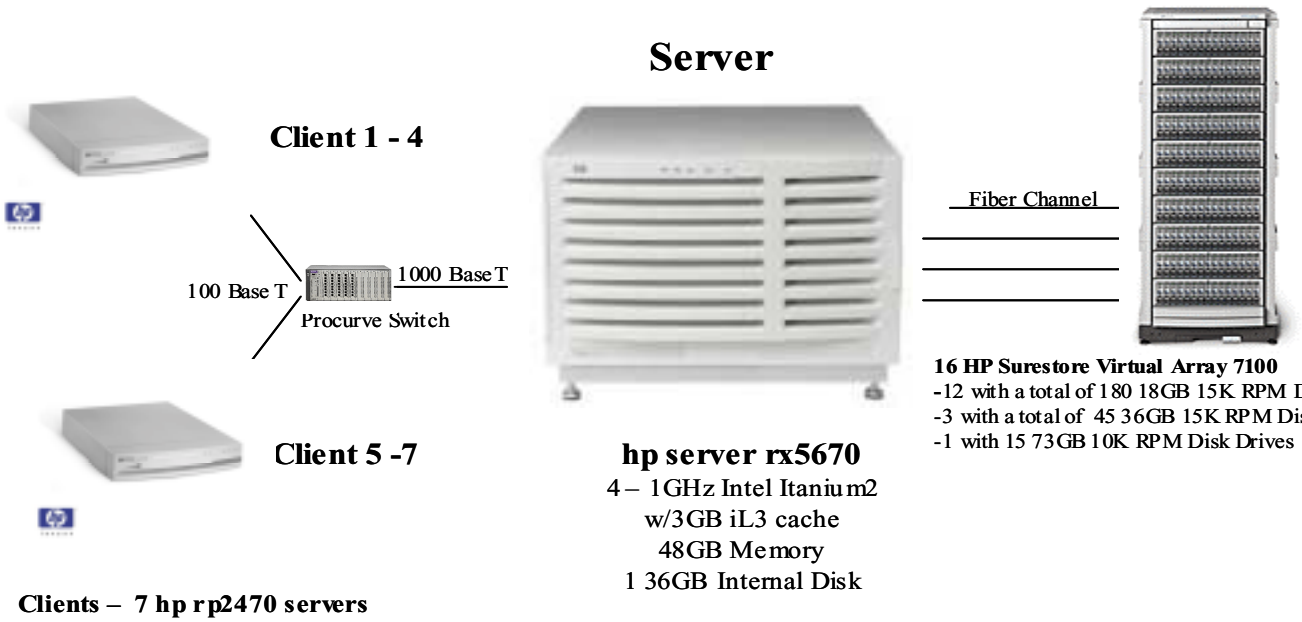
hp server rx5670

TPC-C Revision 5

Report Date: November 13, 2002

Total System Cost	TPC Throughput	Price/Performance	Availability Date
$1,068,647	**80,570.81 tpmC**	**$13.26/tpmC**	**May 11, 2003** Hardware available now

Processors	Database Manager	Operating System	Other Software	Number of Users
4 Intel Itanium2 1GHz	**Oracle10i Database Standard Edition**	**HP-UX 11.i, v1.6 with Performance Pack, 64-bit**	**TUXEDO 8.0**	**64,800**



System Components	*Server (hp server rx5670)*		*each Client (7 hp server rp2470)*	
	Qty	*Type*	*Qty*	*Type*
Processors	4	1GHz Intel Itanium2	1	750MHz PA-RISC 8700
Cache Memory	each	3MB L3 cache	each	.75MB I-cache/1.5 MB D-cache
Memory	48	GB	1	3 GB
Disk Controllers	4	2GB Fibre Channel Adapter	1	Ultra2 SCSI LVD
Disk Drives	16	Surestore Virtual Array 7100, 12 with a total of 180 15K RPM Disks, 3 with a total of 45 36GB 15K RPM Disks and 1 with 15 73GB 10K RPM Disks	1	18 GB
Total Storage	2732.56	GB		
Tape Drives	1	DVD ROM		
Terminals	1	Console Terminal	1	Console Terminal

hp server rx5670

TPC-C Rev 5

Report Date: November 13, 2002

Description	Part Number	Brand	Price Key	US List Price	Qty	Price	3Year Main.Price
Server Hardware							
hp server rx5670 with 1GHZ processor	A6838A		1	26,494	1	26,494	7,052
1GHz Processor with 3MB Cache	A6836A		1	8,250	3	24,750	3,459
4GB PC2100 DDR-SDRAM Memory quad	A6834A		1	8,000	12	96,000	
Memory Carrier Board	A6747A		1	1,981	2	3,962	
36GB 10K Internal Drive	A7048A		1	587	1	587	
DVD ROM	A5557B		1	450	1	450	
Sytstem Console	C1099A		1	550	1	550	
HP-UX Operating Environment	B0916A, Opt AFF		1	520	1	520	
HP-UX 11i, v1.6 Performance Pack Per-Processor Licens	B9429AC		1	2,495	4	9,980	
2GB Fibre Channel Adapter	A6795A		1	2,240	4	8,960	
PowerTrust 3.0VA UPS 230VUPS	A1353A		1	1,735	8	13,880	
Surestore VA 7100 w/dual controllers 512MB cache	A6262A		1	44,250	16	708,000	58,784
18GB 15K RPM FC HDD	A6191A, Opt. 0D1		1	914	180	164,520	
36GB 15K RPM FC HDD.	A6193A, Opt 0D1		1	1,349	45	60,705	
73GB 10K RPM FC HDD.	A6194A, Opt 0D1		1	2,019	15	30,285	
2 meter LC/SC Fibre Optic Cable	C7529A		1	215	20	4,300	
HP FC 1GB/2GB Entry Switch 8B, Field Rack	A7346A		1	6,599	4	26,396	
HP9000 Std. Rack System E41	A4902A		1	1,910	2	3,820	
Modular Power Dist.	A5137AZ		1	145	8	1,160	
200-240 Volts Power Option	A5137AZ, Opt AW4		1	94	8	752	
				Subtotal		**1,186,071**	**69,295**
Server Software							
Oracle10i Database Standard Edition							
Processor License for 3 years		Oracle	2	7,500	4	30,000	
Oracle Database Server Support Package for 3 years			2	6,000	1		6,000
				Subtotal		**30,000**	**6,000**
Client Hardware							
HP server rp2470	A6890A		1	1,865	7	13,055	24,815
750Mhz PA-RISC 8700 CPU	A6892A		1	5,100	7	35,700	
18GB 10K HotPlug Ultra 160 SCSI Internal Disk	A6741A		1	732	7	5,124	
2GB Memory Module	A6114A		1	6,000	7	42,000	
1GB Memory Module	A5841A		1	1,999	7	13,993	
HP-UX 11.i Sys Media, CD-ROM	B3920EA, Opt. AAF		1	520	7	3,640	
				Subtotal		**113,512**	**24,815**
Client Software							
HP C/ANSI C Compiler	B3901BA, Option AH0		1	1600	1	1,600	170
BEA Tuxedo 8.0		Bea Sys.	3	2,850	7	19,950	13,230
				Subtotal		**21,550**	**13,400**
User Connectivity							
HP ProCurve Switch 4000M	J4121A		1	2379	1	2,379	588
HP ProCurve Switch 100/1000Base-T Mod.	J4115B		1	509	1	509	
				Subtotal		**2,888**	**588**
Oracle Mandatory E-Business Discount (license and support)			2	(1,800)	1	(1,800)	
HP's Large configuration Discount and Support Prepayment*						($387,023)	($10,649)
*All discounts are based on US list prices and for similar quantities and configurations				**Total**		**965,198**	**103,449**

1=HP 2= Oracle (Pricing Contact: Herve Lejeune(see Appendix F) 3=BEA Systems

Audited by Lorna Livingtree for Performance Metrics, Inc.

Three Year Cost of Ownership:	**$1,068,647**
tpmC Rating:	**80,571**
$/tpmC:	**$13.26**

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

Numerical Quantities Summary for hp server rx5670

MQTH, Computed Maximum Qualified Throughput **80,570.81 tpmC**

Response Times (in seconds)

	90th %-ile	Maximum	Average
New-Order	1.11s	33.45s	0.61s
Payment	1.10s	33.52s	0.59s
Order-Status	1.12s	32.64s	0.62s
Delivery (interactive portion)	0.61s	32.27s	0.24s
Delivery (deferred portion)	1.11s	32.93s	0.61s
Stock-Level	1.13s	32.67s	0.61s
Menu	0.0001s	0.00125s	0.000004s

Transaction Mix, in percent of total transactions

New-Order	44.96%
Payment	43.01%
Order-Status	4.01%
Delivery	4.002%
Stock-Level	4.01%

Keying/Think Times

	Keying Time			Think Time		
	Min	Avg	Max	Min	Avg	Max
New-Order	18.02s	18.03s	18.18s	0.01s	12.13s	173.27s
Payment	3.01s	3.02s	3.18s	0.01s	12.07s	187.1s
Order-Status	2.01s	2.02s	2.18s	0.01s	10.11s	144.01s
Delivery (interactive)	2.01s	2.02s	2.18s	0.01s	5.05s	67.78s
Stock-Level	2.01s	2.02s	2.18s	0.01s	5.06s	68.44s

Test Duration

Ramp up time	37 minutes
Measurement interval	120 minutes
Transactions during measurement interval	21,503,951
Ramp down time	33.358 minutes

Checkpointing

Number of checkpoints in measurement interval	4
Checkpoint Interval	29.51 minutes

TPC Benchmark C Overview

This is the full disclosure report for a benchmark test of the hp server rx5670 using Oracle10i Database Standard Edition . It meets the requirements of the TPC Benchmark® C Standard Specification, Revision 5 dated March, 2001.

TPC Benchmark® C was developed by the **T**ransaction Processing **P**erformance **C**ouncil (TPC). It is the intent of this group to develop a suite of benchmarks to measure the performance of computer systems executing a wide range of applications. Hewlett-Packard Company Oracle Corporation are active participants in the TPC.

*TPC Benchmark® C is an **O**n **L**ine **T**ransaction **P**rocessing (OLTP) workload. It is a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP application environments. It does so by exercising a breadth of system components associated with such environments, which are characterized by:*

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

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

Despite the fact that this benchmark offers a rich environment that emulates many OLTP applications, this benchmark does not reflect the entire range of OLTP requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C® approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to other environments are not recommended.

Hewlett-Packard Company does not warrant or represent that a user can or will achieve performance similar to the benchmark results contained in this report. No warranty of system performance or price/performance is expressed or implied by 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 output functions.

Appendix A contains the HP C/ANSI C/HP-UX application code used in this TPC-C® test.

1.2 Test Sponsor

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

The Business Critical Systems Division of Hewlett-Packard Company is the test sponsor of this TPC Benchmark® C.

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 options
- Recover/commit options
- Consistency/locking options
- Operating system and application configuration parameter
- 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.

The intent of the above clause is that anyone attempting to recreate the benchmark environment has sufficient information to compile, link, optimize, and execute all software used to produce the disclosed benchmark result.

Appendix A contains the application "make" files. Appendix C contains the HP-UX operating system parameters used to generate the kernel for the configuration used in this benchmark. Also included are all of the Oracle10i Database Standard Edition database parameters and the TUXEDO 8.0 transaction monitor parameters used.

1.4 Configuration Diagrams

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

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

The server System Under Test, an hp server rx5670 depicted in Figure 1.1, consisted of:

- 4 1GHz Intel Itanium2 System Processors
- 48 GB of memory

- 4 2GB Fibre Channel Adapter Adapters
- 16 Surestore Virtual Array 7100 (12 with a total of 180 18GB 15K RPM disks, 3 with a total of 45 36GB 15K RPM, and 1 with 15 73GB 10K RPM disk drives
- Two LAN interfaces

As indicated in Figure 1.2, this benchmark configuration used Remote Terminal Emulator (RTE) programs that executed on 7 rp2470 Enterprise Server drivers to emulate TPC-C user sessions. The emulated users on the driver systems were directly connected to the client systems under test via 28 separate 100 Base-T local area network (LAN) and communicated using TCP/IP. The clients were connected to the SUT via a HP Procurve 4000M switch.

The priced configuration for the hp server rx5670 is shown in Figure 1.1. In the priced configuration, the RTE shown in the benchmark configuration is replaced by the appropriate number of workstations (emulating ANSI terminals) connected to hubs.

Figure 1.1: hp server rx5670 Priced Configuration

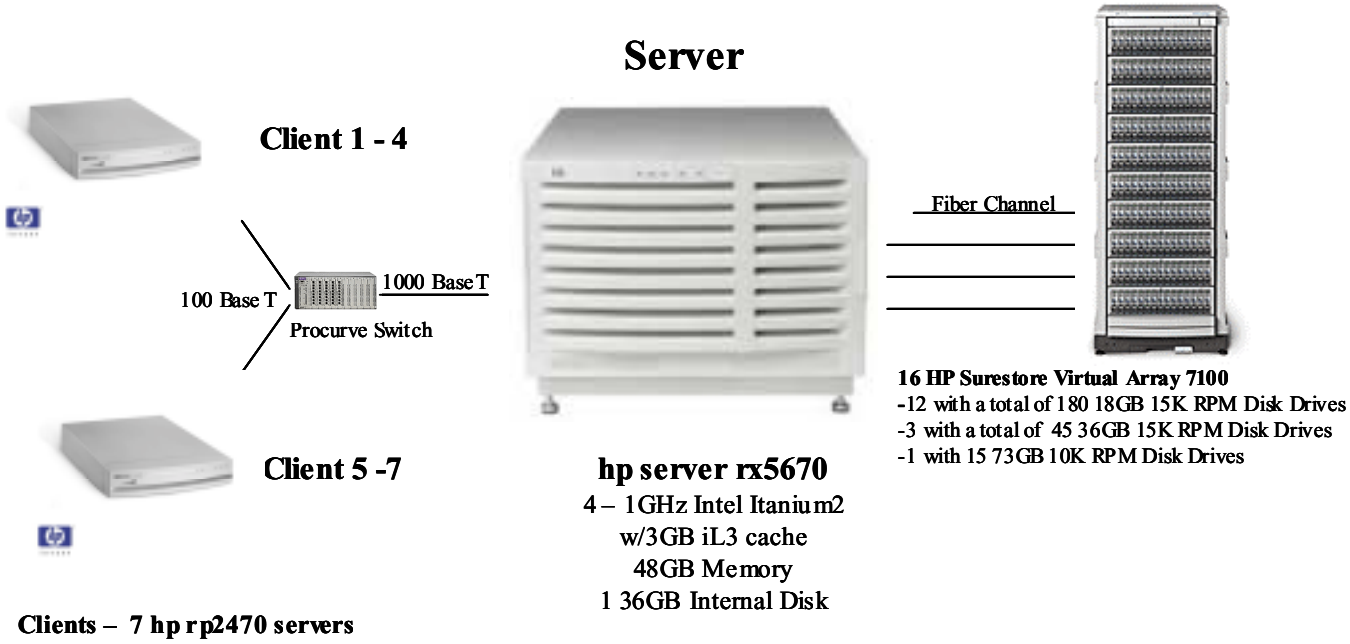
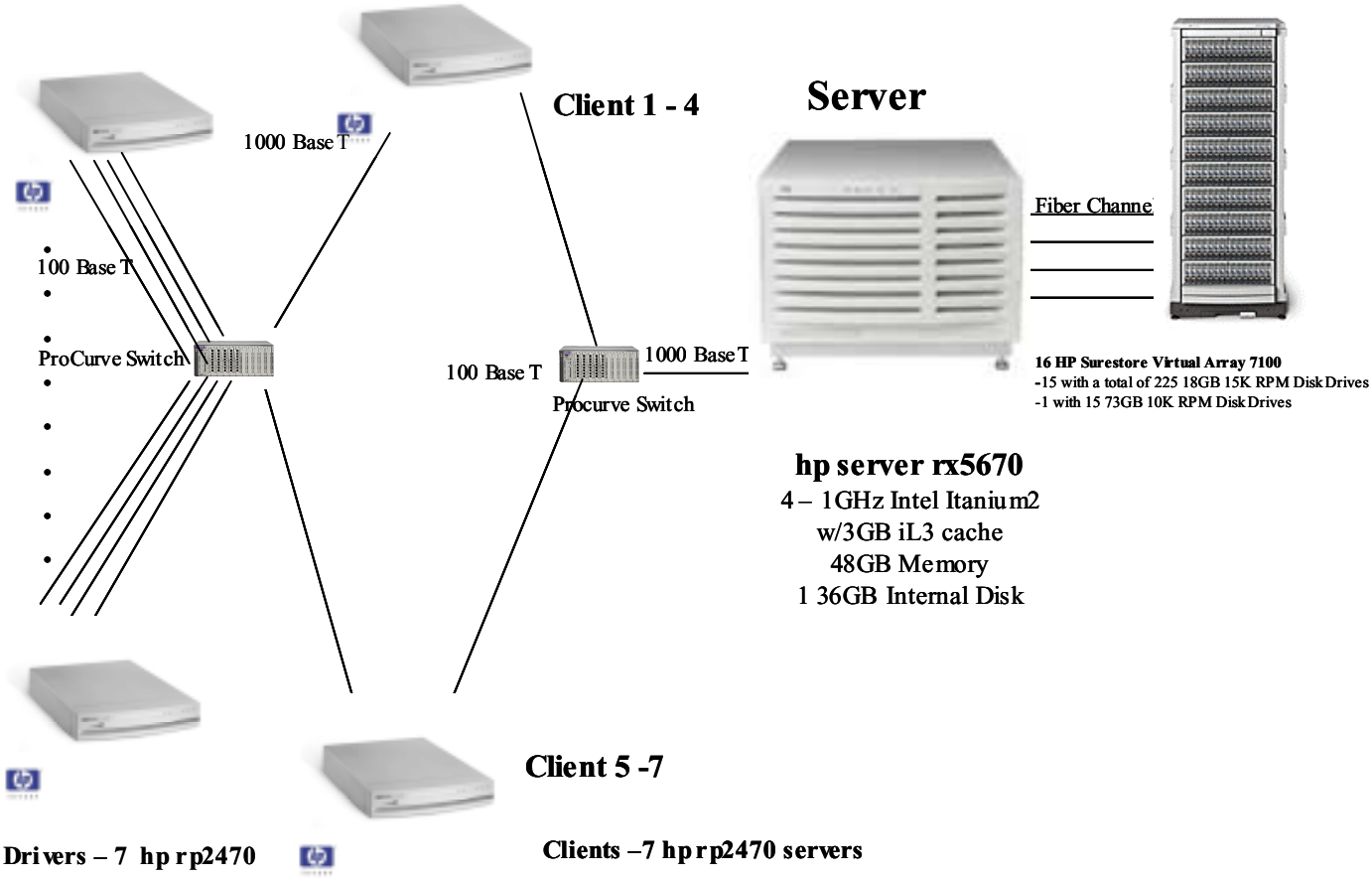


Figure 1.2: hp server rx5670 Benchmark Configuration



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 the Oracle10i Database Standard Edition 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 Oracle10i Database Standard Edition according to the data in section 5.2. The size of the database table space on each disk drive was calculated to provide even distribution of load across the disk drives.

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.

2.4 Partitioning

While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C® benchmark, any such partitioning must be disclosed. Replication of tables, if used, must be disclosed. Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.

Partitioning, replication, and additional or duplicated attributes were not used in this implementation.

3 Clause 2 Related Items

3.1 Random Number Generation

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

The random number generator used can be found in the source appendix. It is from the book "The Art of Computer Systems Performance Analysis" by Raj Jain, page 443. The properties of this random number generator are documented in the book. It is a full-period multiplicative linear-congruential random number generator.

3.2 Input/Output Screen Layout

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

The screen layouts corresponded exactly to those in Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3, and 2.8.3 of the TPC-C® Standard Specification.

3.3 Priced 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. Although not specifically priced, the type and model of the terminals used for the demonstration in 8.1.3.3 must be disclosed and commercially available (including supporting software and maintenance).

The terminal features were verified by manually exercising each specification on an HP 712/80 workstation running an ANSI terminal emulator.

3.4 Presentation Manager or Intelligent Terminal

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

Application code running on the client implemented the TPC-C user interface. A listing of this code is included in Appendix A. Used capabilities of the terminal beyond basic ASCII entry and display were restricted to cursor positioning.

A presentation manager was not used.

Table 3.1: Transaction Statistics

Type	Item	Value
New Order	Home warehouse items	9.53%
	Remote warehouse items	90.47%
	Rolled back transactions	1.00%
	Average items per order	10.00
Payment	Home warehouse	85.03%
	Remote warehouse	14.97%
	Non primary key access	60.01%
Order Status	Non primary key access	60.03%
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.96%
	Payment	43.01%
	Order Status	4.01%
	Delivery	4.002%
	Stock Level	4.01%

3.5 Transaction Statistics

Table 3.1 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 require.

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 mechanism that other transactions used. The only difference was that the call was asynchronous, i.e., control would return to the client process immediately and the deferred delivery part would complete 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 quotes the specification definition of each of these properties and describes the tests done as specified and monitored by the auditor to demonstrate compliance.

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, WAREHOUSE, and DISTRICT tables have been changed appropriately.

The values of w_ytd, d_ytd, c_balance, c_ytd_payment, and c_payment_cnt of a randomly selected warehouse, district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was committed. The values w_ytd, d_ytd, c_balance, c_ytd_payment, and c_payment_cnt were retrieved again. It was verified that all values had been changed appropriately.

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, WAREHOUSE, and DISTRICT tables have NOT been changed

The values of w_ytd, d_ytd, c_balance, c_ytd_payment and c_payment_cnt of a randomly selected warehouse, district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was rolled back. The values of w_ytd, d_ytd, c_balance, c_ytd_payment, c_payment_cnt were retrieved again. It was verified that none of the values had changed.

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 the database is initially in a consistent state.

The TPC Benchmark C standard requires the System Under Test to meet the following 12 consistency conditions (c.f. *TPC Standard Specification, Clauses 3.3.2.1 to 3.3.2.12*):

1. the sum of the district balances in a warehouse is equal to the warehouse balance;
2. for each district, the next order-id minus one is equal to maximum order-id in the ORDER table and equal to the maximum new-order-id in the NEW-ORDER table;

3. for each district, the maximum order-id minus minimum order-id in the ORDER table plus one equals the number of rows in the NEW-ORDER table for that district;
4. for each district, the sum of the order-line counts equals the number of rows in the ORDER-LINE table for that district;
5. for each row in the ORDER table, the carrier-id is set to a null value only if there is a corresponding row in the NEW-ORDER table;
6. for each row in the ORDER table, the order-line count must equal the number of rows in the ORDER-LINE table for that order;
7. for any row in the ORDER-LINE table, the delivery date/time is set to a null value only if the corresponding row in the ORDER table has the carrier-id set to a null value;
8. for each warehouse, the year-to-date amount must equal the sum of the amounts in the HISTORY table for that warehouse;
9. for each district, the year-to-date amount must equal the sum of the amounts in the HISTORY table for that district;
10. for each customer, the balance must equal the sum of the order-line amount minus the sum of the history amount for that customer;
11. for each district, the total orders minus the total new-orders must equal the sum of the customer delivery count;
12. for any randomly selected customer, the balance plus the year-to-date payment must equal the sum of the order-line amount.

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

To demonstrate that consistency is maintained, conditions 1-4 were verified for a sample of warehouses before and after the durability tests.

4.4 Isolation

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

*This property is commonly called **serializability**. Sufficient conditions must be enabled at either the system or application level to ensure serializability of transactions under any arbitrary mix of TPC-C transactions, unless otherwise specified by the transaction profile. The system or application must have full serializability enabled (i.e., repeated reads of the same rows within any committed transaction must return identical data when run concurrently with any arbitrary mix of TPC-C transactions), except in the case of Stock-Level transaction. For the Stock-Level transaction, the isolation requirement is relaxed to simply require that the transaction see only committed data.*

The TPC Benchmark C Standard (Revision 3.5) defines nine required tests to be performed to demonstrate that the required levels of transaction isolation are met.

For conventional locking schemes, isolation should be tested as described below. Systems that implement other isolation schemes may require different validation techniques. It is the responsibility of the test sponsor to disclose those techniques and the tests for them. If isolation schemes other than conventional locking are used, it is permissible to implement these tests differently provided full details are disclosed. (Examples of different validation techniques are shown in Isolation Test 7, Clause 3.4.2.7).

4.4.1 Isolation Test 1

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

The execution of the above test proceeded as follows:

1. An Order-Status transaction T0 was executed for a randomly selected customer, and the order returned was noted. T0 was committed
2. A New-Order transaction T1 was started for the same customer used in T0. T1 was stopped prior to COMMIT.
3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was allowed to complete and was committed.
5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the order inserted by T1.

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

4.4.2 Isolation Test 2

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

The execution of the above test proceeded as follows:

1. An Order-Status transaction T0 was executed for a randomly selected customer and the order returned was noted. T0 was committed.
2. A New-Order transaction T1 with an invalid item number, was started for the same customer used in T0. T1 was stopped immediately prior to ROLLBACK.
3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was allowed to ROLLBACK.
5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the same order that T0 had returned.

4.4.3 Isolation Test 3

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

The execution of the above test proceeded as follows:

1. The D_NEXT_O_ID of a randomly selected district was retrieved.
2. A New-Order transaction T1 was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to COMMIT.
3. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.
4. T1 was allowed to complete. T2 completed and was committed.

5. The order number returned by T1 was the same as the D_NEXT_O_ID retrieved in step 1. The order number returned by T2 was one greater than the order number returned by T1.
6. The D_NEXT_O_ID of the same district was retrieved again. It had been incremented by two (i.e. it was one greater than the order number returned by T2).

4.4.4 Isolation Test 4

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

The execution of the above test proceeded as follows:

1. The D_NEXT_O_ID of a randomly selected district was retrieved.
2. A New-Order transaction T1, with an invalid item number, was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to ROLLBACK.
3. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.
4. T1 was allowed to roll back, and T2 completed and was committed.
5. The order number returned by T2 was the same as the D_NEXT_O_ID retrieved in step 1.
6. The D_NEXT_O_ID of the same district was retrieved again. It had been incremented by one (i.e. one greater than the order number returned by T2).

4.4.5 Isolation Test 5

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

The execution of the above test proceeded as follows:

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

4.4.6 Isolation Test 6

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

The execution of the above test proceeded as follows:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C_BALANCE of the customer found in step 1 was retrieved.

3. A Delivery business transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the ROLLBACK of the database transaction corresponding to the district used in step 1.
4. A Payment transaction T2 was started for the same customer found in step 1. T2 waited.
5. T1 was allowed to ROLLBACK. T2 completed and was committed.

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

4.4.7 Isolation Test 7

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

The execution of the above test proceeded as follows:

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

- Execution followed *Case D* of *Clause 3.4.2.7*.

4.4.8 Isolation Test 8

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

The execution of the above test proceeded as follows:

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

4.4.9 Isolation Test 9

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

The execution of the above test proceeded as follows:

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. 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. A New-Order transaction T2 was started for the same warehouse and district. T2 completed and was committed without being blocked by T1.
5. T1 was resumed and the new_ORDER table was read again. No qualifying row was found.
6. T1 completed and was committed.
7. The NO_D_ID of all new_ORDER rows for the selected warehouse and district was restored to the original value. The changes were committed.

4.5 Durability

The tested system must guarantee durability: the ability to preserve the effects of committed transaction 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)…*

Specified durability tests were executed to demonstrate satisfaction of the durability requirements for this implementation of TPC Benchmark C. One durability test, described below, covering the following failure situations was performed under the auditor's supervision:

- *Permanent irrecoverable failure of any single durable medium containing TPC-C database tables or recovery log data (Clause 3.5.3.1).*

A test was performed under a load of 64,800 users on the full-scale database for the loss of recovery log and loss of data tests. Another durability test, described below, combining the following failure situations was performed under the auditor's supervision:

- *instantaneous interruption which requires system reboot [of processors] to recover. (Clause 3.5.3.2*)
- *failure of all or part of memory. (Clause 3.5.3.3).*

This test was performed under the full performance-measurement load of 64,800 users on the full-scale database built for 64,800 users.

4.5.1 Loss of Log and Data Disks

Because the log and data devices are Redundant Disk Arrays which each function independently of the rest of the system in ensuring data integrity under loss and/or replacement of any individual disk drive (and other failures as well), integrity under such failure and replacement does not entail any interruption in processing. The test below validates the durability by demonstrating persistence of the results of transactions processed both before and during these failures, validating the durability upon database recovery (in this instance, forced) of transactions which completed before the failure and the non-effect of transactions which did not complete.

1. The D_NEXT_O_ID fields for all rows in the DISTRICT table were summed up to determine the initial count of the total number of orders (count1).
2. A test was initiated with 64,800 terminals. On the driver system, completed/rolled-back transactions (including New-Orders) were recorded in a "success" file.
3. After running at steady state throughput levels for 5 minutes, a individual disk containing recovery log was unplugged from the array.
4. Because of the built-in redundancy in the disk array, the test continued normally.
5. On the system log files, messages appeared indicating that a disk were missing.
6. After running again at steady state throughput levels for 5 minutes, a individual disk containing data was unplugged from the array.
7. Because of the built-in redundancy in the disk array, the test continued normally.
8. On the system data files, messages appeared indicating that a disk were missing.
9. The test was finished on the driver.
10. The contents of the "success" file on the driver and the ORDER table were spot-compared to verify that records in the "success" file for completed New-Order transactions had corresponding records in the ORDER table.
11. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 matched exactly the number of records for successful New Orders in the RTE "success" file.
12. Consistency test 3 was run on the database and the results were verified.
13. New disks were installed. The disk arrays automatically copied the mirrored-pair mate of the missing disks onto the new disks. Messages appeared in the system log files indicating redundancy was restored.

4.5.2 Instantaneous Interruption and Loss of Memory

Instantaneous interruption and loss of memory tests were combined because the loss of power erases the contents of memory. This failure was induced while the benchmark was running by turning off the power supplies to the server.

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. Transactions were started at full load. On the driver system, completed/rolled-back transactions (including New-Orders) were recorded in a "success" file.
3. After thirteen minutes, the benchmark throughput reached the steady state level and the server systems were de-powered.
4. The test was aborted on the driver.

5. The server system was restarted.
6. The database was restarted and a recovery performed using the transaction log.
7. The contents of the "success" file on the driver and the ORDERs table were spot-compared to verify that records in the "success" file for completed New-Order transactions had corresponding records in the ORDERs table.
8. Step 1 was repeated to determine the current total number of orders (count2). Count2-count1 (=219,104,310-218,335,68=768,632) was 5 more than the number of records for successful New Orders in the RTE "success" file (=776,430-7,803=768,627 rolled-back). *This difference would be due only to transactions which were committed on the system under test but for which the output data was not displayed on the [emulated] input/output screen before the failure.*
9. Consistency test 3 was run on the database and the results were verified.

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, must be disclosed. If the database was overscaled and inactive rows of the WAREHOUSE table were deleted the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed.

The TPC-C database for this test was configured with 6700 warehouses.

Table	Occurrences
Warehouse	6,700
District	67,000
Customer	201,000,000
History	201,000,000
Orders	201,000,000
New Orders	60,300,000
Order Line	2,010,093,060
Item	100,000
Stock	670,000,000

During the measurement only 6,480 warehouses and their associated data were accessed. This was confirmed using D_NEXT_O_1D and W_YTD as described in *Clause 4.2.2 Comment (2).*

5.2 Database and Growth Layout

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

Table 5.2 indicates the distribution of the database tables over the disks of the tested and priced systems**.**

I) root, swap, file systems:

Use	Device	Size (GB)	Device Model
root+swap	/dev/dsk/c0t0d0s2	18	HP MAN3184MC
file system	/dev/dsk/c3t0d0	18	SEAGATE ST318404LC
swap	/dev/dsk/c2t0d0	8	SEAGATE ST39102LC
swap	/dev/dsk/c2t1d0	8	SEAGATE ST39102LC
swap	/dev/dsk/c2t2d0	8	SEAGATE ST39102LC
swap	/dev/dsk/c2t3d0	8	SEAGATE ST39102LC
swap	/dev/dsk/c2t4d0	8	SEAGATE ST39102LC
swap	/dev/dsk/c2t8d0	8	SEAGATE ST39102LC

II) Database files:

We use 16 arrays. Every array is attached to the system via a Fibre Channel link.

Each array contains 15 18-GB or 73-GB disks drives, allocated to 2 to 6 RAID1 LUNs. After formatting and mirroring, the available capacity of the arrays with 18GB drives is 121.73GB. The capacity of the arrays with 73-GB drives is 497.22GB.

One array contains the redo log, and the log only. It has enough space for 8 hours of redo logs.

On the 15 data arrays, LUN 0 is 76,680MB and is allocated to volume group vgtpcc_raw. Some of the Oracle files are striped across the 15 arrays in this volume groups. The other 71 LUNS are mapped 1-to-1 to files that contain the customer table, customer index icust2, the new order table, the stock table, and files misc_0_1 and misc_0_3 of the misc tablespace.

The 16 arrays and their luns are accessed via the following paths:

/dev/dsk/c18t0[d1|d2]
/dev/dsk/c23t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c24t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c25t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c26t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c28t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c29t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c30t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c31t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c32t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c46t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c4t0[d0|d2|d3]
/dev/dsk/c53t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c54t0[d0|d1|d2|d3|d4|d5]
/dev/dsk/c5t0[d0|d1|d2|d3|d4]
/dev/dsk/c6t0[d0|d1|d2|d3|d4|d5]

4) List of all Oracle datafiles and the corresponding device: (sorted by name)

Datafile	File	Size(M)	Tblspce	Actual disk/lv path

```
                #
--------    ----   -- -- -----      -------       --------------
log_5                    13000                    /dev/rdsk/c18t0d1
log_6                    13000                    /dev/rdsk/c18t0d2
control_001                1.55                   File system file
aux.df                2      120 SYSAUX           /dev/vgtpcc_raw/r10i_lvm_aux.df
cust_0_0              8     7940 CUST_0           /dev/rdsk/c6t0d4
cust_0_1             12     7940 CUST_0           /dev/rdsk/c31t0d5
cust_0_10            41     7940 CUST_0           /dev/rdsk/c24t0d4
cust_0_11            43     7940 CUST_0           /dev/rdsk/c53t0d3
cust_0_12            45     7940 CUST_0           /dev/rdsk/c53t0d4
cust_0_13            47     7940 CUST_0           /dev/rdsk/c53t0d5
cust_0_14            48     7940 CUST_0           /dev/rdsk/c46t0d5
cust_0_15            50     7940 CUST_0           /dev/rdsk/c32t0d4
cust_0_16            52     7940 CUST_0           /dev/rdsk/c29t0d5
cust_0_17            54     7940 CUST_0           /dev/rdsk/c28t0d4
cust_0_18            56     7940 CUST_0           /dev/rdsk/c29t0d4
cust_0_19            57     7940 CUST_0           /dev/rdsk/c32t0d5
cust_0_2             13     7940 CUST_0           /dev/rdsk/c54t0d5
cust_0_20            59     7940 CUST_0           /dev/rdsk/c26t0d5
cust_0_21            61     7940 CUST_0           /dev/rdsk/c5t0d4
cust_0_22            64     7940 CUST_0           /dev/rdsk/c25t0d5
cust_0_23            93     7940 CUST_0           /dev/rdsk/c24t0d5
cust_0_24            94     7940 CUST_0           /dev/rdsk/c28t0d5
cust_0_25            95     7940 CUST_0           /dev/rdsk/c23t0d5
cust_0_3             15     7940 CUST_0           /dev/rdsk/c30t0d4
cust_0_4             17     7940 CUST_0           /dev/rdsk/c54t0d3
cust_0_5             22     7940 CUST_0           /dev/rdsk/c25t0d4
cust_0_6             24     7940 CUST_0           /dev/rdsk/c26t0d4
cust_0_7             25     7940 CUST_0           /dev/rdsk/c4t0d3
cust_0_8             27     7940 CUST_0           /dev/rdsk/c31t0d4
cust_0_9             29     7940 CUST_0           /dev/rdsk/c46t0d4
dist_0_0             31      137 DIST_0           /dev/vgtpcc_raw/r10i_lvm_dist_0_0
dist_0_1            101      137 DIST_0           /dev/vgtpcc_raw/r10i_lvm_dist_0_1
hist_0_0             32     6833 HIST_0           /dev/vgtpcc_raw/r10i_lvm_hist_0_0
hist_0_1             33     6833 HIST_0           /dev/vgtpcc_raw/r10i_lvm_hist_0_1
hist_0_2             34     6833 HIST_0           /dev/vgtpcc_raw/r10i_lvm_hist_0_2
hist_0_3             35     6833 HIST_0           /dev/vgtpcc_raw/r10i_lvm_hist_0_3
hist_0_4             36     6833 HIST_0           /dev/vgtpcc_raw/r10i_lvm_hist_0_4
icust1_0_0           84     5056 ICUST1_0         /dev/vgtpcc_raw/r10i_lvm_icust1_0_0
icust2_0_0           85    11005 ICUST2_0         /dev/rdsk/c23t0d4
idist_0_0            86       27 IDIST_0          /dev/vgtpcc_raw/r10i_lvm_idist_0_0
idist_0_1           102       27 IDIST_0          /dev/vgtpcc_raw/r10i_lvm_idist_0_1
iitem_0_0            88        2 IITEM_0          /dev/vgtpcc_raw/r10i_lvm_iitem_0_0
iordr2_0_0           89    16371 IORDR2_0         /dev/vgtpcc_raw/r10i_lvm_iordr2_0_0
istok_0_0            87    14358 ISTOK_0          /dev/vgtpcc_raw/r10i_lvm_istok_0_0
item_0_0             72       15 ITEM_0           /dev/vgtpcc_raw/r10i_lvm_item_0_0
iware_0_0            83        6 IWARE_0          /dev/vgtpcc_raw/r10i_lvm_iware_0_0
```

misc_0_0		96	2040	ICUST1_0	/dev/vgtpcc_raw/r10i_lvm_misc_0_0
misc_0_1		97	2040	ICUST2_0	/dev/rdsk/c6t0d5
misc_0_2		98	2040	ISTOK_0	/dev/vgtpcc_raw/r10i_lvm_misc_0_2
misc_0_3		99	2040	IORDR2_0	/dev/rdsk/c30t0d2
nord_0_0		82	6795	NORD_0	/dev/rdsk/c30t0d5
ordr_0_0		73	62079	ORDR_0	/dev/vgtpcc_raw/r10i_lvm_ordr_0_0
ordr_0_1		74	62079	ORDR_0	/dev/vgtpcc_raw/r10i_lvm_ordr_0_1
ordr_0_2		75	62079	ORDR_0	/dev/vgtpcc_raw/r10i_lvm_ordr_0_2
ordr_0_3		76	62079	ORDR_0	/dev/vgtpcc_raw/r10i_lvm_ordr_0_3
ordr_0_4		77	62079	ORDR_0	/dev/vgtpcc_raw/r10i_lvm_ordr_0_4
ordr_0_5		78	62079	ORDR_0	/dev/vgtpcc_raw/r10i_lvm_ordr_0_5
ordr_0_6		79	62079	ORDR_0	/dev/vgtpcc_raw/r10i_lvm_ordr_0_6
ordr_0_7		80	62079	ORDR_0	/dev/vgtpcc_raw/r10i_lvm_ordr_0_7
ordr_0_8		81	62079	ORDR_0	/dev/vgtpcc_raw/r10i_lvm_ordr_0_8
roll01		3	13400	UNDO_TS	/dev/vgtpcc_raw/r10i_lvm_roll01
sp_0	1	0	6695	SP_0	/dev/vgtpcc_raw/r10i_lvm_sp_0
stok_0_0		37	8042	STOK_0	/dev/rdsk/c28t0d1
stok_0_1		38	8042	STOK_0	/dev/rdsk/c5t0d3
stok_0_10		18	8042	STOK_0	/dev/rdsk/c29t0d3
stok_0_11		20	8042	STOK_0	/dev/rdsk/c53t0d1
stok_0_12		21	8042	STOK_0	/dev/rdsk/c23t0d1
stok_0_13		23	8042	STOK_0	/dev/rdsk/c5t0d2
stok_0_14		26	8042	STOK_0	/dev/rdsk/c6t0d1
stok_0_15		28	8042	STOK_0	/dev/rdsk/c30t0d1
stok_0_16		30	8042	STOK_0	/dev/rdsk/c5t0d1
stok_0_17		42	8042	STOK_0	/dev/rdsk/c6t0d2
stok_0_18		44	8042	STOK_0	/dev/rdsk/c24t0d2
stok_0_19		46	8042	STOK_0	/dev/rdsk/c54t0d1
stok_0_2		39	8042	STOK_0	/dev/rdsk/c23t0d3
stok_0_20		49	8042	STOK_0	/dev/rdsk/c31t0d1
stok_0_21		51	8042	STOK_0	/dev/rdsk/c26t0d2
stok_0_22		53	8042	STOK_0	/dev/rdsk/c25t0d1
stok_0_23		55	8042	STOK_0	/dev/rdsk/c4t0d2
stok_0_24		58	8042	STOK_0	/dev/rdsk/c23t0d2
stok_0_25		60	8042	STOK_0	/dev/rdsk/c46t0d3
stok_0_26		62	8042	STOK_0	/dev/rdsk/c26t0d3
stok_0_27		63	8042	STOK_0	/dev/rdsk/c24t0d3
stok_0_28		65	8042	STOK_0	/dev/rdsk/c6t0d3
stok_0_29		66	8042	STOK_0	/dev/rdsk/c28t0d2
stok_0_3		40	8042	STOK_0	/dev/rdsk/c31t0d3
stok_0_30		67	8042	STOK_0	/dev/rdsk/c46t0d1
stok_0_31		68	8042	STOK_0	/dev/rdsk/c29t0d1
stok_0_32		69	8042	STOK_0	/dev/rdsk/c31t0d2
stok_0_33		70	8042	STOK_0	/dev/rdsk/c26t0d1
stok_0_34		71	8042	STOK_0	/dev/rdsk/c30t0d3
stok_0_35		90	8042	STOK_0	/dev/rdsk/c25t0d2
stok_0_36		91	8042	STOK_0	/dev/rdsk/c54t0d4
stok_0_37		92	8042	STOK_0	/dev/rdsk/c25t0d3

stok_0_38	6	8042	STOK_0	/dev/rdsk/c32t0d2
stok_0_39	10	8042	STOK_0	/dev/rdsk/c54t0d2
stok_0_4	5	8042	STOK_0	/dev/rdsk/c32t0d3
stok_0_40	19	8042	STOK_0	/dev/rdsk/c24t0d1
stok_0_5	7	8042	STOK_0	/dev/rdsk/c29t0d2
stok_0_6	9	8042	STOK_0	/dev/rdsk/c32t0d1
stok_0_7	11	8042	STOK_0	/dev/rdsk/c53t0d2
stok_0_8	14	8042	STOK_0	/dev/rdsk/c46t0d2
stok_0_9	16	8042	STOK_0	/dev/rdsk/c28t0d3
system_001	1	200	SYSTEM	/dev/vgtpcc_raw/r10i_lvm_system_001
ware_0_0	4	20	WARE_0	/dev/vgtpcc_raw/r10i_lvm_ware_0_0

The distribution of the database tables over the disk arrays of the priced system is an extension of the distribution described in Table 5.2; some ancillary details are mentioned in Appendix E. 60-day storage growth requirements are met with the unused space of this configuration. Figure 1.2 shows the configuration of the priced-system disks.

5.3 Data Model & Interfaces

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 used (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.*

Oracle10i Database Standard Edition is a relational DBMS. SQL stored procedures were used, invoked through the Oracle Call Interface (OCI); the application code appears in Appendix A.

5.4 Partitions/Replications

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

No partitioning or replication was used.

5.5 Growth Requirements

Details of the 60 day space computations along with proof that the database is configured to sustain 8 hours for the dynamic tables (Order, Order-Line, and History) must be disclosed.

See Appendix E.

6 Clause 5 Related Items

6.1 Throughput

Measured tpmC must be reported.

Table 6.1: Measured tpmC

tpmC®	80,570.81

6.2 Response Time

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

Table 6.2: Response Times

Response Times	Average	90th %-ile	Maximum
New-Order	0.61s	1.11s	33.45s
Payment	0.59s	1.10s	33.52s
Order-Status	0.62s	1.12s	32.64s
Delivery (interactive portion)	0.24s	0.61s	32.27s
Delivery (deferred portion)	0.61s	1.11s	32.93s
Stock-Level	0.61s	1.13s	32.67s
Menu	0.000004s	0.0001s	0.00125s

6.3 Keying and Think Times

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

Table 6.3: Keying Times

Keying Times	Minimum	Average	Maximum
New Order	18.02s	18.03s	18.18s
Payment	3.01s	3.02s	3.18s
Order Status	2.01s	2.02s	2.18s
Interactive Delivery	2.01s	2.02s	2.18s
Stock Level	2.01s	2.02s	2.18s

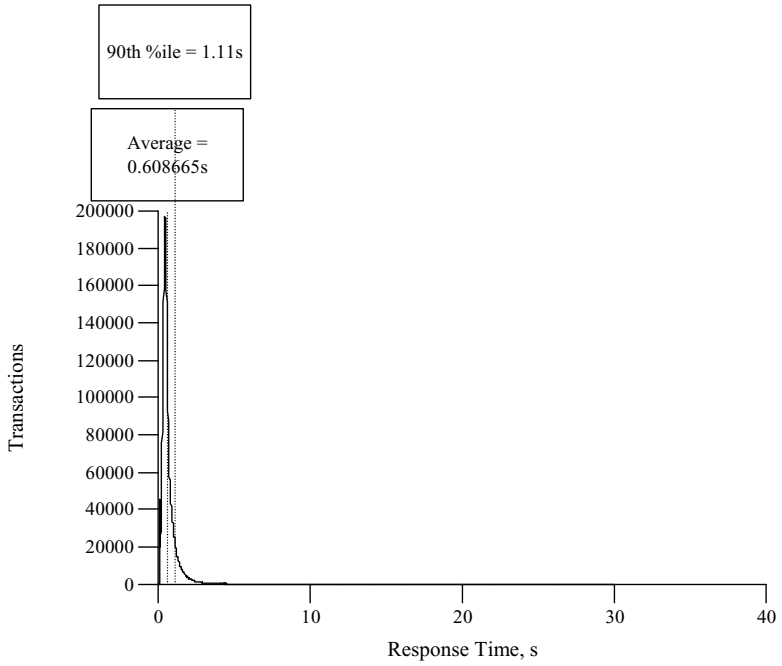
Table 6.4: Think Times

Think Times	Minimum	Average	Maximum
New Order	0.01s	12.13s	173.27s
Payment	0.01s	12.07s	187.1s
Order Status	0.01s	10.11s	144.01s
Interactive Delivery	0.01s	5.05s	67.78s
Stock Level	0.01s	5.06s	68.44s

6.4 Response Time Frequency Distribution Curves and Other Graphs

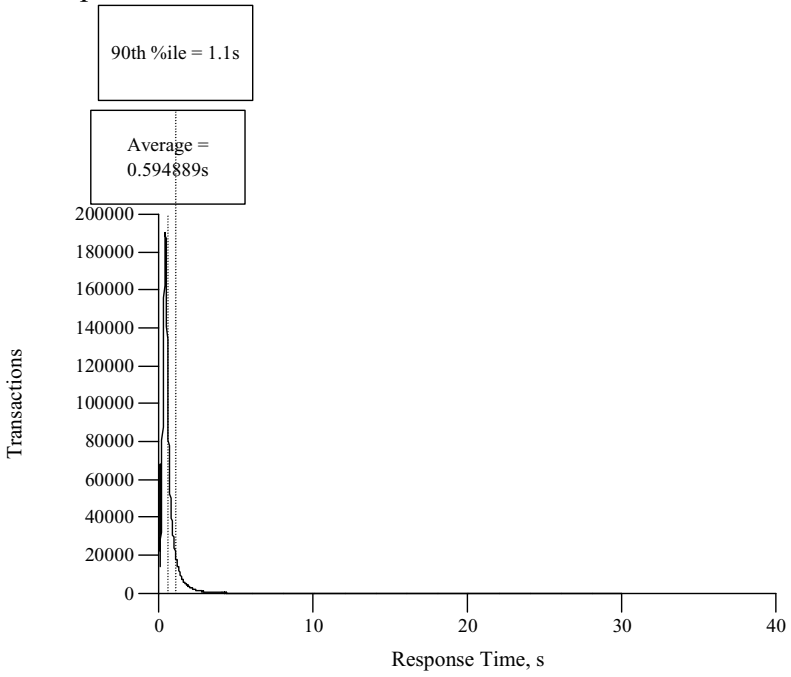
Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type. The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction. The Think Time frequency distribution curve (see Clause 5.6.3) must be reported for the New-Order transaction. A graph of throughput versus elapsed time (see Clause 5.6.5) must be reported for the New-Order transaction, and the measurement interval indicated.

Figure 6.1: New Order Response Time Distribution



Response time frequency distribution for New Order transaction

Figure 6.2: Payment Response Time Distribution



Response time frequency distribution for Payment transaction

Figure 6.3: Order Status Response Time Distribution

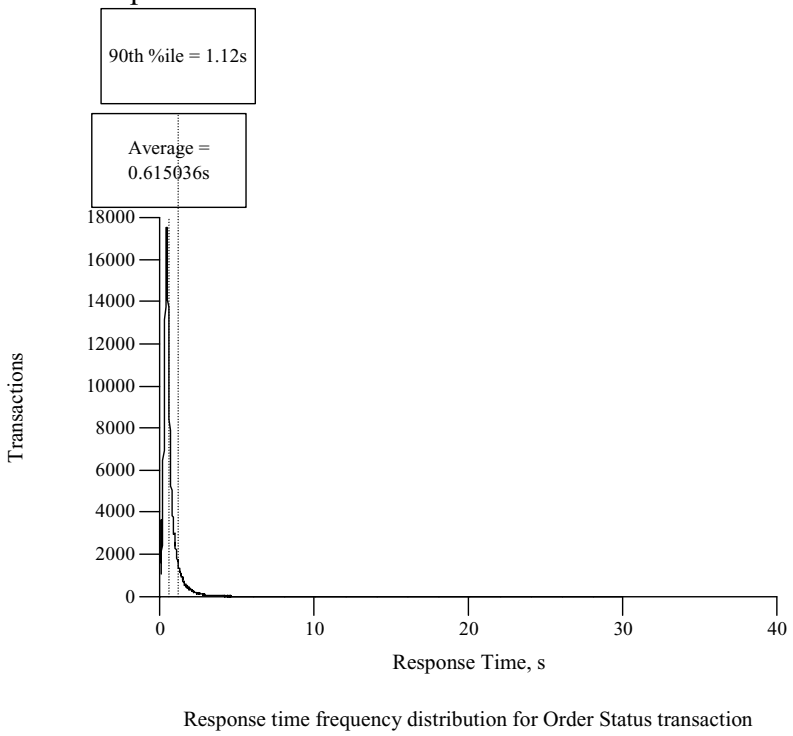


Figure 6.4: (Interactive) Delivery Response Time Distribution

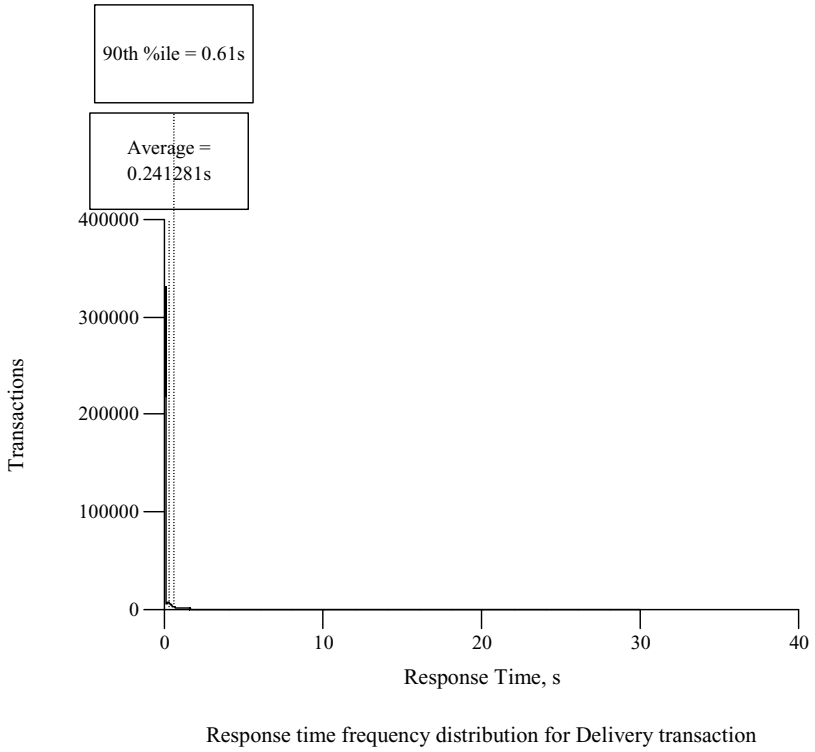
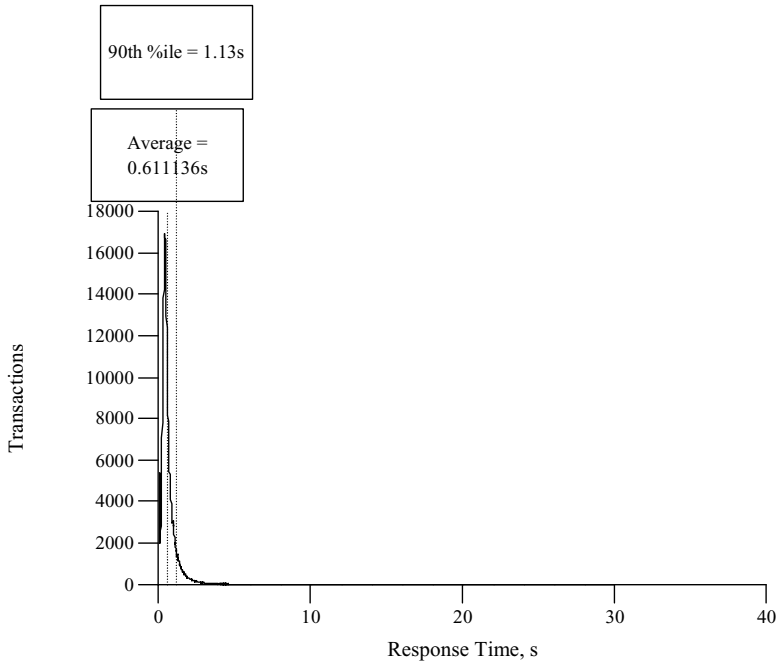
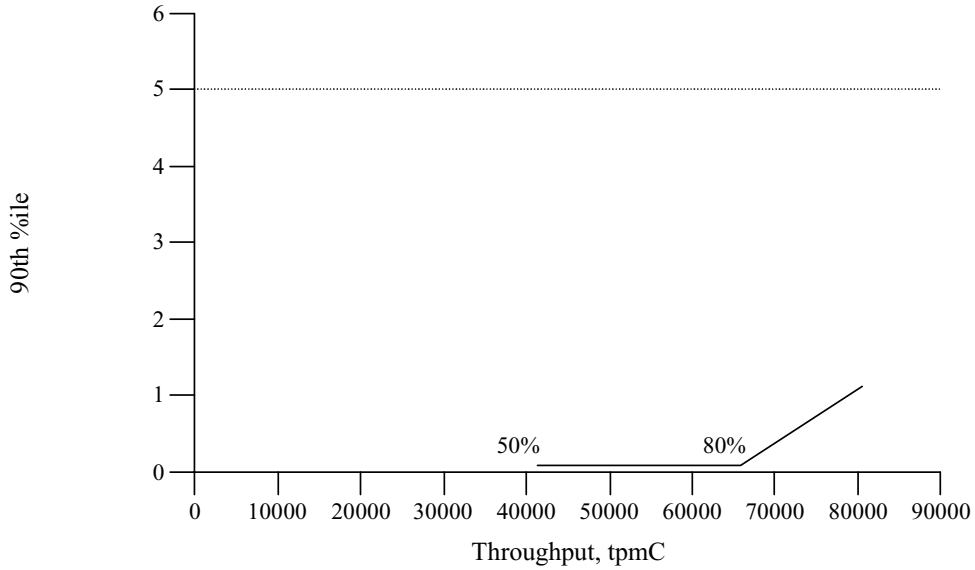


Figure 6.5: Stock Level Response Time Distribution



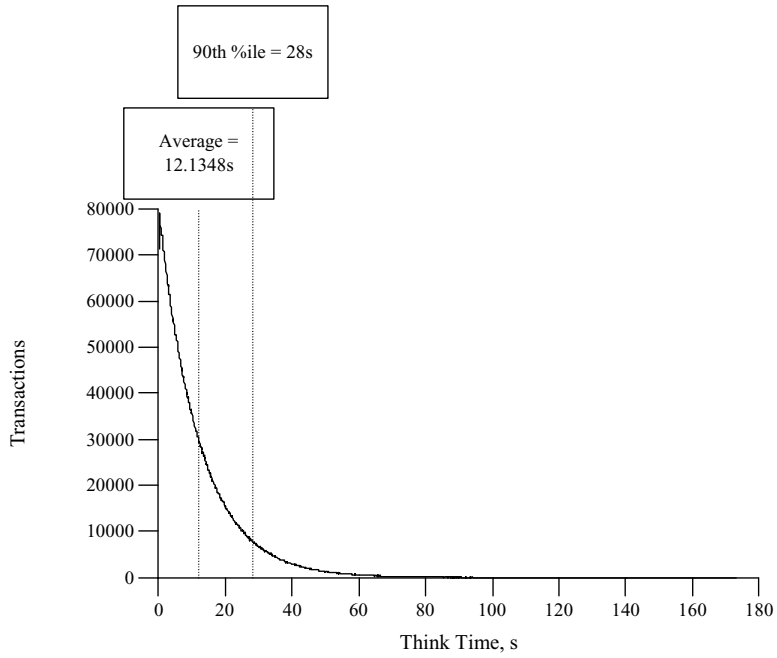
Response time frequency distribution for Stock Level transaction

Figure 6.6: Response Time Versus Throughput



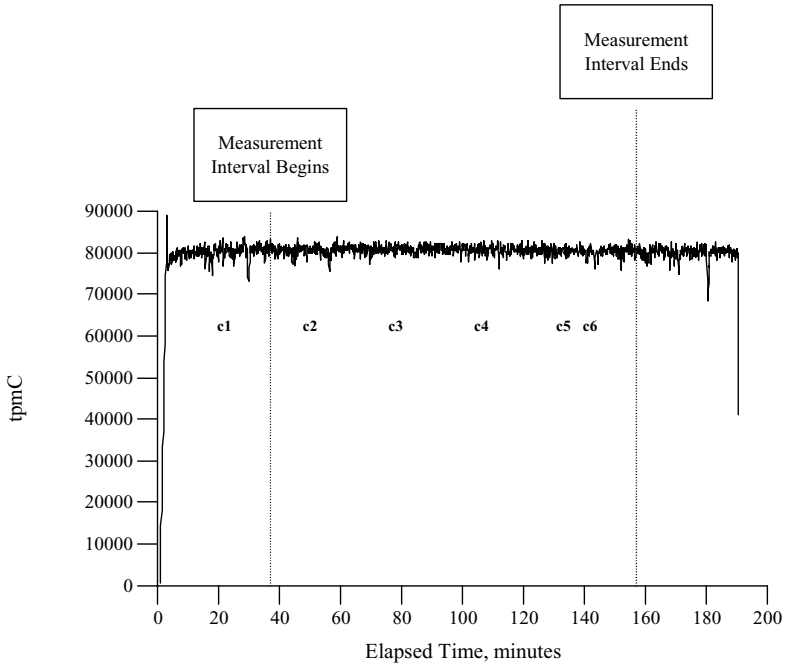
New Order response time versus Throughput

Figure 6.7: New Order Think Time Distribution



Think time frequency distribution for New Order transaction

Figure 6.8: Throughput Versus Time



Throughput of the New-Order transaction versus elapsed time

6.5 Steady State Determination

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

Synchronization techniques employed in the benchmark process ensure that all emulated users are logged into the client and have opened the application before submitting transactions. Once all users are connected, each pauses a random amount of time before submitting transactions. The pause time distribution is controlled by a benchmark input parameter. The ramp-up interval is discernible in the graph of throughput over time. The data reduction also tracks the user load and indicates the point in time at which all users have submitted at least one transaction. The throughput is observed to be steady within the systematic and statistical variability of the measurement after all users are submitting transactions. A checkpoint is initiated upon the end of ramp-up.

6.6 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 migrated to disk on a least-recently-used 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 either when the transaction committed or when the redo log buffer became full. However, due to the rapid commit 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 many transactions' redo log data being written to disk. This is called group commit.

6.6.1 Checkpoint

During an Oracle10i Database Standard Edition checkpoint, all modified blocks in the shared buffer cache which had not been written to disk since the last checkpoint are written to disk.

6.6.2 Checkpoint Conditions

Oracle10i Database Standard Edition performs a checkpoint for 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 amount of time since the last checkpoint reaches the log_checkpoint_timeout.

6.6.3 Checkpoint Implementation

The first method listed above, i.e., a log switch when the redo log file filled up, was used to cause checkpoints. After the initial checkpoint, a log switch was performed every 29.51 minutes in average. All checkpoint intervals were less than 30 minutes.

6.6.4 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 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 of the scope of read consistency from the individual query to the entire transaction. Instead of limiting a query to data committed at the time a 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 as of the beginning of the transaction.

All reads within a serializable transaction see 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 behavior 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 transactions 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 statement receiving the error.

6.7 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 120 minutes.

6.8 Regulation of Transaction Mix

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 selection method of *Clause 5.2.4.1* was used. The weights were not adjusted during the run.

6.9 Transaction Mix

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

Table 6.5: Transaction Mix

Type	Percentage
New Order	44.96%
Payment	43.01%
Order Status	4.01%
Delivery	4.002%
Stock Level	4.01%

6.10 Transaction Statistics

The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed. The average number of order-lines entered per New-Order trasaction must be disclosed. The percentage of remote order-lines entered per New-Order transaction must be disclosed. The percentage of remote Payment transactions must be disclosed. The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed. The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.

See Table 3.1

6.11 Checkpoint Count and Location

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.

A checkpoint was completed before the measurement interval began. There were four checkpoints completed within the measurement interval. The first checkpoint starts approximately 1.4 minutes into the measurement interval.
The average checkpoint interval is 29.5 minutes and a checkpoint lasts approximately 26.6 minutes

The run started at 18:10:21. The Measurement Interval was 18:47:21 to 20:47:21.

The checkpoints during this run were:

Checkpoint	Start time	End time	Duration
	18:10:21	run starts	
#0	18:19:05	18:45:57	26:52
	18:47:21	measurement starts	
#1	18:48:45	19:15:25	26:40
#2	19:18:14	19:45:02	26:48
#3	19:47:51	20:14:30	26:39
#4	20:17:17	20:43:25	26:08
	12:47:21	measurement ends	

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 input parameters, code fragments, functions, et cetera used to generate each transaction input field must be disclosed. Comment: The intent is to demonstrate the RTE was configured to generate transaction input data as specified in Clause 2.

The RTE (Remote Terminal Emulator) on the driver system was developed at Hewlett-Packard and is not commercially available. Appendix D lists RTE input parameters and code fragments used to generate each transaction input field.

For this instance of the TPC-C benchmark, 7 drivers and 7 clients were used. The drivers emulated users logged in to the clients. An overview of the benchmark software on the drivers, clients and server is shown in Figure 7.1.

The benchmark is started with the **run** command on the driver system. **Run** controls the overall execution of the benchmark. After reading a configuration file, **run** starts TUXEDO on the client, collects pre-benchmark audit information and inserts a timestamp into a database audit table. When all the initial steps are completed, **run** invokes another program, **driver,** to start the benchmark. As the benchmark completes, **run** shuts down TUXEDO and collects the benchmark results into a single location.

Driver is the heart of the benchmark software. It simulates users as they log in, execute transactions and view results. **Driver** collects response times for each transaction and saves them in a file for future analysis.

Qualify is the post-processing analysis program. It produces the numerical summaries and histograms needed for the disclosure report.

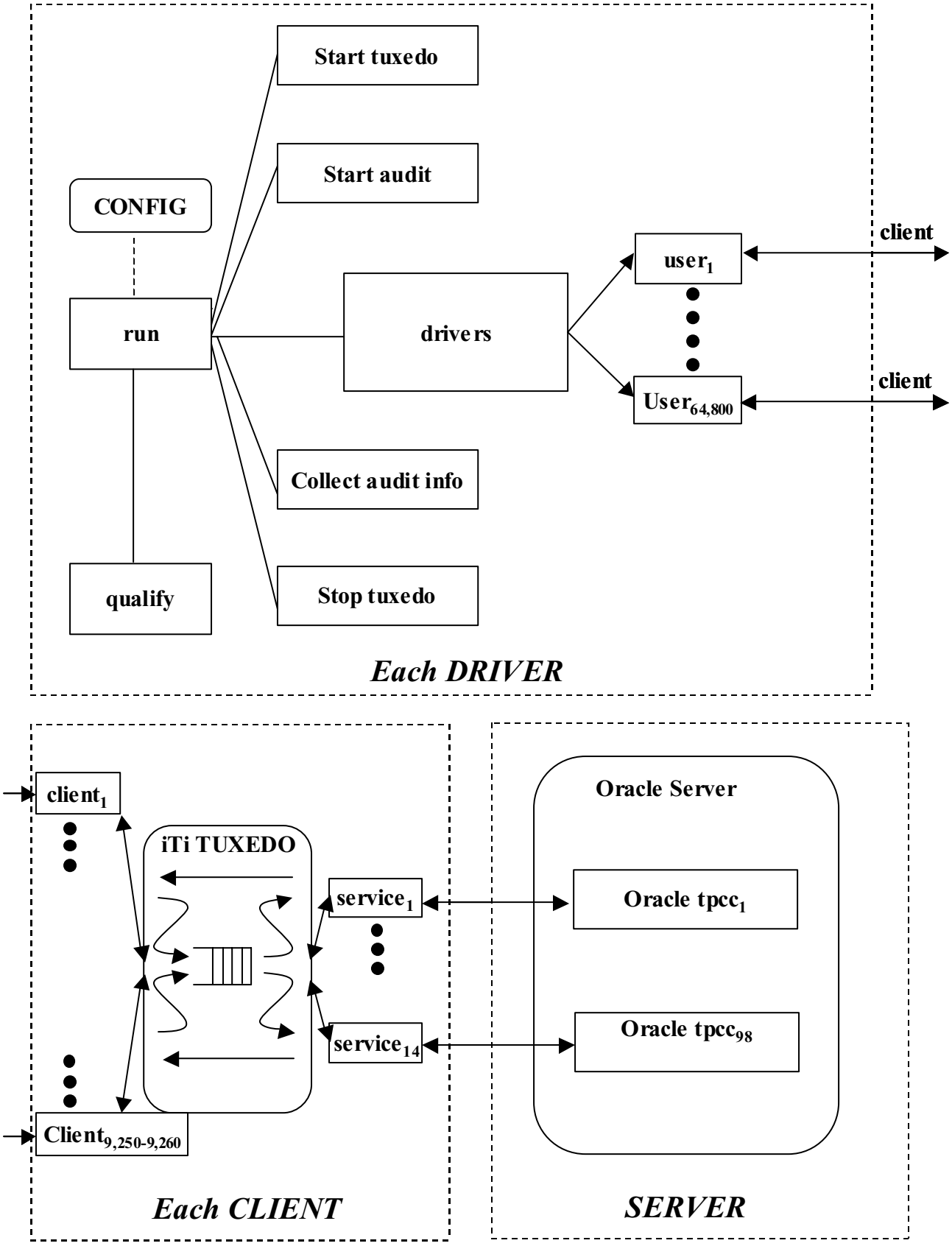


Figure 7.1: Benchmark Software

7.2 Lost Connections

No terminal connections were lost during the measurement interval**.**

7.3 Emulated Components

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

In the benchmark configuration, the 64,800 simulated workstations connected to the clients over 7 100BT lans through a single hp procurve switch In the priced configuration, the 64,800 worksations would connect to the clients via a combination of hubs and switches which eventually mutipexed down to 7 100BT lans.

7.4 Functional Diagrams

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

Figures 1.1 and 1.2 (in Chapter 1) show functional diagrams of the benchmark and configured systems. A description of the RTE and benchmark software is provided above.

7.5 Networks

The network configuration of both the tested and proposed services which are being represented and a thorough explanation of exactly which parts are being replaced with the Driver System must be disclosed.

Figures 1.1 and 1.2 (in Chapter 1) diagram the network configurations of the benchmark and configured systems, and represent the Driver connected via LAN replacing the workstations and HUBs connected via LANs. The clients are connected via 100 Base-T to an 100BT/1000BT-Ethernet switch which in turn is connected via 1000BT-Ethernet to the SUT.

The bandwidth of the networks used in the tested/priced configurations must be disclosed.

Ethernet and 100 Base-T local area networks (LAN) with a bandwidth of 100 megabits per second are used in the tested/priced configurations. The 1000BT used has a bandwidth of 1000 megabits per second.

7.6 Client Substitution

No client substitution was used.

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 item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery data. If package-pricing is used contents of the package must be disclosed. Pricing source(s) and effective date(s) of price(s) must also be reported.

The total 3-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.

Each priced configuration consists of an integrated system package, additional options, and components. Prices for all Hewlett-Packard products that are not provided by a third party quote are HP's US list prices. A one (1) year warranty is standard with all Hewlett-Packard products.

8.2 Support Pricing

The three year support pricing for Hewlett-Packard products is based on twenty-four (24) months of monthly support costs; thirty-six (36) months minus the twelve month warranty period. The Oracle Corporation support pricing is based on thirty-six (36) months of monthly support costs. The following support products were priced in the benchmark:

- HP four-hour on-site repair hardware support,
- HP telephone support for software and updates
- Oracle Corporation Standard Technical Support and,
- BEA TUXEDO Standard Technical Support

8.2.1 HP Hardware Support

HP's on-site support for hardware provides service 24 hour, seven day support.

8.2.2 HP Software Support

HP Software Support provides the following:

- Access to the HP Response Centers for fault isolation and problem solving assistance,
- Guaranteed two (2) hour call return, immediate response for critical calls,
- Electronic access to product and support information,
- Electronic access to software patches,
- Right-to-use and copy software updates.

8.3 Oracle Corporation Standard Technical Support

Oracle Corporation Standard Technical Support includes:

Product updates,

- A regular technical publication,
- Unlimited, toll-free telephone service to assist in product installation, syntax, and usage that is available 24 hours, seven days a week.

8.4 Availability

The committed delivery date for general availability (availability date) of products used in the price calculation 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.

see below

8.5 Priced System Configuration

The hardware, software, and support/maintenance products priced in this benchmark are detailed on page *v*.

8.6 Throughput, Price/Performance, and Availability Date

A statement of the measured tpmC® *as well as the respective calculations for the 3-year pricing, price/performance (price/tpmC®).*

For Throughput and Price/Performance, please see page iv and v. The Price/Performance calculation spreadsheet appears on page v.

All hardware components in this test of the hp server rx5670 system are currently available. HP-UX 11.i, v1.6 with Performance Pack, 64-bit incorporating will be available April 30, 2003. Oracle10i Database Standard Edition will be available May 11, 2003.

9 Clause 9 Related Items

9.1 Auditor's Report

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

If audited, the auditor's attestation letter must be made readily available to the public as part of the Full Disclosure Report, but a detailed report from the auditor is not required.

This implementation of the TPC Benchmark® C on the hp server rx5670 was audited by Lorna Livingtree for Performance Metrics, Inc..

Lorna Livingtree
Performance Metrics, Inc.
137 Yankton Street, Suite 101
Folsom, CA 95630
U.S.A.
Phone: 916 985-1131
Fax: 916 985-1185

The attestation letter is shown on the following pages.

November 11, 2002

Andreas Hotea
Performance Manager
Hewlett-Packard Company
19111 Pruneridge Avenue
Cupertino CA 95014

I have verified by remote the TPC Benchmark™ C for the following configuration:

Platform: **hp server rx5670 Client/Server**
Database Manager: **Oracle10i Database Standard Edition**
Operating System: **HP-UX 11i 64-bit Base OS**
Transaction Monitor: **BEA TUXEDO 8**

Server: hp server rx5670 Client/Server				
CPU's	**Memory**	**Disks (total)**	**90% Response**	**TpmC**
4 Itanium 2 @ 1 GHz	**Main: 48 GB L3 Cache: 3 MB**	**227 @ 18GB 15 @ 73GB**	**1.11**	**80,570.81**
7 Clients: hp server rp2470				
1 PA-RISC 8700 @ 750 MHz	**Main: 3 MB I-cache: 750 KB D-cache: 1.5MB**	**1 @ 18GB**	**Na**	**Na**

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

The following attributes of the benchmark were given special attention:

- **The transactions were correctly implemented.**
- **The database files were properly sized and populated.**
- **The database was properly scaled with 6,700 warehouses of which 6,480 were active during the measured interval. The columns w_ytd and d_next_oid contained initial values for the unused warehouses.**
- **The ACID properties were successfully demonstrated.**
- **Data loss durability was demonstrated on the SUT with 64,800 active users.**
- **Input data was generated according to the specified percentages.**
- **Eight hours of mirrored log space was present on the tested system.**
- **Eight hours of growth space for the dynamic tables was present on the tested system.**
- **The data for the 60-day space calculation was verified. Three arrays of 36GB disks were substituted in the priced configuration – see Auditor Notes.**
- **The CNUM for load was "1"; the CNUM for the RTE was "86".**
- **The steady state portion of the test was 120 minutes.**
- **One checkpoint was taken before the measured interval.**
- **Four checkpoints were taken during the measured interval.**
- **Pricing for maintenance and component counts was verified.**

Auditor Notes:

This measurement used the Sorted Hash Cluster access strategy which is new to Oracle 10i. Additional tests were performed to verify that primary keys could be updated and that multiple updates to the same row in one transaction performed properly.
An earlier experiment had shown that the 36GB disks were slightly faster than their 18GB counterpart. In addition, the 2 18GB drives in the server enclosure were replaced with 1 36GB drive. The first drive had the root directory and the 2nd drive was empty.
There were additional disks connected to the database server which were used for backup; I verified that these disks were not used during the measurements.

Sincerely,

Lorna Livingtree
Auditor

10 Report Availability

Requests for this TPC Benchmark C Full Disclosure Report should be sent to:

Transaction Processing
Performance Council
c/o Shanley Public Relations
650 N. Winchester Blvd.
Suite 1
San Jose, CA 95128

or your local Hewlett-Packard sales office.

Appendix A Client/Server Source

This appendix contains the source and makefiles for all client and server programs.

A.1 Client Front-End

client/client.c

```
/*********************************************************************************
 @(#) Version: A.10.10 $Date: 2002/08/20 16:06:01 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*********************************************************************************/
/*********************************************************************
History
941101 JVM Fixed login screen to detect broken connection (used to loop)
941013 JVM Added audit strings to the login form
941013 VM modified the getfield procedure to add digit and char check
      according to the field type.
941014 VM added the status_msg routine to display transaction results.
941015 VM added zip routine to format zip codes and phone routine
      to format phone numbers.
*********************************************************************/
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <fcntl.h>
#include <errno.h>
#include <pthread.h>

#include "key_chars.h"
#include "tpcc.h"

/*
 * Input/Output Buffer management
 */
typedef struct {
   int  ifd;   /* input file descriptor */
   int  ofd;   /* output file descriptor */
   char *beg;
   char *end;   /* for output buffers */
   char *max;
   char *cur;   /* for input buffers */
} iobuf;


/* Macro do define an I/O buffer of x characters, initialized to empty */
#define define_iobuf(name, size)          \
__thread char name##_data[size];         \
__thread iobuf name[1]


#define init_iobuf(name, size, _ifd, _ofd)   \
    name->ifd = _ifd;                \
    name->ofd = _ofd;                \
    name->beg = name##_data;         \
    name->end = name##_data;         \
    name->max = name##_data+size;      \
    name->cur = name##_data;

#define reset(b)  if (1) {                     \
   (b)->cur = (b)->end = (b)->beg;              \
   *(b)->beg = '\0';                          \
   } else (void)0

#define flush(b)  if(1) {                       \
       display(b);                        \
       reset(b);                          \
    } else (void)0

#define pushc(b,c) if (1) {                                        \
   if ((b)->end >= (b)->max) {                                      \
       error("out_buf overflow:  beg=0x%x end=%d  max=%d\n",              \
        (b)->beg, (b)->end-(b)->beg,(b)->max-(b)->beg);                  \
   }                                                                \
   *((b)->end++) = (c);                                             \
   *((b)->end) = '\0';  /* debug */                                  \
   } else (void)0

/*
 * Input/Output buffers + screen buffers
 */

#define INPUT_BUF_SIZE 1024
#define OUTPUT_BUF_SIZE 4096
#define NEWORDER_FORM_SIZE 900
#define PAYMENT_FORM_SIZE 400
#define ORDSTAT_FORM_SIZE 300
#define DELIVERY_FORM_SIZE 300
#define STOCKLEV_FORM_SIZE 300

define_iobuf(output_stuff, OUTPUT_BUF_SIZE);
define_iobuf(input_stuff, INPUT_BUF_SIZE);
define_iobuf(payment_form, PAYMENT_FORM_SIZE);
define_iobuf(neworder_form, NEWORDER_FORM_SIZE);
define_iobuf(ordstat_form, ORDSTAT_FORM_SIZE);
define_iobuf(delivery_form, DELIVERY_FORM_SIZE);
define_iobuf(stocklev_form, STOCKLEV_FORM_SIZE);

/*
 * global variables set up during initialization
 */
__thread int    user;
__thread ID     warehouse;
__thread ID     district;
__thread iobuf *in_buf;
__thread iobuf *out_buf;

/* Number of Threads per Tuxedo Context */
#define MAX_THREADS_PER_CONTEXT 16

/* Maximum number of threads per server */
#define MAX_USERS_PER_PROCESS   1024

/* Process local only */
long tux_context;  /* Tuxedo context to use */

int port_number                 = 11000; /* address to listen on */
int user_connections              = 0;/* number of current connections*/
int number_of_servers             = 15; /* number of servers to spawn */
pthread_t user_ids[MAX_USERS_PER_PROCESS] = {0}; /* thread ids spawned per server */

struct thread_data {
  int  fd;          /* Stream file descriptor */
  long tux_context;  /* Tuxedo context to use */
};
typedef struct thread_data thread_data;



/*
 * Prototype definitions
 */
static void display(iobuf *scr);
static int  getkey(void);
static int  next_field(int current, int key, int max);
static int  neworder(neworder_trans *t);
static int  neworder_read(neworder_trans *t);
static void neworder_write(neworder_trans *t);
static void neworder_setup(void);
static int  payment(payment_trans *t);
static void payment_setup(void);
static int  payment_read(payment_trans *t);
static void payment_write(payment_trans *t);
static int  ordstat(ordstat_trans *t);
static void ordstat_setup(void);
static int  ordstat_read(ordstat_trans *t);
static void ordstat_write(ordstat_trans *t);
static int  delivery(delivery_trans *t);
static void delivery_setup(void);
static int  delivery_read(delivery_trans *t);
static void delivery_write(delivery_trans *t);
static int  stocklev(stocklev_trans *t);
static void stocklev_setup(void);
static int  stocklev_read(stocklev_trans *t);
static void stocklev_write(stocklev_trans *t);
static int  valid_char(int key, FIELD_TYPE ftype);
static int  getfield(int row, int col, char buf[], int width, FIELD_TYPE ftype);
static int  read_text(int row, int col, char *s, int width);
static int  read_money(int row, int col, double *m, int width);
static int  read_number(int row, int col, int *n, int width);
static void clear_screen(void);
static void position(int row, int col);
static void trigger(void);
static void trigger2(void);
static void status(int row, int col, int status);
static void blanks(int row, int col, int len);
static void empty(int row, int col, int len);
static void zip(int row, int col, char *str);
static void phone(int row, int col,char *str);
static void text(int row, int col, char str[]);
```

```
static void long_text(int row, int col, char *str, int width);
static void money(int row, int col, double x, int width);
static void date_only(int row, int col, char *date_str);
static void date(int row, int col, char *date_str);
static void real(int row, int col, double x, int width, int dec);
static void number(int row, int col, int  n, int width);
static void string(char str[]);
static void cleanup(void);
static int  setup(int fd);
static void msgline(char *str);
static int  menu_read(void);
static void menu_setup(void);
static int  login(void);

void *
client_main(void *arg)
{
   int key;
   /* a generic transaction variable. */
   generic_trans generic_transaction;

   thread_data *td = (thread_data *)arg;
   generic_trans *trans=&generic_transaction;

   /* setup Tuxedo Context */
   thread_transaction_begin(td->tux_context);

   /* setup the transactions */
   key = setup(td->fd);

   /* repeat until done */
   while (key != '9' && key != EOF)
     {

      /* get the menu choice */
      key = menu_read();

      /* process according to the choice */
      switch(key)
         {
         case '1': key = neworder(&trans->neworder); break;
         case '2': key = payment(&trans->payment); break;
         case '3': key = ordstat(&trans->ordstat); break;
         case '4': key = delivery(&trans->delivery); break;
         case '5': key = stocklev(&trans->stocklev); break;
         case EOF: break;
         case '9': break;
         default:    msgline("Please enter a valid menu choice");
         }
      }

   /* done */
   cleanup();

   /* Close socket */
   close(td->fd);

   /* Exit Thread */
   pthread_exit(NULL);
}



/*************************************************************************
**************************************************************************

Neworder form processing

**************************************************************************
**************************************************************************/


static int
neworder(neworder_trans *t)
{
   int key;
   display(neworder_form);
   key = neworder_read(t);
   if (key != ENTER) return key;
   neworder_transaction(t);
   neworder_write(t);
   return key;
}


static int
neworder_read(neworder_trans *t)
{
   int i;
   int field;
   int key;
   int ol;
                int ol_count;
                int all_local;
                int move_slot;

   /* Our warehouse number is fixed */
   t->W_ID = warehouse;
   t->D_ID = EMPTY_NUM;

   /* assume nothing set yet */
   t->C_ID = EMPTY_NUM;
   for (i=0; i<15; i++)
      {
      t->item[i].OL_I_ID = EMPTY_NUM;
      t->item[i].OL_QUANTITY = EMPTY_NUM;
      t->item[i].OL_SUPPLY_W_ID = EMPTY_NUM;
      }

   /* Process fields until done */
   for (field = 1; field > 0; field = next_field(field, key, 47))
      retry: switch (field)
         {

         case 1: key = read_number(4, 29, &t->D_ID, 2);
                 break;

         case 2: key = read_number(5, 12, &t->C_ID, 4);
                 break;

         case 3: case 6: case 9: case 12: case 15:
         case 18: case 21: case 24: case 27: case 30:
         case 33: case 36: case 39: case 42: case 45:
            ol = (field - 3) / 3;
            key = read_number(9+ol, 2, &t->item[ol].OL_SUPPLY_W_ID,6);
            break;

         case 4: case 7: case 10: case 13: case 16:
         case 19: case 22: case 25: case 28: case 31:
         case 34: case 37: case 40: case 43: case 46:
            ol = (field - 3) / 3;
            key = read_number(9+ol,10, &t->item[ol].OL_I_ID, 6);
            break;

         case 5: case 8: case 11: case 14: case 17:
         case 20: case 23: case 26: case 29: case 32:
         case 35: case 38: case 41: case 44: case 47:
            ol = (field - 3) / 3;
            key = read_number(9+ol, 45, &t->item[ol].OL_QUANTITY, 2);
            break;
         }

   /* abort the screen if requested */
   if (key != ENTER)
      return key;

   /* make sure all necessary fields are filled in */
   if (t->D_ID == EMPTY_NUM)
      {field=1; msgline("Please specify district"); goto retry;}
   if (t->C_ID == EMPTY_NUM)
      {field=2; msgline("Please specify customer id"); goto retry;}

   /* calculate how many items were entered */
                ol_count = 0;
                all_local = 1;
                move_slot = -1;
   for (i=0; i < 15; i++) {
      if ((t->item[i].OL_I_ID == EMPTY_NUM) &&
                  (t->item[i].OL_SUPPLY_W_ID == EMPTY_NUM) &&
                  (t->item[i].OL_QUANTITY == EMPTY_NUM)) {
                                    /* All are clear, so no item */
                                    if (move_slot == -1) {
                                                move_slot = i;
                                    }
                        } else {
                                    /* this is potentially an order line, so check it out */
                  if (t->item[i].OL_SUPPLY_W_ID == EMPTY_NUM) {
                     field=i*3+3;
                                                msgline("Please enter supply
warehouse");
                                                goto retry;
                  }
                        if (t->item[i].OL_I_ID == EMPTY_NUM) {
                                                field=i*3+4;
                                                msgline("Please enter Item id");
                                                goto retry;
                  }
                        if (t->item[i].OL_QUANTITY == EMPTY_NUM ||
                     t->item[i].OL_QUANTITY <= 0) {
                                                field=i*3+5;
                                                msgline("Please enter quantity > 0");
                                                goto retry;
                                    }
                                    /* It is a complete orderline, so count it */
                                    ol_count++;

                              /* decide if they were all local */
                  if (t->item[i].OL_SUPPLY_W_ID != t->W_ID) {
                                                all_local = 0;
                                    }
```

```
                                        if (move_slot != -1) {
                                                /* Move the item up to fill in a hole */
                                                t->item[move_slot] = t->item[i];

                                                move_slot++; /* bump up to the next
slot */
                                        }
                                }
    }

    if (ol_count == 0)
        {field=3; msgline("Please enter at least one orderline"); goto retry;}

    t->O_OL_CNT  = ol_count;
    t->all_local = all_local;

    /* display number of order lines */
    number(6, 42, t->O_OL_CNT, 2);

    msgline("");
    flush(out_buf);
    return key;
}


static void
neworder_write(neworder_trans *t)
{
    int i;
    MONEY amount, total_amount, cost;

    /* Rev. 3.3 error checking: both of the following branches are
     * skipped.  We'll go to status and print an error message.
     */

    /* CASE: invalid item, display only these values */
    if (t->status == E_INVALID_ITEM)
        {
        text(5, 25, t->C_LAST);
        text(5,52,  t->C_CREDIT);
        number(6, 15, t->O_ID, 8);
        }

    /* CASE: everything OK, display everything */
    else if (t->status == OK)
        {
        text(5, 25, t->C_LAST);
        text(5,52,  t->C_CREDIT);
        number(6, 15, t->O_ID, 8);
        date(4, 61, t->O_ENTRY_D);
        real(5, 64, t->C_DISCOUNT * 100, 5, 2);
        real(6, 59, t->W_TAX*100, 5, 2);
        real(6, 74, t->D_TAX*100, 5, 2);

        total_amount = 0;
        for (i=0; i < t->O_OL_CNT; i++)
            {

            /* keep track of amount of each line and total */
            amount = t->item[i].I_PRICE * t->item[i].OL_QUANTITY;
            total_amount += amount;

            /* display the item line */
            number(9+i, 2, t->item[i].OL_SUPPLY_W_ID,6);
            number(9+i,10, t->item[i].OL_I_ID, 6);
            text(9+i, 19, t->item[i].I_NAME);
            number(9+i,45, t->item[i].OL_QUANTITY, 2);
            number(9+i, 51, t->item[i].S_QUANTITY, 3);
            position(9+i, 58);   pushc(out_buf,t->item[i].brand_generic);
            money(9+i, 62, t->item[i].I_PRICE, 7);
            money(9+i, 71, amount, 8);
            }

        /* Clear the screen of any empty input fields */
        clear_screen();

        /* display the total cost */
        text(24, 63, "Total:");
        cost = total_amount * (1 - t->C_DISCOUNT) * (1 + t->W_TAX + t->D_TAX);
        money(24, 71, cost, 9);
        }

    /* display the status message */
    status(24, 1, t->status);
}

static void
neworder_setup(void)
{
    int item;
    iobuf *old;

    /* start with an empty form */
    reset(neworder_form);

    /* redirect the data to a special menu buffer */
    old = out_buf;  out_buf = neworder_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 36, "New Order");
    text(4, 1, "Warehouse:");
    number(4, 12, warehouse, 6);
    text(4, 19, "District:");
    empty(4, 29, 2);
    text(4, 55, "Date:");
    text(5, 1, "Customer:");
    empty(5, 12, 4);
    text(5, 19, "Name:");
    text(5, 44, "Credit:");
    text(5, 57, "Disc.:");
    text(6, 1, "Order Number:");
    text(6, 25, "Number of Lines:");
    text(6, 52, "W_Tax:");
    text(6, 67, "D_Tax:");
    text(8, 2, "Supp_W Item_Num   Item_Name");
    text(8, 45, "Qty Stock   B/G    Price  Amount");

    /* display blank fields for each item */
    for (item = 1; item <= 15; item++)
        {
        empty(8+item, 2, 6);
        empty(8+item, 10, 6);
        empty(8+item, 45, 2);
        }

    /* restore to the previous I/O buffer */
    out_buf = old;
}

/*****************************************************************
*****************************************************************

Payment form processing

*****************************************************************
*****************************************************************/



static int
payment(payment_trans *t)
{
    int key;
    display(payment_form);
    key = payment_read(t);
    if (key != ENTER)  return key;
    payment_transaction(t);
    payment_write(t);
    return key;
}


static void
payment_setup(void)
{
    int item;
    iobuf *old;

    /* start with an empty form */
    reset(payment_form);

    /* redirect the data to a special menu buffer */
    old = out_buf;  out_buf = payment_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 38, "Payment");
    text(4, 1, "Date:");
    text(6, 1, "Warehouse:");
    number(6, 12, warehouse, 6);
    text(6, 42, "District:");
    empty(6, 52, 2);
    text(11, 1, "Customer:");
    empty(11, 11, 4);
    text(11, 17, "Cust-Warehouse:");
    empty(11, 33, 6);
    text(11, 40, "Cust-District:");
    empty(11, 54, 2);
    text(12, 1, "Name:");
    empty(12, 29, 16);
    text(12, 50, "Since:");
    text(13, 50, "Credit:");
```

```
    text(14, 50, "%Disc:");
    text(15, 50, "Phone:");
    text(17, 1, "Amount Paid:");
    empty(17, 23, 8);
    text(17, 37, "New Cust-Balance:");
    text(18, 1, "Credit Limit:");
    text(20, 1, "Cust-Data:");

    out_buf = old;
}

static int
payment_read(payment_trans *t)
{
    int i;
    int field;
    int key;

    /* Our warehouse number is fixed */
    t->W_ID = warehouse;
    t->C_ID = EMPTY_NUM;
    t->D_ID = EMPTY_NUM;
    t->C_W_ID = EMPTY_NUM;
    t->C_D_ID = EMPTY_NUM;
    t->H_AMOUNT = EMPTY_FLT;
    t->C_LAST[0] = '\0';

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 6))
        retry: switch (field)
            {

            case 1: key = read_number(6, 52, &t->D_ID, 2);
                    break;

            case 2:
                /* if last name specified, skip this field */
                if (t->C_LAST[0] != '\0')
                    break;

                /* read in the customer id */
                key = read_number(11, 11, &t->C_ID, 4);

                /* if specified, don't allow last name to be entered */
                if (t->C_ID != EMPTY_NUM)
                    {
                    blanks(12, 29, 16);
                    t->C_LAST[0] = '\0';
                    }

                /* refresh the C_LAST underlines, if possibly needed */
                else if (t->C_LAST[0] == '\0')
                    empty(12, 29, 16);
                break;

            case 3: key = read_number(11, 33,  &t->C_W_ID, 6);
                    break;

            case 4: key = read_number(11, 55, &t->C_D_ID, 2);
                    break;

            case 5:
                /* skip this field if C_ID was already specified */
                if (t->C_ID != EMPTY_NUM)
                    break;

                /* read in the customer last name */
                key = read_text(12, 29, t->C_LAST, 16);

                /* if specified, don't allow c_id to be entered */
                if (t->C_LAST[0] != '\0')
                    {
                    blanks(11, 11, 4);
                    t->C_ID = EMPTY_NUM;
                    }

                /* refresh the C_ID underlines, if possibly needed */
                else if (t->C_ID == EMPTY_NUM)
                    empty(11, 11, 4);
                break;

            case 6: key = read_money(17, 23, &t->H_AMOUNT, 8);
                    break;
            }

    /* if Aborted, then done */
    if (key != ENTER)
        return key;


    /* Make sure all the fields were entered */
    if (t->D_ID == EMPTY_NUM)
        {field=1; msgline("Please enter district id"); goto retry;}
    if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
        {field=2; msgline("C_ID or C_LAST must be entered"); goto retry;}
    if (t->C_W_ID == EMPTY_NUM)
        {field=3; msgline("Please enter customer's warehouse"); goto retry;}
    if (t->C_D_ID == EMPTY_NUM)
        {field=4; msgline("please enter customer's district"); goto retry;}
    if (t->H_AMOUNT == EMPTY_FLT)
        {field=6; msgline("Please enter payment amount"); goto retry;}
    if (t->H_AMOUNT <= 0)
                {field=6; msgline("Please enter a positive payment"); goto retry;}


    t->byname = (t->C_ID == EMPTY_NUM);
    msgline("");
    flush(out_buf);
    return key;
}


static void
payment_write(payment_trans *t)
{

    /* if errors, display a message and quit */
    if (t->status != OK)
        {
        status(24, 1, t->status);
        return;
        }

    /* display the screen */
    date(4, 7, t->H_DATE);
    text(7, 1, t->W_STREET_1);
    text(7, 42, t->D_STREET_1);
    text(8, 1, t->W_STREET_2);
    text(8, 42, t->D_STREET_2);
    text(9, 1, t->W_CITY);
    text(9, 22, t->W_STATE);
    zip(9, 25, t->W_ZIP);
    text(9, 42, t->D_CITY);
    text(9, 63, t->D_STATE);
    zip(9, 66, t->D_ZIP);
    number(11, 11, t->C_ID, 4);
    text(12, 9, t->C_FIRST);
    text(12, 26, t->C_MIDDLE);
    text(12, 29, t->C_LAST);
    date_only(12, 58, t->C_SINCE);
    text(13, 9, t->C_STREET_1);
    text(13, 58, t->C_CREDIT);
    text(14, 9, t->C_STREET_2);
    real(14, 58, t->C_DISCOUNT*100, 5, 2);  /* percentage or fraction? */
    text(15, 9, t->C_CITY);
    text(15, 30, t->C_STATE);
    zip(15, 33, t->C_ZIP);
    phone(15, 58, t->C_PHONE);
    money(17, 17, t->H_AMOUNT,14);
    money(17, 55, t->C_BALANCE, 15);
    money(18, 17, t->C_CREDIT_LIM, 14);

    /* Display cust data if bad credit. */
    if (t->C_CREDIT[0] == 'B' && t->C_CREDIT[1] == 'C')
        long_text(20, 12, t->C_DATA, 50);

    trigger2();
}



/*****************************************************************
******************************************************************

ORDSTAT form processing

******************************************************************
*****************************************************************/



static int
ordstat(ordstat_trans *t)
{
    int key;
    display(ordstat_form);
    key = ordstat_read(t);
    if (key != ENTER)  return key;
    ordstat_transaction(t);
    ordstat_write(t);
    return key;
}


static void
ordstat_setup(void)
```

```
{
    int item;
    iobuf *old;

    /* start with an empty form */
    reset(ordstat_form);

    /* redirect the data to a special menu buffer */
    old = out_buf;  out_buf = ordstat_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 35, "Order-Status");
    text(4, 1, "Warehouse:");
    number(4, 12, warehouse, 6);
    text(4, 19, "District:");
    empty(4, 29, 2);
    text(5, 1, "Customer:");
    empty(5, 11, 4);
    text(5, 18, "Name:");
    empty(5, 44, 16);
    text(6, 1, "Cust-Balance:");
    text(8, 1, "Order-Number");
    text(8, 26, "Entry-Date:");
    text(8, 60, "Carrier-Number:");
    text(9, 1, "Supply-W");
    text(9, 14, "Item-Num");
    text(9, 25, "Qty");
    text(9, 33,"Amount");
    text(9, 45, "Delivery-Date");

    /* done */
    out_buf = old;
}



static int
ordstat_read(ordstat_trans *t)
{
    int i;
    int field;
    int key;

    /* Our warehouse number is fixed */
    t->W_ID = warehouse;
    t->C_ID = EMPTY_NUM;
    t->D_ID = EMPTY_NUM;
    t->C_LAST[0] = '\0';

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 3))
        retry: switch (field)
            {

            case 1: key = read_number(4, 29, &t->D_ID, 2);
                    break;

            case 2:
                /* if last name specified, skip this field */
                if (t->C_LAST[0] != '\0')
                    break;

                /* read in the customer id */
                key = read_number(5, 11, &t->C_ID, 4);

                /* if specified, don't allow last name to be entered */
                if (t->C_ID != EMPTY_NUM)
                    {
                    blanks(5, 44, 16);
                    t->C_LAST[0] = '\0';
                    }

                /* refresh the C_LAST underlines, if possibly needed */
                else if (t->C_LAST[0] == '\0')
                    empty(5, 44, 16);
                break;

            case 3:
                /* skip this field if C_ID was already specified */
                if (t->C_ID != EMPTY_NUM)
                    break;

                /* read in the customer last name */
                key = read_text(5, 44, t->C_LAST, 16);

                /* if specified, don't allow c_id to be entered */
                if (t->C_LAST[0] != '\0')
                    {
                    blanks(5, 11, 4);
                    t->C_ID = EMPTY_NUM;
                    }

                /* refresh the C_ID underlines, if possibly needed */
                else if (t->C_ID == EMPTY_NUM)
                    empty(5, 11, 4);
                break;
            }

    /* if Aborted, then done */
    if (key != ENTER)
        return key;


    /* ensure all the necessary fields were entered */
    if (t->D_ID == EMPTY_NUM)
        {field=1; msgline("Please enter district id"); goto retry;}
    if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
        {field=2;  msgline("C_ID or C_LAST must be entered"); goto retry;}


    t->byname = (t->C_ID == EMPTY_NUM);
    msgline("");
    flush(out_buf);
    return key;
}

static void
ordstat_write(ordstat_trans *t)
{
    int i;

    /* if errors, display a status message and quit */
    if (t->status != OK)
        {
        status(24, 1, t->status);
        return;
        }

    /* display the results */
    number(5, 11, t->C_ID, 4);
    text(5, 24, t->C_FIRST);
    text(5, 41, t->C_MIDDLE);
    text(5, 44, t->C_LAST);
    money(6, 15, t->C_BALANCE, 10);
    number(8, 15, t->O_ID, 8);
    date(8, 38, t->O_ENTRY_DATE);
    if (t->O_CARRIER_ID > 0)
        number(8, 76, t->O_CARRIER_ID, 2);

    for (i=0; i< t->ol_cnt; i++)
        {
        number(i+10, 3, t->item[i].OL_SUPPLY_W_ID, 6);
        number(i+10, 14, t->item[i].OL_I_ID, 6);
        number(i+10, 25, t->item[i].OL_QUANTITY, 2);
        money(i+10, 32, t->item[i].OL_AMOUNT, 9);
        date_only(i+10, 47, t->item[i].OL_DELIVERY_DATE);
        }
    trigger2();
}




/*****************************************************************
******************************************************************

delivery form processing

******************************************************************
******************************************************************/



static int
delivery(delivery_trans *t)
{
    int key;
    display(delivery_form);
    key = delivery_read(t);
    if (key != ENTER)  return key;
    delivery_enque(t);
    delivery_write(t);
    return key;
}


static void
delivery_setup(void)
{
    int item;
    iobuf *old;
```

```
    /* start with an empty form */
    reset(delivery_form);

    /* redirect the data to a special menu buffer */
    old = out_buf;  out_buf = delivery_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 38, "Delivery");
    text(4, 1, "Warehouse:");
    number(4, 12, warehouse, 6);
    text(6, 1, "Carrier Number:");
    empty(6, 17, 2);

    /* done */
    out_buf = old;
}

static int
delivery_read(delivery_trans *t)
{
    int i;
    int field;
    int key;

    /* Our warehouse number is fixed */
    t->W_ID = warehouse;
    t->O_CARRIER_ID = EMPTY_NUM;

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 1))
        retry: switch (field)
            {
            case 1: key = read_number(6, 17, &t->O_CARRIER_ID, 2);
                    break;
            }

    /* if Aborted, then done */
    if (key != ENTER)
        return key;

    /* Must enter the carrier id */
    if ((t->O_CARRIER_ID == EMPTY_NUM) ||
                    (t->O_CARRIER_ID < 1) ||
                    (t->O_CARRIER_ID > 10))
        {field=1; msgline("Please enter a Carrier Number within 1 and 10"); goto retry; }

    /* clear the message line */
    msgline("");
    flush(out_buf);
    return key;
}

static void
delivery_write(delivery_trans *t)
{
    if (t->status == OK) {
        text(8, 1, "Execution Status: Delivery has been queued");
                    trigger2();
    } else
        status(8, 1, t->status);
}



/*****************************************************************
******************************************************************

stocklev form processing

******************************************************************
*****************************************************************/


static int
stocklev(stocklev_trans *t)
{
    int key;
    display(stocklev_form);
    key = stocklev_read(t);
    if (key != ENTER)  return key;
    stocklev_transaction(t);
    stocklev_write(t);
    return key;
}


static void
stocklev_setup(void)
{
    int item;
    iobuf *old;

    /* start with an empty form */
    reset(stocklev_form);

    /* redirect the data to a special menu buffer */
    old = out_buf;  out_buf = stocklev_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 35, "Stock-Level");
    text(4, 1, "Warehouse:");
    number(4, 12, warehouse, 6);
    text(4, 19, "District:");
    number(4, 29, district, 2);
    text(6, 1, "Stock Level Threshold:");
    empty(6, 24, 2);
    text(8, 1, "low stock");

    /* done */
    out_buf = old;
}



static int
stocklev_read(stocklev_trans *t)
{
    int field;
    int key;

    t->W_ID = warehouse;
    t->D_ID = district;
    t->threshold = EMPTY_NUM;

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 1))
        retry: switch (field)
            {
            case 1: key = read_number(6, 24, &t->threshold, 2);
                    break;
            }

    /* if Aborted, then done */
    if (key != ENTER)
        return key;


    /* make sure the necessary fields were entered */
    if ((t->threshold == EMPTY_NUM) ||
        (t->threshold < 10) ||
                    (t->threshold > 20))

        {field=1; msgline("Please enter a threshold within 10 and 20"); goto retry; }

    /* clear the message line */
    msgline("");
    flush(out_buf);
    return key;
}

static void
stocklev_write(stocklev_trans *t)
{
    if (t->status == OK) {
        number(8, 12, t->low_stock, 3);
                    trigger2();
    } else
        status(10, 1, t->status);
}


/*****************************************************************
******************************************************************

login form processing

******************************************************************
*****************************************************************/

static int
login(void)
{
    int field;
    int key;
    char auditstr[21];
    int w_id, d_id;

    /* assume the default values */
    w_id = warehouse;
```

```
    d_id = district;
    auditstr[0] = '\0';

    /* display the login menu */
    position(1,1);  clear_screen();
    text(3, 30, "Please login.");
    text(5,5,"Warehouse:");
    number(5, 16, w_id, 6);
    text(5, 24, "District:");
    number(5, 34, d_id, 2);
    text(15, 5, "Audit String:");
    text(15, 19, CLIENT_AUDIT_STRING);
    empty(16, 19, 20);

    /* Get values until done */
    for (field = 1; field > 0;  field = next_field(field, key, 3))
        retry: switch (field)
            {
            case 1:
                key = read_number(5, 16, &w_id, 6);
                break;

            case 2:
                key = read_number(5, 34, &d_id, 2);
                break;

            case 3:
                key = read_text(16, 19, auditstr, 20);
                break;
            }

    if (key != ENTER)
        return EOF;

    if (w_id == EMPTY_NUM && warehouse == EMPTY_NUM)
        {
        msgline("You must enter a warehouse id");
        field =1;
        goto retry;
        }

    if (d_id == EMPTY_NUM && district == EMPTY_NUM)
        {
        msgline("You must enter a district id");
        field = 2;
        goto retry;
        }

    if (w_id != EMPTY_NUM)
        warehouse = w_id;
    if (d_id != EMPTY_NUM)
        district = d_id;

    /* done */
#if 0
    flush(out_buf);
#endif
    return key;
}




/*****************************************************************
******************************************************************

menu form processing

******************************************************************
*****************************************************************/

static void
menu_setup(void)
{

    /* display the menu on the iobuf -- never erased */
    position(1, 1);
    clear_screen();
    string("(1)New-Order (2)Payment (3)Order-Status ");
    string("(4)Delivery (5)StockLevel (9)Exit");
}

static int
menu_read(void)
{
    position(1, 1);
    trigger();
    return getkey();
}
```

```
static int
next_field(int current, int key, int max)
{

    if (key == BACKTAB)
        if (current == 1)    return max;
        else                 return current-1;
    else if (key == TAB)
        if (current == max)  return 1;
        else                 return current+1;
    else
        return 0;
}

static void
msgline(char *str)
{
    position(24, 1);
    clear_screen();
    string(str);
#if 0
    flush(out_buf);  /* Needed? */
#endif
}

static int
setup(int fd)
{
    int key;

    /* Initialize the forms */
    init_iobuf(neworder_form, NEWORDER_FORM_SIZE, fd, fd);
    init_iobuf(payment_form,  PAYMENT_FORM_SIZE, fd, fd);
    init_iobuf(ordstat_form,  ORDSTAT_FORM_SIZE, fd, fd);
    init_iobuf(delivery_form, DELIVERY_FORM_SIZE, fd, fd);
    init_iobuf(stocklev_form, STOCKLEV_FORM_SIZE, fd, fd);

    /* Initialize input/output */
    init_iobuf(output_stuff, OUTPUT_BUF_SIZE, fd, fd);
    init_iobuf(input_stuff,  INPUT_BUF_SIZE, fd, fd);

    /* Setup Input and Output buffers */
    in_buf  = input_stuff;
    out_buf = output_stuff;

    /* get the user, warehouse and district numbers */
    warehouse = EMPTY_NUM;
    district  = EMPTY_NUM;
    key       = login();
    user      = warehouse*DIST_PER_WARE + district + 1;

    /* set up the forms */
    menu_setup();
    neworder_setup();
    payment_setup();
    ordstat_setup();
    delivery_setup();
    stocklev_setup();

    /* connect to the delivery queue */
    delivery_init(user);

    return key;
}

static void
cleanup(void)
{

    /* detach from the delivery queue */
    delivery_done();

    /* clear the screen */
    position(1, 1);
    clear_screen();
    trigger();
    flush(out_buf);
}


/*****************************************************************
******************************************************************

Screen Output Routines

******************************************************************
*****************************************************************/


static void
number(int row, int col, int  n, int width)
{
    char str[81];
    fmt_num(str, n, width);
    text(row, col, str);
```

```
}


static void
real(int row, int col, double x, int width, int dec)
{
   char str[81];
   fmt_flt(str, x, width, dec);
   text(row, col, str);
}

static void
date(int row, int col, char *date_str)
{
   text(row, col, date_str);
}

static void
date_only(int row, int col, char *date_str)
{
   date_str[10] = '\0';
   text(row, col, date_str);
}

static void
money(int row, int col, double x, int width)
{
   char str[81];
   fmt_money(str, x, width);
   text(row, col, str);
}


static void
long_text(int row, int col, char *str, int width)
{
   int pos;

   /* repeat until the entire string is written out */
   for (pos = width; *str != '\0';  str++, pos++)
      {

      /* if at end of line, position the cursor to next line */
      if (pos >= width)
         {
         position(row, col);
         pos = 0;
         row++;
         }

      /* output the next character */
      pushc(out_buf,*str);
      }
}

static void
text(int row, int col, char str[])
{
   position(row, col);
   string(str);
}


static void
phone(int row, int col,char *str)
{
      char temp[30];

      fmt_phone(temp,str);
      text(row,col,temp);
}


static void
zip(int row, int col, char *str)
{
   char temp[30];

   fmt_zip(temp,str);
   text(row,col,temp);
}

static void
empty(int row, int col, int len)
{
   position(row, col);
   while (len-- > 0)
      pushc(out_buf,'_');
}

static void
blanks(int row, int col, int len)
{
   position(row, col);
   while (len-- > 0)
      pushc(out_buf,' ');
}


static void
status(int row, int col, int status)
/*********************************************************************
status displays the transaction status
  Note: must correspond to 'get_status' in driver/keystroke.c
*********************************************************************/
{
  text(row, col, "Execution Status: ");

  if (status == OK)
     string("Transaction Committed");
  else if (status == E_INVALID_ITEM)
     string("Item number is not valid");
  /* Do the rev. 3.3 error checking here.       */
  else if (status == E_INVALID_INPUT)
     string("Invalid input, transaction not executed");
  else
     {
     string("Rollback -- ");
     number(row, col+30, status, 5);
     }
  trigger2();
}

/*********************************************************************
*********************************************************************

ASCII terminal control

*********************************************************************
*********************************************************************/


static void
trigger(void)
/*********************************************************************
trigger sends a turnaround sequence to let the driver know to send input
*********************************************************************/
{
   pushc(out_buf,TRIGGER);
}

static void
trigger2(void)
/*********************************************************************
trigger2 sends another turnaround sequence to let the driver know what
is going on.
*********************************************************************/
{
   pushc(out_buf,TRIGGER2);
}

static void
position(int row, int col)
/*********************************************************************
position positions the cursor at the given row and column
*********************************************************************/
{
   pushc(out_buf,ESCAPE);
   pushc(out_buf,'[');
   if (row >= 10)
      pushc(out_buf,'0' + row/10);
   pushc(out_buf,'0'+ row%10);
   pushc(out_buf,';');
   if (col >= 10)
      pushc(out_buf,'0' + col/10);
   pushc(out_buf,'0' + col%10);
   pushc(out_buf,'H');
}

static void
clear_screen(void)
/*********************************************************************
clear_screen clears the iobuf from cursor position to end of iobuf
*********************************************************************/
{
   pushc(out_buf,ESCAPE);
   pushc(out_buf,'[');
   pushc(out_buf,'J');
}

/*********************************************************************
*********************************************************************

Screen Input Routines

*********************************************************************
*********************************************************************/
#define funny(key)  (key != ENTER && key !=TAB && key != BACKTAB)
```

```
static int
read_number(int row, int col, int *n, int width)
/*********************************************************************
read_number reads an integer field
*********************************************************************/
{
   char temp[81];
   int key;
   int err;
   debug("read_number: row=%d  col=%d  width=%d n=%d \n",row, col,width,*n);

   /* generate the current characters */
   fmt_num(temp, *n, width);
   err = NO;

   /* repeat until a valid number or a funny key is pressed */
   for (;;)
      {
      /* Let the user edit the field */
      key = getfield(row, col, temp, width, Num);
                  if (funny(key)) return key;

      /* convert the field to a number */
      *n = cvt_num(temp);
      if (*n != INVALID_NUM) break;

      msgline("Invalid digit entered");
                  pushc(out_buf,BELL);
      err = YES;
      }

   /* display the new number */
   number(row, col, *n, width);
   if (err) msgline("");
   debug("read_number: n=%d  key=%d\n", *n, key);
   return key;
}


static int
read_money(int row, int col, double *m, int width)
{
   char temp[81];
   int key;
   int err;

   err = NO;
   fmt_money(temp, *m, width);

   /* repeat until a valid number or a funny key is pressed */
   for (;;)
      {
      key = getfield(row, col, temp, width, Money);
                  if (funny(key)) return key;

      *m = cvt_money(temp);
      if (*m != INVALID_FLT)  break;

      msgline("Please enter amount $99999.99");
      pushc(out_buf,BELL);
      err = YES;
                  }

   money(row, col, *m, width);
   if (err) msgline("");
   return key;
}

static int
read_text(int row, int col, char *s, int width)
{
   char temp[81];
   int key;
   int i;

   /* generate the current characters */
   fmt_text(temp, s, width);

   /* let the user edit the field */
   key = getfield(row, col, temp, width, Text);
   if (funny(key)) return key;

   /* Strip off leading and trailing space characters */
   cvt_text(temp, s);

   /* redisplay the current text */
   fmt_text(temp, s, width);
   text(row, col, temp);

   return key;
}

static int
getfield(int row, int col, char buf[], int width, FIELD_TYPE ftype)
{
   int pos, key;

   debug("getfield: width=%d  buf=%*s\n", width, width, buf);

   /* go to the beginning of the field */
   position(row, col);
   trigger();
   pos = 0;

   /* repeat until a special control character is pressed */
   for (;;)
      {

      /* get the next character */
      key = getkey();

      /* CASE: Add to buf if it fits and Is it a valid character ? */
      if (pos < width  && valid_char(key, ftype))
         {
         buf[pos] = key;
         pos++;
         pushc(out_buf,key);
         }

      /* CASE: char is BACKSPACE.  Erase last character. */
      else if (key == BACKSPACE && pos > 0)
         {
         pos--;
         buf[pos] = '_';
         pushc(out_buf,BACKSPACE);
         pushc(out_buf,'_');
         pushc(out_buf,BACKSPACE);
         }

      /* CASE: enter, tab, backtab, ^c.  Exit loop */
      else if (key==ENTER || key==TAB || key==BACKTAB || key==CNTRLC
            || key == EOF)
         break;

      else if (key=='\031')   /* for debugging, let ^X == ENTER */
         {key=ENTER; break;}

      /* Otherwise, ignore the character and beep */
      else
         pushc(out_buf,BELL);
      }

   debug("getfield: final key: %d  buf=%*s\n", key, width, buf);
   return key;
}

static int
valid_char(int key, FIELD_TYPE ftype)
/*********************************************************************
valid_char is true if the key is valid for this type of field
*********************************************************************/
{
   int valid;
   switch(ftype)
                  {
                  case Num : valid = (isdigit(key) || key == '-' || key == ' ');
                              break;

      case Text : valid = (isprint(key) || key == ' ');
                              break;

      case Money : valid = (isdigit(key) || key == '-' || key == '.'
                                          || key == '$' || key == ' ');
                        break;

      default     : valid = NO;
                              break;
      }

    return valid;
}

static pthread_t
spawn_user(int c_fd, long tc)
{
  int pid;
  int ret;
  pthread_t t;
  thread_data *td;

  td = (thread_data *)malloc(sizeof(thread_data));
  if (td == NULL) {
     syserror("Can't create thread argument data\n");
  }
  td->fd          = c_fd;
  td->tux_context = tc;
  ret = pthread_create(&t, NULL, client_main, (void *)td);
  if (ret != 0) {
     syserror("Can't create client thread\n");
```

```
    }
  return t;
}

int
connect_client(int server_fd)
/*********************************************************************
connect_client connects the clients who are waiting
*********************************************************************/
{
   int fd, vfd;
   struct sockaddr dummy_addr;
   int dummy_size = sizeof(dummy_addr);

   /* accept a connection to a new client.  Exit if no more */
   fd = accept(server_fd, &dummy_addr, &dummy_size);
   if (fd < 0)
      syserror("Can't accept new client\n");

   /* set the socket parameters */
   if (prepare_socket(fd) < 0)
                syserror("Can't set socket parameters\n");

   return fd;
}

int
server_socket(int port)
/*********************************************************************
server_socket creates a socket for a server with the given name
*********************************************************************/
{
   int fd;
   struct sockaddr_in address;

   /* create a socket */
   fd = socket(AF_INET, SOCK_STREAM, 0);
   if (fd < 0)
                syserror("Can't create a socket\n");
   if (prepare_socket(fd) < 0)
                syserror("Can't configure the socket\n");

   /* build up an internet style address */
   address.sin_family = AF_INET;
   address.sin_port = htons(port);
   address.sin_addr.s_addr = INADDR_ANY;

   /* set up the socket to listen at the given address */
   if (bind(fd, &address, sizeof(address)) < 0)
      syserror("Can't bind the socket to address\n");
   if (listen(fd, SOMAXCONN) < 0)
      syserror("Can't listen\n");

   return fd;
}

static void
Usage(char *programName)
{
   printf("usage: %s [[-s <num_of_servers>] port_number]\n",programName);
}

static void
GetArgs(int argc, char **argv)
{
   extern char *optarg;
   extern int optind;
   char *programName;
   char c;

   programName = argv[0];
   while((c = getopt(argc, argv, "s:")) != EOF)  {
                switch (c) {
                   case 's':
                                number_of_servers = atoi(optarg);
                                if (number_of_servers <= 0) number_of_servers = 1;
                                break;
        default:
                                Usage(programName);
                                exit(1);
     }
    }
    if (optind < argc) {
                if ((argc - optind) == 1) {
       port_number = atoi(argv[optind]);
     }
    }
}


int
main(int argc, char **argv)
{
   int server_fd;
   int client_fd;
   int i;
   int pid;
   long tux_context;       /* Tuxedo context to use */

   /* We don't want zombie children */
   signal(SIGCHLD, SIG_IGN);

   /* Ignore SIGPIPE, since they occur normally */
   signal(SIGPIPE, SIG_IGN);

   if (rtprio(0, 80) < 0) {
      perror("Server can't run real-time");
   }

   GetArgs(argc, argv);

   /* create a socket to accept new requests */
   server_fd = server_socket(port_number);
   if (server_fd < 0)  {
      syserror("Can't create a listening socket\n");
   }

   /* Create more servers if requested */
   for(i = 0; i < (number_of_servers-1); i++) {
                if ((pid = fork()) == -1) {
                                syserror("Could not fork a new helper process\n");
                } else if (pid == 0) {
                                /* Child */
                                break;
                } else {
                                /* Parent */
                }
   }

   /* repeat forever in each child */
   while (user_connections < MAX_USERS_PER_PROCESS) {
      client_fd = connect_client(server_fd);
                if ((user_connections % MAX_THREADS_PER_CONTEXT) == 0 ) {
        /* connect to the transaction processor */
                   tux_context = transaction_begin();
                }
                user_ids[user_connections] = spawn_user(client_fd, tux_context);
                user_connections++;
   }

   /* Close listening socket */
   close(server_fd);

   for(i = 0; i < user_connections; i++) {
                if (pthread_join(user_ids[i], NULL) != 0) {
                   message("Pthread message, error = %d, thread_id = %d, id = %d\n",
                                                errno, user_ids[i], i);
                                syserror("Pthread_join error\n");
                }
   }

   /* detach from transaction engine */
   transaction_done();

   return 0;
}

#define popc(b)   (*(b)->cur++)

static void
string(char str[])
{
   for (; *str != '\0'; str++)
      pushc(out_buf,*str);
}

static void
display(iobuf *scr)
{
/* Note: if problems doing output, let the input routine detect it */
   char *p;
   int len;
   for (p = scr->beg; p < scr->end; p+=len) {
      len = write(scr->ofd, p, scr->end - p);
      if (len <= 0) break;
   }
}


static void
input(iobuf *scr)
{
   int len;

   /* read in as many characters as are available */
   len = read(scr->ifd, scr->end, scr->max - scr->end);

   /* if end of input, then pretend we read an END character */
   if (len == 0 || (len == -1 && errno == ECONNRESET)) {
      *scr->end = EOF;
```

```
        len = 1;
    }

    /* Check for errors */
    else if (len == -1)
        syserror("input(scr): unable to read stdin\n");

    /* update the pointers to reflect the new data */
    scr->end += len;
    *scr->end='\0';  /* for debugging */
}

static int
getkey(void) {
    if (in_buf->cur == in_buf->end) {
        flush(out_buf);
        reset(in_buf);
        input(in_buf);
    }

    return popc(in_buf);
}
```

client/tux_transaction.c

```
/*********************************************************************************
 @(#) Version: A.10.10 $Date: 2002/07/18 22:26:19 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*********************************************************************************/

#include <varargs.h>
#include <errno.h>

#include "tpcc.h"
#include "atmi.h"
#include "Uunix.h"

#define MYMAX(a, b)  (a > b) ? a : b

__thread void *data_ptr;

static
tux_error(format, va_alist)
char *format;
va_dcl
{
    va_list argptr;

    va_start(argptr);
    vmessage(format, argptr);

    message("Tuxedo error %d\n", tperrno);

    errno = Uunixerr;
    if (tperrno == TPEOS) {
        syserror("Tuxedo encountered O/S error\n");
    }

    if (tperrno == TPESVCERR || tperrno == TPETIME)      {
                    message("Retrying transaction\n");
                    if (tperrno == TPETIME)
                        sleep (1);
    } else {
                    error("EXITING !!!\n");
    }

}


TPCONTEXT_T
transaction_begin()
{
    static TPINIT *initialization_buffer = NULL;
    TPCONTEXT_T ctx = NULL;

    /* Create buffer needed to indicate MultiContexts operation */
    if (initialization_buffer == NULL) {
                    initialization_buffer = (TPINIT *)tpalloc("TPINIT", NULL,
                                    TPINITNEED(0));
        if (initialization_buffer == NULL) {
            tux_error("Unable to allocate Tuxedo TPINIT memory\n");
        }
                    initialization_buffer->flags = TPMULTICONTEXTS;
    }

    /* attach to Tuxedo */
    if (tpinit(initialization_buffer) == -1) {
            tux_error("Failed to attach to Tuxedo\n");
    }

    /* get the context */
    if (tpgetctxt(&ctx, 0) == -1) {
                    tux_error("Failed to get Tuxedo context\n");
    }
    return ctx;
}

void
thread_transaction_begin(TPCONTEXT_T ctx)
{
    unsigned long alloc_size;

    if (tpsetctxt(ctx, 0) == -1) {
                    tux_error("Could not set Tuxedo context\n");
    }

    /* allocate structures for each transaction */
    alloc_size = MYMAX(sizeof(neworder_trans), sizeof(payment_trans));
    alloc_size = MYMAX(alloc_size, sizeof(ordstat_trans));
    alloc_size = MYMAX(alloc_size, sizeof(stocklev_trans));
    alloc_size = MYMAX(alloc_size, sizeof(delivery_trans));
    data_ptr = (void *)tpalloc("CARRAY", NULL, alloc_size);

    if (data_ptr == NULL) {
        tux_error("Unable to allocate Tuxedo memory\n");
    }
}

void
transaction_done(void)
{
    if (tpterm() == -1) {
        tux_error("Unable to detach from Tuxedo\n");
    }
}

void
neworder_transaction(neworder_trans *t)
{
    long result;
    *((neworder_trans *)data_ptr) = *t;
    while (tpcall("NEWO_SVC", (char *)data_ptr, sizeof(neworder_trans),
            (char **)(&data_ptr), &result, TPSIGRSTRT|TPNOTIME) == -1) {
        tux_error("Tuxedo failed for neworder transaction\n");
                    *((neworder_trans *)data_ptr) = *t;
    }
    *t = *((neworder_trans *)data_ptr);
}

void
payment_transaction(payment_trans *t)
{
    long result;
    *((payment_trans *)data_ptr) = *t;
    while (tpcall("PMT_SVC", (char *)data_ptr, sizeof(payment_trans),
                (char **)(&data_ptr), &result, TPSIGRSTRT|TPNOTIME) == -1) {
        tux_error("Tuxedo failed for payment transaction\n");
                    *((payment_trans *)data_ptr) = *t;
    }
    *t = *((payment_trans *)data_ptr);
}


void
ordstat_transaction(ordstat_trans *t)
{
    long result;
    *((ordstat_trans *)data_ptr) = *t;
    while (tpcall("ORDS_SVC", (char *)data_ptr, sizeof(ordstat_trans),
                (char **)(&data_ptr), &result, TPSIGRSTRT|TPNOTIME) == -1) {
                    tux_error("Tuxedo failed for ordstat transaction\n");
                    *((ordstat_trans *)data_ptr) = *t;
    }
    *t = *((ordstat_trans *)data_ptr);
}

void
stocklev_transaction(stocklev_trans *t)
{
    long result;
    *((stocklev_trans *)data_ptr) = *t;
    while (tpcall("STKL_SVC", (char *)data_ptr, sizeof(stocklev_trans),
                (char **)(&data_ptr), &result, TPSIGRSTRT|TPNOTIME) == -1) {
        tux_error("Tuxedo failed for stocklev transaction\n");
                    *((stocklev_trans *)data_ptr) = *t;
    }
    *t = *((stocklev_trans *)data_ptr);
}


void
delivery_init(int u)
{
}
```

```
void
delivery_enque(delivery_trans *t)
{
   gettimeofday(&t->enque[0], NULL);
   t->status = OK;

   *((delivery_trans *)data_ptr) = *t;
   while (tpacall("DVRY_SVC", (char *)data_ptr, sizeof(delivery_trans),
            TPNOREPLY) == -1) {
      tux_error("Tuxedo failed enqueing delivery transaction\n");
                  *((delivery_trans *)data_ptr) = *t;
   }
}

void
delivery_done(void)
{
}
```

client/Makefile

```
#*******************************************************************************
*
#@(#) Version: A.10.10 $Date: 2002/08/06 12:02:41 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************
*

#
# Makefile for compiling the client, batch-tpcc, and service code
#

P      = ${WORK_DIR}/src
I      = $(P)/lib
L      = $(P)/lib
D      = $(P)/driver
Q      = $(P)/que
S      = $(P)/client

MV=cp -r

include ../buildenv.mk
# include $(ORACLE_HOME)/rdbms/lib/env_rdbms.mk
LIBDIR=lib32

SH_OPT     = -Wl,-a,shared
OPT        = -Wl,-E
LDOPTS     = -E -ldld -a archive_shared

# Check tkvcinin.sql to set ORA_NO_NULL_DATE
# Check tpccload     to set ORA_LOAD_NULL_DATE

TUX_INCLUDE= -I${ROOTDIR}/include

COMMON_FLAGS =$(OPTIMIZE) $(ARCHFLAGS) $(CINCLUDES)
SH_CFLAGS  = $(COMMON_FLAGS) ${SH_OPT} ${TUX_INCLUDE}
CFLAGS           = $(COMMON_FLAGS) ${OPT} ${TUX_INCLUDE}
CFLAGS_ORA = $(COMMON_FLAGS) ${INCLUDE} ${OPT} ${TUX_INCLUDE}
CFLAGS_ORA_BATCH = $(COMMON_FLAGS) ${INCLUDE} ${OPT}


# LDFLAGS=$(LDFLAGS32)
# LDFLAGS_ORA = $(CFLAGS_ORA) ${L}/tpc_lib.a -L${LIBHOME32}
LDFLAGS_ORA_BATCH = $(CFLAGS_ORA_BATCH) ${L}/tpc_lib.a -L${LIBHOME32}
# LDFLAGS_ORA = $(CFLAGS_ORA) ${L}/tpc_lib.a -L${LIBHOME}
# LDFLAGS_ORA_BATCH = $(CFLAGS_ORA_BATCH) ${L}/tpc_lib.a -L${LIBHOME}

LDFLAGS_ORA=-Wl,+n -L$(ORACLE_HOME)/rdbms/lib32/ -L$(ORACLE_HOME)/lib32/ -
dynamic\
-L$(ORACLE_HOME)/rdbms/lib32/ -L$(ORACLE_HOME)/lib32/ \
$(ORACLE_HOME)/rdbms/lib32/defopt.o -lclntst9 -lnsslb9 -lnl9 -ln9\
`cat $(ORACLE_HOME)/lib32/sysliblist` \
-L$(ORACLE_HOME)/lib32 -lm
LDFLAGS_ORA_BATCH = $(LDFLAGS_ORA)


I_SYM=-I

INCLUDE=$(I_SYM). $(I_SYM)$(ORACLE_HOME)/rdbms/demo \
$(I_SYM)$(ORACLE_HOME)/rdbms/public \
$(I_SYM)$(ORACLE_HOME)/rdbms/include \
$(I_SYM)$(ORACLE_HOME)/plsql/public \
$(I_SYM)$(ORACLE_HOME)/network/public \
$(I_SYM)$(DPB_LIB_DIR)


PROGRAMS     = client service client_batch msg_server raw

all: ${PROGRAMS}

all_ora: others_oracle service_oracle tpcc_client
tpcc_client: client
                $(MV) client ${WORK_DIR}/bin/
others_oracle: raw client_batch_ora msg_server_ora
                $(MV) raw client_batch msg_server ${WORK_DIR}/bin
service_oracle: service_ora
                $(MV) service_ora ${WORK_DIR}/bin/service

${S}/oracle/transaction.o: ${S}/oracle/transaction.c
                $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/transaction.c;

ORA_OBJS=plnew.o plord.o plpay.o pldel.o plsto.o tpccpl.o
plnew.o: ${S}/oracle/plnew.c
                $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/plnew.c;
plord.o: ${S}/oracle/plord.c
                $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/plord.c;
plpay.o: ${S}/oracle/plpay.c
                $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/plpay.c;
pldel.o: ${S}/oracle/pldel.c
                $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/pldel.c;
plsto.o: ${S}/oracle/plsto.c
                $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/plsto.c;
tpccpl.o: ${S}/oracle/tpccpl.c
                $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/tpccpl.c;

raw: raw.o
                cc ${CFLAGS} raw.o $(L)/tpc_lib.a -o raw

client.o: client.c
                cc ${CFLAGS} -D_REENTRANT -c client.c

tux_transaction.o: tux_transaction.c
                cc ${CFLAGS} -D_REENTRANT -c tux_transaction.c

client: client.o  tux_transaction.o
                ${ROOTDIR}/bin/buildclient -v -o client \
        -f "$(COMMON_FLAGS) -D_REENTRANT $(OPT) -Wl,+pi 256K -Wl,+pd 4K -Wl,-R
0x100000 \
                client.o tux_transaction.o $(L)/tpc_lib.a" \
                -l "-lnsl  -lm -Wl,-ashared -lc"

service_ora: service.o ${S}/oracle/transaction.o $(ORA_OBJS) $(L)/tpc_lib.a
                ${ROOTDIR}/bin/buildserver -v -b shm \
                -s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC \
                -o service_ora \
                -f "-Wl,+pi 4M -Wl,+pd 256K \
                   -Wl,-R 0x400000 -Wl,-D 0x40100000 \
                                        service.o transaction.o $(ORA_OBJS) $L/tpc_lib.a \
                                ${LDFLAGS_ORA} $(DEF_OPT) $(TTLIBS)" \
                -l "-lnsl  -lclntst9"

client_batch_ora: $(D)/driver.o $(D)/generate.o ${S}/oracle/transaction.o \
                                $(ORA_OBJS) $(Q)/dummy_que.o $(L)/tpc_lib.a  \
                                $(L)/server_default.o
                        $(CC) $(D)/driver.o $(D)/generate.o transaction.o $(ORA_OBJS) \
                          $(Q)/dummy_que.o $(L)/server_default.o $(L)/tpc_lib.a \
                          ${LDFLAGS_ORA_BATCH} $(DEF_OPT) $(TTLIBS) -lclntst9 -o
client_batch;

msg_server_ora: $(Q)/msg_server.o ${S}/oracle/transaction.o $(ORA_OBJS) \
                                $(L)/tpc_lib.a
                        $(CC) $(Q)/msg_server.o transaction.o $(ORA_OBJS) ${LDFLAGS_ORA} \
                         $(L)/tpc_lib.a $(DEF_OPT) $(TTLIBS) -lclntst9 -o msg_server;

clean:
                rm -f *.o

clobber: clean
                rm -f ${PROGRAMS}

install: ${PROGRAMS}
                cp ${PROGRAMS} ${WORK_DIR}/bin
```

A.2 Tpc_lib Source

lib/tpcc.h

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 2002/07/18 22:07:51 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.

  History
  @022801 ML  Added Client Substitution Report for TPC-C TAB ID 334.

*******************************************************************************/
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED
```

```
#include <values.h>


/* The auditor can define these 20 char strings to be anything */
#define DRIVER_AUDIT_STRING  "driver audit string"
#define CLIENT_AUDIT_STRING  "client audit string"

#ifdef DEBUG
#define debug printf
#else
#define debug (void)
#endif

#include <stdio.h>

typedef int ID;          /* All id's */
typedef double MONEY;    /* Large integer number of cents */
typedef char TEXT;       /* Add an extra byte for null terminator */
typedef double TIME;     /* Elapsed seconds from start of run (float?) */
typedef int COUNT;       /* integer numbers of things */
typedef double REAL;     /* real numbers */
typedef int LOGICAL;     /* YES or NO */
typedef struct {         /* days and seconds since Jan 1, 1900 */
   int day;              /* NULL represented by negative day */
   int sec;
   } DATE;

/* Macro to convert time of day to TIME */
#include <time.h>
extern struct timeval start_time;
#define elapsed_time(t) ( ((t)->tv_sec - start_time.tv_sec) + \
                  ((t)->tv_usec - start_time.tv_usec) / 1000000.0 )

typedef enum {Num,Money,Text,Time,Real,Date} FIELD_TYPE;   /* screen field types */


/* Various TPCC constants */
#define W_ID_LEN       4
#define D_ID_LEN       2
#define C_ID_LEN       4
#define I_ID_LEN       6
#define OL_QTY_LEN     2
#define PMT_LEN        7
#define C_ID_LEN       4
#define C_LAST_LEN     16
#define CARRIER_LEN    2
#define THRESHOLD_LEN  2
#define DIST_PER_WARE  10
#define CUST_PER_DIST  3000
#define ORD_PER_DIST   3000
#define MAXITEMS       100000
#define MAX_DIGITS     3      /* # of digits of the NURand number selected
                                 to generate the customer last name */
#define MAXWAREHOUSE   2000   /* maximum # of warehouses - scaling factor */
#define LOADSEED       42 /* # of digits of the NURand number selected

/*************************************************************************/
/* database identifiers and populations                                  */
/*************************************************************************/

#define no_item MAXITEMS                  /*  100000          */
#define no_dist_pw DIST_PER_WARE
#define no_cust_pd  CUST_PER_DIST         /*   3000           */
#define no_ord_pd ORD_PER_DIST            /*   3000           */

/* fields to add to each transaction for acid testing */
#define ACID_STUFF    \
   char acid_txn[2]; \
   int  acid_timing; \
   int  acid_action; \
   FILE *acid_res

typedef struct {
   ID OL_SUPPLY_W_ID;
   ID OL_I_ID;
   TEXT I_NAME[24+1];
   COUNT OL_QUANTITY;
   COUNT S_QUANTITY;
   MONEY I_PRICE;
   char brand_generic;
   } neworder_item;

typedef struct {
   int status;
   LOGICAL all_local;
   ID W_ID;
   ID D_ID;
   ID C_ID;
   TEXT C_LAST[C_LAST_LEN+1];
   TEXT C_CREDIT[2+1];
   REAL C_DISCOUNT;
   COUNT O_OL_CNT;
   ID O_ID;
   TEXT O_ENTRY_D[20];  /* dates as text fields */
   REAL W_TAX;
   REAL D_TAX;
   neworder_item item[15];
   ACID_STUFF;
   } neworder_trans;


typedef struct {
   int status;
   LOGICAL byname;
   ID W_ID;
   ID D_ID;
   ID C_ID;
   ID C_D_ID;
   ID C_W_ID;
   MONEY H_AMOUNT;
   TEXT H_DATE[20];  /* date as text field */
   TEXT W_STREET_1[20+1];
   TEXT W_STREET_2[20+1];
   TEXT W_CITY[20+1];
   TEXT W_STATE[2+1];
   TEXT W_ZIP[9+1];
   TEXT D_STREET_1[20+1];
   TEXT D_STREET_2[20+1];
   TEXT D_CITY[20+1];
   TEXT D_STATE[2+1];
   TEXT D_ZIP[9+1];
   TEXT C_FIRST[16+1];
   TEXT C_MIDDLE[2+1];
   TEXT C_LAST[16+1];
   TEXT C_STREET_1[20+1];
   TEXT C_STREET_2[20+1];
   TEXT C_CITY[20+1];
   TEXT C_STATE[2+1];
   TEXT C_ZIP[9+1];
   TEXT C_PHONE[16+1];
   TEXT C_SINCE[20]; /* date as text field */
   TEXT C_CREDIT[2+1];
   MONEY C_CREDIT_LIM;
   REAL C_DISCOUNT;
   REAL C_BALANCE;
   TEXT  C_DATA[200+1];
   ACID_STUFF;
   } payment_trans;


typedef struct {
   int status;
   LOGICAL byname;
   ID W_ID;
   ID D_ID;
   ID C_ID;
   TEXT C_FIRST[16+1];
   TEXT C_MIDDLE[2+1];
   TEXT C_LAST[16+1];
   MONEY C_BALANCE;
   ID O_ID;
   TEXT O_ENTRY_DATE[20];   /* date as text field */
   ID O_CARRIER_ID;
   COUNT ol_cnt;
   struct {
      ID OL_SUPPLY_W_ID;
      ID OL_I_ID;
      COUNT OL_QUANTITY;
      MONEY OL_AMOUNT;
      TEXT OL_DELIVERY_DATE[20]; /* date as text field */
      } item[15];
   ACID_STUFF;
   } ordstat_trans;


typedef struct {
   int status;
   ID W_ID;
   ID D_ID;
   COUNT threshold;
   COUNT low_stock;
   ACID_STUFF;
   } stocklev_trans;

typedef struct {
   int status;
   ID W_ID;
   ID O_CARRIER_ID;
   struct {
      ID O_ID;
                  int status;
      } order[10];
   struct timeval enque[1];
   struct timeval deque[1];
   struct timeval complete[1];
   ACID_STUFF;
   } delivery_trans;
```

```
typedef union {
    neworder_trans neworder;
    payment_trans payment;
    ordstat_trans ordstat;
    delivery_trans delivery;
    stocklev_trans stocklev;
    int status;
    } generic_trans;


/*******************************************************
Record formats for results
*******************************************************/

#ifdef NOTYET
typedef struct
    {
    float t1, t2, t3, t4, t5;
    int status          :8;
    unsigned int type    :3;
    unsigned int ol_cnt  :4;
    unsigned int remote_ol_cnt :4;
    unsigned int byname :1;
    unsigned int remote  :1;
    unsigned int skipped  :4;
    int clnt_no;                            /* @022801 ML */
    int userid;                             /* @022801 ML */
    } success_t;
#endif

typedef struct
    {
    TIME t1, t2, t3, t4, t5;
    int status;
    unsigned int type    :3;
    unsigned int ol_cnt  :4;
    unsigned int remote_ol_cnt :4;
    unsigned int byname :1;
    unsigned int remote  :1;
    unsigned int skipped  :4;
    int clnt_no;                            /* @022801 ML */
    int userid;                             /* @022801 ML */
    } success_t;

typedef struct
    {
    struct timeval start_time;
    } success_header_t;


/*************************************************************************
Record formats for loading routines.  (DB's have own internal formats
*************************************************************************/
typedef struct
    {
    ID W_ID;
    TEXT W_NAME[10+1];
    TEXT W_STREET_1[20+1];
    TEXT W_STREET_2[20+1];
    TEXT W_CITY[20+1];
    TEXT W_STATE[2+1];
    TEXT W_ZIP[9+1];
    REAL W_TAX;
    MONEY W_YTD;
    } warehouse_row;

typedef struct
    {
    ID D_ID;
    ID D_W_ID;
    TEXT D_NAME[10+1];
    TEXT D_STREET_1[20+1];
    TEXT D_STREET_2[20+1];
    TEXT D_CITY[20+1];
    TEXT D_STATE[2+1];
    TEXT D_ZIP[9+1];
    REAL D_TAX;
    MONEY D_YTD;
    ID D_NEXT_O_ID;
    } district_row;


typedef struct
    {
    ID C_ID;
    ID C_D_ID;
    ID C_W_ID;
    TEXT C_FIRST[16+1];
    TEXT C_MIDDLE[2+1];
    TEXT C_LAST[16+1];
    TEXT C_STREET_1[20+1];
    TEXT C_STREET_2[20+1];
    TEXT C_CITY[20+1];
    TEXT C_STATE[2+1];
    TEXT C_ZIP[9+1];
    TEXT C_PHONE[16+1];
    DATE C_SINCE;
    TEXT C_CREDIT[2+1];
    MONEY C_CREDIT_LIM;
    REAL C_DISCOUNT;
    MONEY C_BALANCE;
    MONEY C_YTD_PAYMENT;
    COUNT C_PAYMENT_CNT;
    COUNT C_DELIVERY_CNT;
    TEXT  C_DATA[500+1];
    } customer_row;

typedef struct
    {
    ID H_C_ID;
    ID H_C_D_ID;
    ID H_C_W_ID;
    ID H_D_ID;
    ID H_W_ID;
    DATE H_DATE;
    MONEY H_AMOUNT;
    TEXT H_DATA[24+1];
    } history_row;

typedef struct
    {
    ID NO_O_ID;
    ID NO_D_ID;
    ID NO_W_ID;
    } neworder_row;

typedef struct
    {
    ID O_ID;
    ID O_D_ID;
    ID O_W_ID;
    ID O_C_ID;
    DATE O_ENTRY_D;
    ID O_CARRIER_ID;
    COUNT O_OL_CNT;
    LOGICAL O_ALL_LOCAL;
    } order_row;

typedef struct
    {
    ID OL_O_ID;
    ID OL_D_ID;
    ID OL_W_ID;
    ID OL_NUMBER;
    ID OL_I_ID;
    ID OL_SUPPLY_W_ID;
    DATE OL_DELIVERY_D;
    COUNT OL_QUANTITY;
    MONEY OL_AMOUNT;
    TEXT OL_DIST_INFO[24+1];
    } orderline_row;

typedef struct
    {
    ID I_ID;
    ID I_IM_ID;
    TEXT I_NAME[24+1];
    MONEY I_PRICE;
    TEXT I_DATA[50+1];
    } item_row;

typedef struct
    {
    ID S_I_ID;
    ID S_W_ID;
    COUNT S_QUANTITY;
    TEXT S_DIST_01[24+1];
    TEXT S_DIST_02[24+1];
    TEXT S_DIST_03[24+1];
    TEXT S_DIST_04[24+1];
    TEXT S_DIST_05[24+1];
    TEXT S_DIST_06[24+1];
    TEXT S_DIST_07[24+1];
    TEXT S_DIST_08[24+1];
    TEXT S_DIST_09[24+1];
    TEXT S_DIST_10[24+1];
    COUNT S_YTD;
    COUNT S_ORDER_CNT;
    COUNT S_REMOTE_CNT;
    TEXT S_DATA[50+1];
    } stock_row;


/* Empty field values */
#define EMPTY_NUM (MAXINT-1)
#define INVALID_NUM (MAXINT)
#define EMPTY_FLT   (MAXDOUBLE)
#define INVALID_FLT (MINDOUBLE)

/* Status conditions */
```

```
#define OK 0
#define E 1
#define E_INVALID_ITEM 2
#define E_NOT_ENOUGH_ORDERS 3
#define E_DB_ERROR 4
#define E_INVALID_INPUT 5
#define E_DB_IRRECERR 6

/* Error message strings */
extern const char *e_mesg[];


#define YES 1
#define NO 0



double cvt_flt();
double cvt_money();
TIME getclock();
TIME getlocalclock();

#define TPC_MSG_QUE  150


/***************************************
Transaction specific stuff
***************************************/

/* types of transactions */
#define NEWORDER 1
#define PAYMENT  2
#define ORDSTAT  3
#define DELIVERY 4
#define STOCKLEV 5
#define DEFERRED 6   /* deferred portion of delivery */

/* the name of each transaction */
extern const char *transaction_name[];

#endif /* TPCC_INCLUDED */
```

lib/key_chars.h

```
#ifndef __TPCC_KEY_CHARS__
#define __TPCC_KEY_CHARS__

/* Standard characters used for screen control */
#define ENTER '\015'
#define TAB   '\t'
#define BACKTAB '\02'  /*  ^B  */
#define CNTRLC '\03'
#define BACKSPACE '\010'
#define BELL '\07'
#define BLANK ' '
#define UNDERLINE '_'
#define ESCAPE '\033'
#define TRIGGER '\021'  /* dc1 */
#define TRIGGER2 '\022'  /* dc2 */
#endif
```

lib/errlog.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/12/06 13:49:16 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#include <stdio.h>
#include <varargs.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <fcntl.h>

int userid;
int msgfile_fd = -10000;
#define MSG_BUF_SIZE 3*1024

static msg_buf();

error(format, va_alist)
/************************************************************************
error formats a message and outputs it to a standard location (stderr for now)
************************************************************************/
   char *format;
                va_dcl
   {
   va_list argptr;

   msg_buf("error \n", strlen("error \n"));

   /* point to the list of arguments */
   va_start(argptr);

   /* format and print to stderr */
   vmessage(format, argptr);

   /* done */
   va_end(argptr);

   /* take an error exit */
   exit(1);
   }


syserror( format, va_alist )
/************************************************************************
syserror logs a message with the system error code
************************************************************************/
   char *format;
                va_dcl
   {
   va_list argptr;
   int save_errno = errno;

   msg_buf("syserror \n", strlen("syserror \n"));
   /* point to the list of arguments */
   va_start(argptr);

   /* format and print to stderr */
   vmessage(format, argptr);

   /* done */
   va_end(argptr);

   /* display the system error message */
   message("    System error message: %d %s\n", save_errno, strerror(save_errno));

   /* take an error exit */
   exit(1);
   }



message(format, va_alist)
/************************************************************************
message formats a message and outputs it to a standard location (stderr for now)
************************************************************************/
   char *format;
                va_dcl
{
   va_list argptr;

   msg_buf("message \n", strlen("message \n"));
   /* point to the list of arguments */
   va_start(argptr);

   /* format and print to stderr */
   vmessage(format, argptr);

   /* done */
   va_end(argptr);
   }



vmessage(format, argptr)
/*************************************************

*************************************************/
   char *format;
   va_list argptr;
{
  char buf[MSG_BUF_SIZE];

  /* format a message id */
  sprintf(buf, "Host %-8s User %-6d Pid %-6d ", getenv("HOST_NAME"), userid, getpid());

  /* format the string and print it */
  vsprintf(buf+strlen(buf), format, argptr);
  if (getenv("NO_ERROR_LOG") == NULL)
     msg_buf(buf, strlen(buf));
  if (getenv("NO_STDERR") == NULL)
     write(2, buf, strlen(buf));
}
```

```
static msg_buf(buf, size)
  char *buf;
  int size;
  {
  char  *fname;
  time_t tepoch = time(NULL);
  char writebuf[MSG_BUF_SIZE+66];
  int ltimestamp;

  ltimestamp = strftime(writebuf, 64, "%m/%d %T ", localtime(&tepoch));

  /* get the file name to use */
  fname = getenv("ERROR_LOG");
  if (fname == NULL)
    fname = "/tmp/ERROR_LOG";

  /* get exclusive access to the error log file */
  if (msgfile_fd == -10000)     {
      msgfile_fd= open(fname, O_WRONLY | O_CREAT | O_APPEND, 0666);
      if (msgfile_fd < 0)
                  console_error("Can't open tpc error log file 'ERROR_LOG'\n");
  }
  strncpy(writebuf+ltimestamp, buf, size);
  write(msgfile_fd, writebuf, ltimestamp + size);
  }



console_error(str)
   char *str;
   {
   int fd = open("/dev/tty", O_WRONLY);
   write(fd, str, strlen(str));
   close(fd);
   exit(1);
   }
```

lib/fmt.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2002/07/18 22:07:33 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#include "tpcc.h"
#include <math.h>    /* needed for ceil (VM) */
#include <strings.h>

/* formatting routines.  */

/* Note: Currently use integer routines to format and convert.  Need to
   modify the code for cases when integers don't work.  */

fmt_money(str, m, width)
    char *str;
    MONEY m;
    int width;
    {

    if (m == EMPTY_FLT)
        {
        memset(str, '_', width);
        str[width] = '\0';
        return;
        }

    /* format it as a number with a leading blank */
    *str = ' ';
    fmt_flt(str+1, m/100, width-1, 2);

    /* fill in a leading dollar */
    while (*(str+1) == ' ')
        str++;
    *str = '$';
    }


double cvt_money(str)
    char *str;
    {
    char temp[81], *t, *s;
    double cvt_flt(), f;

    /* skip leading and trailing blanks */
    cvt_text(str, temp);

    /* remove leading $ */
    if (*temp == '$')   t = temp + 1;
    else                t = temp;

    /* start scan at current character */
    s = t;

    /* allow leading minus sign */
    if (*s == '-')
                s++;

    /* allow leading digits */
    while (isdigit(*s))
                s++;

    /* allow decimal pt and two decimal digits */
    if (*s == '.')    s++;
    if (isdigit(*s))  s++;
    if (isdigit(*s))  s++;

    /* There should be no more characters */
    if (*s != '\0') return INVALID_FLT;

    /* convert the floating pt number */
    f = cvt_flt(t);
    if (f == EMPTY_FLT)        return EMPTY_FLT;
    else if (f == INVALID_FLT) return INVALID_FLT;
    else                       return rint(f*100);
    }


fmt_num(str, n, width)
    char str[];
    int n;
    int width;
    {
    /* mark the end of the string */
    str[width] = '\0';

    /* if empty number, return the empty field */
    if (n == EMPTY_NUM)
        memset(str, '_', width);

    /* otherwise, convert the integer */
    else
        fmtint(str, n, width, ' ');

    debug("fmt_num:  n=%d  str=%s\n", n, str);
    }



cvt_num(str)
    char str[];
    {
    char text[81];
    cvt_text(str, text);
    if (*text == '\0')
        return EMPTY_NUM;
    else
        return cvtint(text);
    }



fmt_flt(str, x, width, dec)
/*****************************************************************************
fmt_flt converts a floating pt number to a string   "999999.9999"
*****************************************************************************/
    char *str;
    double x;
    int width;
    int dec;
    {
    int negative;
    int integer, fract;
    double absolute;

    static const double pow10[] =
    {1., 10., 100., 1000., 10000., 100000., 1000000., 10000000., 100000000.};

    /* mark the end of string */
    str[width] = '\0';

    /* if empty value, make it be an empty field */
    if (x == EMPTY_FLT)
        {
        memset(str, '_', width);
        return;
        }

    absolute = (x < 0)? -x: x;

    /* separate into integer and fractional parts */
    integer = (int) absolute;
    fract   = (absolute - integer) * pow10[dec] + .5;

    /* let the integer portion contain the sign */
    if (x < 0) integer = -integer;
```

```
    /* Format integer and fraction separately */
    fmtint(str, integer, width-dec-1, ' ');
    str[width-dec-1] = '.';
    fmtint(str+width-dec, fract, dec, '0');
    }



double cvt_flt(str)
    char str[];
    {
    char text[81];
    char *t;
    double value;
    int div;
    int fract;
    int negative;
    int i;

    /* normalize the text */
    cvt_text(str, text);
    if (*text == '\0')
        return EMPTY_FLT;

    negative = NO;
    fract = NO;
    value = 0;
    div = 1.0;

    negative = (text[0] == '-');
    if (negative)    t = text+1;
    else             t = text;

    for (;  *t !=  '\0' ; t++)
        {

        if (*t == '.')
                        if (fract)  return INVALID_FLT;
            else        fract = YES;

        else if (isdigit(*t))
            {
            value = value*10 + (int)*t - (int)'0';
            if (fract) div *= 10;
            }

        else
            return INVALID_FLT;
        }

    if (fract)
        value /= div;

    if (negative)
        value = -value;

    return value;
    }



fmt_text(s, text, width)
    char *s, *text;
    int width;
    {

    /* if an empty string, then all underscores */
    if (*text == '\0')
        for (; width > 0; width--)
            *s++  = '_';

    /* otherwise, blank fill it */
    else
        {

        /* copy the text into the new buffer */
        for ( ; *text != '\0';  width--)
            *s++ = *text++;

        /* fill in the rest with blanks */
        for (;  width > 0;  width--)
            *s++ = ' ';
        }

    /* and finally, terminate the string */
    *s = '\0';
    }



cvt_text(s, text)
    char *s;
    char *text;
    {
    char *lastnb;

    /* skip leading blanks and underscores */
    for (; *s == ' ' || *s == '_'; s++)
        ;

    /* copy the characters, keeping track of last blank or underscore */
    lastnb = text-1;
    for (; *s != '\0'; *text++ = *s++)
        if (*s != ' ' && *s != '_')
            lastnb = text;

    /* truncate the text string to last nonblank character */
    *(lastnb+1) = '\0';
    }



fmtint(field, value, size, fill)
/*********************************************************************
fmtint formats an integer value into a character field to make the integer
right-justified within the character field, padded with leading fill
characters (e.g. leading blanks if a blank is passed in for the fill argument
*********************************************************************************/
    int value;
    char *field;
    int size;
    char fill;
    {
    int negative;
    int dividend;
    int remainder;
    char *p;

    /* create characters from right to left */
    p = field + size - 1;

    /* make note if this is a negative number */
    negative = value < 0;
    if (negative)
        value = -value;

    /* Case: Null field.  Can't do anything */
    if (p < field)
        ;

    /* Case: value is zero.  Print a leading '0' */
    else if (value == 0)
        *p-- = '0';

    /* Otherwise, convert each digit in turn */
    else do
        {

        dividend = value / 10;
        remainder = value - dividend * 10;
        value = dividend;

        *p-- = (char) (  (int)'0'  +  remainder  );

        } while (p >= field && value > 0);

    /* insert a minus sign if appropriate */
    if (negative && p >= field)
        *p-- = '-';

    /* fill in leading characters */
    while (p >= field)
        *p-- = fill;
    }


int cvtint(str)
/*****************************************************************
getint extracts an integer value from the given character field
(ex: turns the string "123" into the integer 123)
*****************************************************************/
    char *str;
    {
    int value;
    char c;
    int negative;
    debug("cvtint: str=%s\n", str);

    negative = (*str == '-');
    if (negative)  str++;

    /* convert the integer */
    for (value = 0; isdigit(*str); str++)
        value = value*10 + (int)(*str) - (int)'0';

    /* if any non-digit characters, error */
    if (*str != '\0')
```

```
                return INVALID_NUM;

    /* make negative if there was a minus sign */
    if (negative)
        value = -value;

    debug("cvtint: value=%d\n", value);
    return value;
    }



fmt_phone(str, phone)
    char str[20];
    char *phone;

    {
    /* copy phone number and insert dashes  999999-999-999-9999 */
    str[0] = phone[0];  str[1] = phone[1]; str[2] = phone[2];
    str[3] = phone[3];  str[4] = phone[4]; str[5] = phone[5];
    str[6] = '-';
    str[7] = phone[6];  str[8] = phone[7]; str[9] = phone[8];
    str[10] = '-';
    str[11] = phone[9]; str[12] = phone[10]; str[13] = phone[11];
    str[14] = '-';
    str[15] = phone[12]; str[16] = phone[13]; str[17] = phone[14];
    str[18] = phone[15];
    str[19] = '\0';
    }


fmt_zip(str,zip)
    char str[20];
    char *zip;
    {
    /* copy zip code and insert dashes 99999-9999 */
    str[0] = zip[0];  str[1] = zip[1];  str[2] = zip[2];
    str[3] = zip[3];  str[4] = zip[4];
    str[5] = '-';
    str[6] = zip[5]; str[7] = zip[6]; str[8] = zip[7];  str[9] = zip[8];
    str[10] = '\0';
    }
```

lib/iobuf.h

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/

/*********************************************************************
History
941220 LAN Added definition and initialization of the line_col[] array.
       This was needed for modifications made of client program to do
       block I/O using a WYSE terminal.
*********************************************************************/

/* structure for screen emulation */
typedef struct
    {
    int row;
    int col;
    char buf[25][81];
    } screen_t;


typedef struct {
    char *beg;
    char *end;   /* for output buffers */
    char *max;
    char *cur;   /* for input buffers */
    } iobuf;

/* Macro do define an I/O buffer of x characters, initialized to empty */
#define define_iobuf(name, size)                    \
char name##_data[size];                             \
iobuf name[1] = {{name##_data, name##_data,       \
                name##_data+size, name##_data}}

#define reset(buf)  if (1) { \
    (buf)->cur = (buf)->end = (buf)->beg;               \
    *(buf)->beg = '\0';                                 \
    } else (void)0

#define flush()  if(1) { \
        display(out_buf);    \
        reset(out_buf);    \
     } else (void)0


/* Standard I/O to and from in_buf and out_buf */
#ifdef DECLARE_IO_BUFFERS
define_iobuf(output_stuff, 4*1024);
define_iobuf(input_stuff, 1024);
iobuf *in_buf = input_stuff;
iobuf *out_buf = output_stuff;
#else
iobuf *in_buf;
iobuf *out_buf;
#endif

#define pushc(c) if (1) { \
    if (out_buf->end >= out_buf->max)   \
        error("out_buf overflow:  beg=0x%x end=%d  max=%d\n",  \
          out_buf->beg, out_buf->end-out_buf->beg,out_buf->max-out_buf->beg); \
    *(out_buf->end++) = (c);  \
    *(out_buf->end) = '\0';  /* debug */ \
    } else (void)0

#define popc()  \
    (*in_buf->cur++)


/* Standard characters used for screen control */
#define ENTER '\015'
#define TAB   '\t'
#define BACKTAB '\02'  /*  ^B  */
#define CNTRLC '\03'
#define BACKSPACE '\010'
#define BELL '\07'
#define BLANK ' '
#define UNDERLINE '_'
#define ESCAPE '\033'
/*#define EOF    ((char)-1) */
#define TRIGGER '\021'  /* dc1 */
```

lib/iobuf.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#define DECLARE_IO_BUFFERS
#include "iobuf.h"
#undef DECLARE_IO_BUFFERS
#include "tpcc.h"
#include <errno.h>



string(str)
    char str[];
    {
    for (; *str != '\0'; str++)
        pushc(*str);
    }


push(str, len)
    char *str;
    int len;
    {
    for (; len > 0; len --)
        pushc(*str++);
    }


display(scr)
    iobuf *scr;
    {
/* Note: if problems doing output, let the input routine detect it */
    char *p;
    int len;
    for (p = scr->beg; p < scr->end; p+=len)
        {
        len = write(1, p, scr->end - p);
        if (len <= 0) break;
        }
    }


input(scr)
    iobuf *scr;
    {
    int len;

    /* read in as many characters as are available */
    len = read(0, scr->end, scr->max - scr->end);

    /* if end of input, then pretend we read an END character */
```

```
    if (len == 0 || (len == -1 && errno == ECONNRESET))
        {
        *scr->end = EOF;
        len = 1;
        }

    /* Check for errors */
    else if (len == -1)
        syserror("input(scr): unable to read stdin\n");

    /* update the pointers to reflect the new data */
    scr->end += len;
    *scr->end='\0';  /* for debugging */
    }



getkey()
    {
    if (in_buf->cur == in_buf->end)
        {
        flush();
        reset(in_buf);
        input(in_buf);
        }

    return popc();
    }
```

lib/random.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2002/07/18 22:07:40 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#include "tpcc.h"
#include "string.h"
#include "random.h"

double drand48();

char lastNames[1000][16];
char customerData1[10][301];
char customerData2[10][201];
char stockData1[10][27];
char stockData2[10][25];
char historyData1[10][13];
char historyData2[10][13];
char citystreetData1[10][11];
char citystreetData2[10][11];
char firstNameData1[10][9];
char firstNameData2[10][9];
char StockDistrict[10][25];
char phoneData[10][17];

static long RandySeedIter = 7;

void GenerateLastNames()
{
    int     i;
    char *name;
    static const char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
                    "ESE", "ANTI", "CALLY", "ATION", "EING"};

    for(i = 0; i < 1000; i++) {
        name = lastNames[i];
        strcpy(name, n[(i/100)%10]);
        strcat(name, n[(i/10) %10]);
        strcat(name, n[(i/1)  %10]);
    }
}

int MakeNumberString(min, max, num)
    int min;
    int max;
    TEXT num[];
    {
    static const char digit[]="0123456789";
    int length;
    int i;

    length = RandomNumber(min, max);

    for (i=0;  i<length;  i++)
        num[i] = digit[RandomNumber(0,9)];
    num[length] = '\0';

    return length;
    }

ID RandomWarehouse(local, scale, percent)
    ID local;
    ID scale;
    int percent;   /* percent of remote transactions */
    {
    ID w_id;

    /* For the given percent of the time, pick the local warehouse */
    if (RandomNumber(1, 100) > percent || scale == 1)
        w_id = local;

    /* Otherwise, pick a non-local warehouse */
    else
        {
        w_id = RandomNumber(2, scale);
        if (w_id == local)
            w_id = 1;
        }
    return w_id;
    }


/* Initialize a table of Random strings for the stock-district
   field in the stock table.  We can use a table of 10 elements
   and select randomly from this table via rule 4.3.2.2 in
   the TPC-C spec */
void InitRandomStrings()
{
    int i;

    for (i=0; i < 10; i++) {
                    MakeAlphaString(24,24,&StockDistrict[i]);

                    MakeAlphaString(300,300,&customerData1[i]);
                    MakeAlphaString(0,200,&customerData2[i]);

                    MakeAlphaString(26,26,&stockData1[i]);
                    MakeAlphaString(0,24,&stockData2[i]);

                    MakeAlphaString(12,12,&historyData1[i]);
                    MakeAlphaString(0,12, &historyData2[i]);

                    MakeAlphaString(10,10,&citystreetData1[i]);
                    MakeAlphaString(0 ,10,&citystreetData2[i]);

                    MakeAlphaString(8,8,&firstNameData1[i]);
                    MakeAlphaString(0,8,&firstNameData2[i]);

                    MakeNumberString(16,16,&phoneData[i]);
                }
                GenerateLastNames();
}

int MakeAlphaString(min, max, str)
    int min;
    int max;
    TEXT str[];
    {
    static const char character[] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    int length;
    int i;

    length = RandomNumber(min, max);

    for (i=0;  i<length;  i++) {
        /* NOTE:  we use sizeof(character)-2 because of the following:
           subtract 1 because we are numbering from 0 instead of 1 and
           subtract 1 because the sizeof(character) is 1 greater than
           the data in character because of the inivisible C string
           terminator at the end. */
        str[i] = character[RandomNumber(0, sizeof(character)-2)];
    }
    str[length] = '\0';

    return length;
    }

void RandomPermutation(perm, n)
    int perm[];
    int n;
    {
    int i, r, t;

    /* generate the identity permutation to start with */
    for (i=1; i<=n; i++)
        perm[i] = i;

    /* randomly shuffle the permutation */
    for (i=1; i<=n; i++)
        {
        r = RandomNumber(i, n);
        t = perm[i]; perm[i] = perm[r]; perm[r] = t;
        }
```

```
    }


void RandomDelay(mean, adjust)
/*********************************************************************
random_sleep sleeps according to the TPC specification
*********************************************************************/
    double mean;
    double adjust;
    {
    double secs;
    double exponential();

    secs = exponential(mean);

    delay(secs+adjust);
    }

double exponential(mean)
/*********************************************************************
exponential generates a reverse exponential distribution
*********************************************************************/
    double mean;
    {
    double x;
    double log();

#ifdef USE_DRAND48
    x = -log(1.0-drand48()) * mean;
#else
    x = -log(1.0-randy()) * mean;
#endif

    return x;
    }

void SetRandomSeed(val)
long val;
    {
#ifdef USE_DRAND48
    srand48(val);
#else
    RandySeedIter = val;
    randy();
#endif
    }

void Randomize()
    {
    SetRandomSeed(time(0)+getpid());
    }

/* Random number generator from Proceeding of the ACM */
#define RANDY_A_VAL 16807
/* 2^31 - 1 */
#define RANDY_M_VAL 2147483647
/* m / a */
#define RANDY_Q_VAL 127773
/* m % a */
#define RANDY_R_VAL 2836

double randy()
{
  long    hi, lo, test;

  hi = RandySeedIter / RANDY_Q_VAL;
  lo = RandySeedIter % RANDY_Q_VAL;

  test = (RANDY_A_VAL * lo) - (RANDY_R_VAL * hi);
  RandySeedIter = (test > 0) ? test : test + RANDY_M_VAL;

  return( (double)RandySeedIter / (double)RANDY_M_VAL );
}  /* end of fn randy  */
```

lib/Makefile

```
#*****************************************************************************
*
#@(#) Version: A.10.10 $Date: 2002/07/18 22:02:15 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************
*

include ../buildenv.mk

CFLAGS= $(BUILDFLAGS) -Wl,-a,archive_shared

utils=delay.o errlog.o fmt.o random.o tas.o null_key.o null_select.o results_file.o  date.o
prepare_socket.o shm.o spinlock.o tpcc.o

all: tpc_lib.a  server_default.o

tpc_lib.a: ${utils}
        rm -f tpc_lib.a
        ar -r tpc_lib.a ${utils}

clean:
        rm -f *.o
        rm -f *.a

clobber: clean

.s.o:
        cc $(DATA_MODEL_FLAGS) -c $*.s
```

lib/date.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2002/07/18 22:07:16 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#include "tpcc.h"
#include <time.h>

/* macro to get starting day of a particular year (1901 thru 2100) */
#define YEAR(yr)  ( (yr-1900)*365 + (yr-1900-1)/4 )

CurrentDate(date)
/*********************************************************************
CurrentDate fetches the current date and time
*********************************************************************/
    DATE *date;
    {
    struct timeval time;
    struct timezone tz;

    /* get the current time of day */
    if (gettimeofday(&time, &tz) < 0)
        syserror("Can't get time of day\n");

    /* adjust the time of day by the timezone */
    time.tv_sec -= tz.tz_minuteswest * 60;

    /* convert seconds and days since EPOCH (Jan 1, 1970) */
    date->day = time.tv_sec / (24*60*60);
    date->sec = time.tv_sec - date->day * (24*60*60);

    /* convert to days since Jan 1, 1900 */
    date->day += YEAR(1970);
    }


EmptyDate(date)
/*****************************************************
Get a NULL date and time
*****************************************************/
    DATE *date;
    {
    date->day = -1;
    date->sec = 0;
    }

int IsEmptyDate(date)
    DATE *date;
    {
    return (date->day == -1 && date->sec == 0);
    }


#define Feb29 (31+29-1)

fmt_date(str, date)
/*********************************************************************
fmt_date formats the DATE into a string  MM-DD-YY HH-MM-SS
*********************************************************************/
    char str[20];
    DATE *date;
    {
    /* Note: should probably do date and time separately */

    int quad, year, month, day;
    int hour, minute, sec;

    static int dur[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    static int first = YES;

    day = date->day;
    sec = date->sec;

    /* if NULL date, then return empty string */
    if (day == -1 && sec == 0)
        {str[0] = '\0';  return;}
```

```
/* 2100, 1900 are NOT leap years.  If we are Feb 29 or later, add a day */
if (day >= Feb29 + YEAR(2100))  day++;
if (day >= Feb29)               day++;

/* figure out which quad and day within quad we are in */
quad = day / (4*365+1);
day = day - quad * (4*365+1);

/* get our year within quad and day within the year */
if      (day < 1*365+1)    {year = 0;}
else if (day < 2*365+1)    {year = 1; day -= 1*365+1;}
else if (day < 3*365+1)    {year = 2; day -= 2*365+1;}
else                       {year = 3; day -= 3*365+1;}

/* if this is a leap year, february has 29 days */
if (year == 0)          dur[1] = 29;
else                    dur[1] = 28;

/* decide which day and month we are */
for (month = 0; day >= dur[month]; month++)
   day -= dur[month];

/* decide what time of day it is */
minute = sec / 60;
sec = sec - minute * 60;
hour = minute / 60;
minute = minute - hour * 60;

/* format the date and time */
fmtint(str+0,  day+1, 2, ' ');
str[2]='-';
fmtint(str+3, month+1, 2, '0');
str[5]='-';
fmtint(str+6, 1900+quad*4+year, 4, '0');
str[10] = ' ';
fmtint(str+11, hour, 2, ' ');
str[13] = ':';
fmtint(str+14, minute, 2, '0');
str[16] = ':';
fmtint(str+17, sec, 2, '0');
str[19] = '\0';
}
```

/libprepare_socket.c

```
/*********************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*********************************************************************************/
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <netdb.h>

int prepare_socket(fd)
   int fd;
   {
   int yes = 1;

   if (setsockopt(fd, SOL_SOCKET, SO_KEEPALIVE, &yes, sizeof(yes)) < 0)
      return -1;
   if (setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(yes)) < 0)
      return -1;
   if (setsockopt(fd, SOL_SOCKET, TCP_NODELAY, &yes, sizeof(yes)) < 0)
                  return -1;
   /* SO_SNDBUF, SO_RCVBUF, TPC_MAXSEG  should be set ?? */
   return 0;
   }
```

/lib/server_default.c

```
/*********************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*********************************************************************************/
int server_default = 1;
```

A.3 Transaction Source

client/service.c

```
/*********************************************************************************
 @(#) Version: A.10.10 $Date: 2001/12/06 12:31:26 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*********************************************************************************/

#include <unistd.h>
#include <sys/types.h>
#include "tpcc.h"
#include "atmi.h"

extern int userid;
char *cmd          = NULL;

int tpsvrinit(argc, argv)
int    argc;
char **argv;
{
   char c;
   int ret;
   time_t t;

   t = time((time_t *) NULL);
   userlog("starting up at time %s", ctime(&t));
   /*
    * search for the options
    *    "-n" server number
    *    "-S" server program
    * purpose: to get svr_id & progname for DVRY_LOG files
    *
    */
   while ((c = getopt(argc, argv, "n:S:h:")) != EOF) {
      switch(c) {
      case 'n':
         userid = atoi(optarg);
         break;
      case 'S':
         cmd  = optarg;
         break;
      }
   }

   ret = transaction_begin(userid);
   results_open(userid);

   return 0;
}


void NEWO_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
   neworder_transaction((neworder_trans *)svcinfo->data);
   tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}


void PMT_SVC(svcinfo)
TPSVCINFO *svcinfo;
{

   payment_transaction((payment_trans *)svcinfo->data);
   tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void ORDS_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
   ordstat_transaction((ordstat_trans *)svcinfo->data);
   tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void STKL_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
   stocklev_transaction((stocklev_trans *)svcinfo->data);
   tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void DVRY_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
   delivery_trans *t = (delivery_trans *)svcinfo->data;
   gettimeofday(t->deque, NULL);
   delivery_transaction(t);
   gettimeofday(t->complete, NULL);
   results(t);
```

```
    /* Why do we return things ? */
    tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

/*************************************************************
tpsrvdone cleans up after the TPC transaction service
*************************************************************/
void tpsvrdone()
{
    transaction_done();
    results_close();

    /* Log a message saying we are done */
    userlog("TUXEDO service %s has shutdown\n", cmd);
}
```

client/oracle/transaction.c

```
#include "ora_tpcc.h"
#include <time.h>
#include "tpcc.h"

/* Always use plsql for delivery.                    */
#define PLSQLDEL

#ifdef __STDC__
#include "ociapr.h"
#else
#include "ocikpr.h"
#endif

extern TPCinit(int, char*, char*);

int     numtrans = 0;

transaction_done ()
{
    /* fprintf(stderr, "About to call TPCexit\n");fflush(stderr); */
    TPCexit();
    /* fprintf(stderr, "TPCexit after %d transcations \n", numtrans);fflush(stderr); */
}

/* void  */
transaction_begin(id)
int id;
{
    int ret;

    if ((ret=TPCinit(id, "tpcc", "tpcc")) == -1)
    {
                    fprintf(stderr, "TPCinit failure!\n");fflush(stderr);
        /* Error */
    }
    numtrans = 0;
    return ret;
}

void neworder_transaction(str)
neworder_trans *str;
{
    int i;
    struct newstruct ora_str;

    ora_str.newin.w_id = str->W_ID;
    ora_str.newin.d_id = str->D_ID;
    ora_str.newin.c_id = str->C_ID;
    for (i = 0; i < str->O_OL_CNT; i++) {
      ora_str.newin.ol_i_id[i] = str->item[i].OL_I_ID;
      ora_str.newin.ol_supply_w_id[i] = str->item[i].OL_SUPPLY_W_ID;
      ora_str.newin.ol_quantity[i] = str->item[i].OL_QUANTITY;
    }
    for (i = str->O_OL_CNT; i < 15; i++) {
      ora_str.newin.ol_i_id[i] = 0;
      ora_str.newin.ol_supply_w_id[i] = 0;
      ora_str.newin.ol_quantity[i] = 0;
    }

    numtrans++;
    if (TPCnew(&ora_str) == -1) {
      str->status = E_DB_ERROR;
      return;
    } else {
      str->status = OK;
    }

    str->O_ID = ora_str.newout.o_id;
    str->O_OL_CNT = ora_str.newout.o_ol_cnt;
    strncpy (str->C_LAST, ora_str.newout.c_last, 17);
    strncpy (str->C_CREDIT, ora_str.newout.c_credit, 3);
    str->C_DISCOUNT = (REAL) ora_str.newout.c_discount;
    str->W_TAX = (REAL) ora_str.newout.w_tax;
    str->D_TAX = (REAL) ora_str.newout.d_tax;
    strncpy (str->O_ENTRY_D, ora_str.newout.o_entry_d, 20);
    for (i = 0; i < ora_str.newout.o_ol_cnt; i++) {
      strncpy (str->item[i].I_NAME, ora_str.newout.i_name[i], 25);
      str->item[i].S_QUANTITY = ora_str.newout.s_quantity[i];
      str->item[i].brand_generic = ora_str.newout.brand_generic[i];
      str->item[i].I_PRICE = ora_str.newout.i_price[i]*100.0;  /* needs to be in cents */
    }
    str->status = ((ora_str.newout.status[0] != '\0') ? E_INVALID_ITEM : OK);
}

/*************************************************************************
 * Payment Query
 *************************************************************************/

void
payment_transaction(str)
payment_trans *str;
{
    int i;

    struct paystruct ora_str;

    ora_str.payin.w_id = str->W_ID;
    ora_str.payin.d_id = str->D_ID;
    ora_str.payin.c_w_id = str->C_W_ID;
    ora_str.payin.c_d_id = str->C_D_ID;
    ora_str.payin.h_amount = str->H_AMOUNT; /* Amount in cents */
    ora_str.payin.bylastname = str->byname;
    if (ora_str.payin.bylastname) {
      ora_str.payin.c_id = 0;
      strncpy (ora_str.payin.c_last, str->C_LAST, 17);
      ora_str.payin.c_last[16] = '\0';
      for (i = 15; ((i >= 0) && (ora_str.payin.c_last[i] == ' ')); i--)
        ora_str.payin.c_last[i] = '\0';
    }
    else {
      ora_str.payin.c_id = str->C_ID;
      strcpy (ora_str.payin.c_last, " ");
    }
    retries = 0;

    numtrans++;
    if (TPCpay (&ora_str)) {
      str->status = E_DB_ERROR;
      return;
    } else {
      str->status = OK;
    }

    strncpy (str->W_STREET_1, ora_str.payout.w_street_1, 21);
    strncpy (str->W_STREET_2, ora_str.payout.w_street_2, 21);
    strncpy (str->W_CITY, ora_str.payout.w_city, 21);
    strncpy (str->W_STATE, ora_str.payout.w_state, 3);
    strncpy (str->W_ZIP, ora_str.payout.w_zip, 10);
    strncpy (str->D_STREET_1, ora_str.payout.d_street_1, 21);
    strncpy (str->D_STREET_2, ora_str.payout.d_street_2, 21);
    strncpy (str->D_CITY, ora_str.payout.d_city, 21);
    strncpy (str->D_STATE, ora_str.payout.d_state, 3);
    strncpy (str->D_ZIP, ora_str.payout.d_zip, 10);
    str->C_ID = ora_str.payout.c_id;
    strncpy (str->C_FIRST, ora_str.payout.c_first, 17);
    strncpy (str->C_MIDDLE, ora_str.payout.c_middle, 3);
    strncpy (str->C_LAST, ora_str.payout.c_last, 17);
    strncpy (str->C_STREET_1, ora_str.payout.c_street_1, 21);
    strncpy (str->C_STREET_2, ora_str.payout.c_street_2, 21);
    strncpy (str->C_CITY, ora_str.payout.c_city, 21);
    strncpy (str->C_STATE, ora_str.payout.c_state, 3);
    strncpy (str->C_ZIP, ora_str.payout.c_zip, 10);
    strncpy (str->C_PHONE, ora_str.payout.c_phone, 17);
    strncpy (str->C_SINCE, ora_str.payout.c_since, 11);

    strncpy (str->C_CREDIT, ora_str.payout.c_credit, 3);
    str->C_CREDIT_LIM = (MONEY) ora_str.payout.c_credit_lim*100.0; /* needs to be in cents
*/
    str->C_DISCOUNT = (REAL) ora_str.payout.c_discount;
    str->C_BALANCE = (REAL) ora_str.payout.c_balance*100.0;  /* needs to be in cents */
    /* Oracle passes 201 characters, we copy 200 and terminate on 201.        */
    strncpy (str->C_DATA, ora_str.payout.c_data, 200);
    str->C_DATA[200] = '\0';
    strncpy (str->H_DATE, ora_str.payout.h_date, 20);
}

void
ordstat_transaction(str)
ordstat_trans *str;
{
    int i;

    struct ordstruct ora_str;

    ora_str.ordin.w_id = str->W_ID;
    ora_str.ordin.d_id = str->D_ID;
    ora_str.ordin.bylastname = str->byname;
```

```
  if (ora_str.ordin.bylastname) {
    ora_str.ordin.c_id = 0;
    strncpy (ora_str.ordin.c_last, str->C_LAST, 17);
    ora_str.ordin.c_last[16] = '\0';
    for (i = 15; ((i >= 0) && (ora_str.ordin.c_last[i] == ' ')); i--)
      ora_str.ordin.c_last[i] = '\0';
  }
  else {
    ora_str.ordin.c_id = str->C_ID;
    strcpy (ora_str.ordin.c_last, " ");
  }
  retries = 0;

  numtrans++;
  if (TPCord (&ora_str)) {
    str->status = ora_str.ordout.terror;
    if (ora_str.ordin.bylastname) {
                message("Order status error: wid = %d, did = %d, name = %s\n", str->W_ID, str-
>D_ID, ora_str.ordin.c_last);
    } else      {
                message("Order status error: wid = %d, did = %d, ID = %d\n", str->W_ID, str-
>D_ID, str->C_ID);
    }
    return;
  } else {
    str->status = OK;
  }

  str->C_ID = ora_str.ordout.c_id;
  strncpy (str->C_LAST, ora_str.ordout.c_last, 17);
  strncpy (str->C_FIRST, ora_str.ordout.c_first, 17);
  strncpy (str->C_MIDDLE, ora_str.ordout.c_middle, 3);
  str->C_BALANCE = (MONEY) ora_str.ordout.c_balance*100.0; /* needs to be in cents */
  str->O_ID = ora_str.ordout.o_id;
  strncpy (str->O_ENTRY_DATE, ora_str.ordout.o_entry_d, 20);
  str->O_CARRIER_ID = ora_str.ordout.o_carrier_id;
  str->ol_cnt = ora_str.ordout.o_ol_cnt;
  for (i = 0; i < ora_str.ordout.o_ol_cnt; i++) {
    str->item[i].OL_SUPPLY_W_ID = ora_str.ordout.ol_supply_w_id[i];
    str->item[i].OL_I_ID = ora_str.ordout.ol_i_id[i];
    str->item[i].OL_QUANTITY = ora_str.ordout.ol_quantity[i];
    str->item[i].OL_AMOUNT = (MONEY) ora_str.ordout.ol_amount[i]*100.0;  /* needs to be in
cents */
    strncpy (str->item[i].OL_DELIVERY_DATE, ora_str.ordout.ol_delivery_d[i], 11);
  }
}

/*****************************************************************************
 * Delivery Query
 *****************************************************************************/

void delivery_transaction(str)
delivery_trans *str;
{
  double tr_end;
  int i;

  struct delstruct ora_str;

    /* set plsql or OCI delivery */
# ifdef PLSQLDEL
  ora_str.delin.plsqlflag=1;
# else
  ora_str.delin.plsqlflag=0;
# endif

  ora_str.delin.w_id        = str->W_ID;
  ora_str.delin.o_carrier_id = str->O_CARRIER_ID;
  retries     = 0;

  numtrans++;
  if (TPCdel (&ora_str)) {
    str->status = E_DB_ERROR;
    return;
  } else {
    str->status = OK;
  }

  for (i = 0; i < 10; i++) {
    if (del_o_id[i] <= 0) {
                str->order[i].status = E_NOT_ENOUGH_ORDERS;
    } else {
                str->order[i].status = OK;
                str->order[i].O_ID = del_o_id[i];
    }
  }
}

/*****************************************************************************
 * Stock Level Query
 *****************************************************************************/

void stocklev_transaction(str)
stocklev_trans *str;
{
  struct stostruct ora_str;
  ora_str.stoin.w_id = str->W_ID;
  ora_str.stoin.d_id = str->D_ID;
  ora_str.stoin.threshold = str->threshold;
  retries = 0;

  numtrans++;
  if (TPCsto (&ora_str)) {
    str->status = E_DB_ERROR;
    return;
  } else {
    str->status = OK;
  }
  str->low_stock = ora_str.stoout.low_stock;
}
```

client/oracle/tpccpl.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: tpccpl.c 7030100.2 96/04/02 17:51:34 plai Generic<base> $ Copyr (c) 1994 Oracle";
#endif /* RCSID */


/*+============================================================================
 |      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
 |                  OPEN SYSTEMS PERFORMANCE GROUP               |
 |                       All Rights Reserved                     |
 +============================================================================
 | FILENAME
 |   tpccpl.c
 | DESCRIPTION
 |   TPC-C transactions in PL/SQL.
 +============================================================================*/

#include <stdio.h>
#include <time.h>
#include "ora_tpcc.h"
#ifdef TUX
#include <userlog.h>
#else
#include <stdarg.h>
#endif

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

FILE *lfp;
FILE *fopen ();
#ifdef ORA_NT
#undef boolean
#include "dpbcore.h"
#define gettime dpbtimef
#else
extern double gettime ();
#endif
int proc_no = 0;
static int logon = 0;
static int new_init = 0;
static int pay_init = 0;
static int ord_init = 0;
static int del_init_oci = 0;
static int del_init_plsql = 0;
static int sto_init = 0;
static int res_init = 0;

int execstatus;
int errcode;

OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
OCIStmt *curi;


/* for stock-level transaction */

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

/* for delivery transaction */

int del_o_id[10];
int retries;
```

```
/* for order-status transaction */

int bylastname;
char c_last[17];
char c_first[17];
char c_middle[3];
double c_balance;
int o_id;
text o_entry_d[20];
ub4  datelen;
int o_carrier_id;
int o_ol_cnt;
int ol_supply_w_id[15];
int ol_i_id[15];
int ol_quantity[15];
int ol_amount[15];
ub4  ol_del_len[15];
text ol_delivery_d[15][11];
/* xnie - begin */
OCIRowid *o_rowid;
/* xnie - end */

/* for payment transaction */

int c_w_id;
int c_d_id;
int h_amount;
char w_street_1[21];
char w_street_2[21];
char w_city[21];
char w_state[3];
char w_zip[10];
char d_street_1[21];
char d_street_2[21];
char d_city[21];
char d_state[3];
char d_zip[10];
char c_street_1[21];
char c_street_2[21];
char c_city[21];
char c_state[3];
char c_zip[10];
char c_phone[17];
ub4  sincelen;
text c_since_d[11];
float c_discount;
char c_credit[3];
int c_credit_lim;
char c_data[201];
ub4 hlen;
text h_date[20];

/* for new order transaction */

int nol_i_id[15];
int nol_supply_w_id[15];
int nol_quantity[15];
int nol_quanti10[15];
int nol_quanti91[15];
int nol_ytdqty[15];
int nol_amount[15];
int o_all_local;
float w_tax;
float d_tax;
float total_amount;
char i_name[15][25];
int s_quantity[15];
char brand_gen[15];
int i_price[15];
char brand_generic[15][1];
int status;
int tracelevel = 0;

OCIDate cr_date;
OCIDate c_since;
OCIDate o_entry_d_base;
OCIDate ol_d_base[15];
dvoid *xmem;

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

/*
extern char oracle_home[256];
*/

/* NewOrder Binding stuff */



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

/* vmm313 void ocierror(fname, lineno, errhp, status)  */
int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
  text errbuf[512];
  sb4 errcode;
  sb4 lstat;
  ub4 recno=2;

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


FILE  *vopen(fnam,mode)
char  *fnam;
char  *mode;
{
FILE *fd;

#ifdef  DEBUG
  fprintf(stderr, "tkvuopen() fnam: %s,   mode: %s\n", fnam, mode);
#endif

    fd = fopen((char *)fnam,(char *)mode);
    if (!fd){
        fprintf(stderr," fopen on %s failed %d\n",fnam,fd);
        exit(-1);
    }
    return(fd);
}

int sqlfile(fnam,linebuf)
```

```
char   *fnam;
text   *linebuf;
{
FILE *fd;
int nulpt = 0;
char realfile[512];

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

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

#ifdef NOT
void vgetdate (unsigned char *oradt)
{
 struct tm *loctime;
 time_t  int_time;

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

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

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

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

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

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


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

 return;

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

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


       int day,month,year;

       memcpy(&Date,oradt,7);

       year = (Date.century-100)*100 + Date.year-100;
       month = Date.month;
       day = Date.day;
       sprintf(outdate,"%02d-%02d-%4d\0",day,month,year);

       return;
}

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

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


       int day,month,year;
       int hour,min,sec;

       memcpy(&Date,oradt,7);

       year = (Date.century-100)*100 + Date.year-100;
       month = Date.month;
       day = Date.day;
       hour = Date.hour - 1;
       min = Date.minute - 1;
       sec = Date.second - 1;

       sprintf(outdate,"%02d-%02d-%4d %02d:%02d:%02d\0",
                    day,month,year,hour,min,sec);

       return;
}
#endif


void TPCexit (void)

{

  if (new_init) {
    tkvcndone();
    new_init = 0;
  }
  if (pay_init) {
    tkvcpdone();
    pay_init = 0;
  }
  if (ord_init) {
    tkvcodone();
    ord_init = 0;
  }
  if (del_init_oci) {
    tkvcddone(0);
    del_init_oci = 0;
  }
  if (del_init_plsql) {
    tkvcddone(1);
    del_init_plsql = 0;
  }
  if (sto_init) {
    tkvcsdone();
    sto_init = 0;
  }


  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 (lfp) {
    fclose (lfp);
    lfp = NULL;
  }

}



TPCinit (id, uid, pwd)

int id;
char *uid;
char *pwd;

{

  char filename[40];
```

```
  text stmbuf[100];

  proc_no = id;
  sprintf (filename, "tpcc_%d.del", proc_no);
  if ((lfp = fopen (filename, "w")) == NULL) {
#ifdef TUX
    userlog ("Error in TPC-C server %d: Failed to open %s\n",
          proc_no, filename);
#else
    fprintf (stderr, "Error in TPC-C server %d: Failed to open %s\n",
          proc_no, filename);
#endif
    return (-1);
  }

  OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(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_SERVER, errhp);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
                OCI_ATTR_PASSWORD, errhp);
  OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));

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

  /* run all transaction in serializable mode */

  OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT);
  OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT);
  OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
  OCIHandleFree(curi, OCI_HTYPE_STMT);

/*
This is done in cvdrv.c
  if (tracelevel == 2) {
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
    memset(stmbuf,0,100);
    sprintf ((char *) stmbuf, SQLTXTTRC);
    OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                                        OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
    OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
  }
*/
  if (tracelevel == 3) {
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
    memset(stmbuf,0,100);
    sprintf ((char *) stmbuf, SQLTXTTIM);
    OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                                        OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
    OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
  }

  logon = 1;

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

  if (tkvcninit ()) {  /* new order */
    TPCexit ();
    return (-1);
  }
  else
    new_init = 1;

  if (tkvcpinit ()) {   /* payment */
    TPCexit ();
    return (-1);
  }
  else
    pay_init = 1;

  if (tkvcoinit ()) {   /* order status */
    TPCexit ();
    return (-1);
  }
  else
    ord_init = 1;


  if (tkvcdinit (0)) {   /* delivery */
    TPCexit ();
    return (-1);
  }
  else
    del_init_oci = 1;

  if (tkvcdinit (1)) {   /* delivery */
    TPCexit ();
    return (-1);
  }
  else
    del_init_plsql = 1;

  if (tkvcsinit ()) {   /* stock level */
    TPCexit ();
    return (-1);
  }
  else
    sto_init = 1;

  return (0);

}



TPCnew (str)

struct newstruct *str;

{

  int i;

  w_id = str->newin.w_id;
  d_id = str->newin.d_id;
  c_id = str->newin.c_id;
  for (i = 0; i < 15; i++) {
    nol_i_id[i] = str->newin.ol_i_id[i];
    nol_supply_w_id[i] = str->newin.ol_supply_w_id[i];
    nol_quantity[i] = str->newin.ol_quantity[i];
  }
  retries = 0;

#ifndef AVOID_DEADLOCK

  for (i = NITEMS; i > 0; i--) {
    if (nol_i_id[i-1] > 0) {
      ordl_cnt = i;
      break;
    }
  }

  for (i = 0; i < NITEMS; i++) indx[i] = i;

  q_sort(nol_i_id, str, 0, ordl_cnt-1);

#endif

/*
  vgetdate(cr_date); */

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

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

  /* fill in date for o_entry_d from time in beginning of txn*/
/*
  cvtdmyhms(cr_date,o_entry_d);
*/
   datelen = sizeof(o_entry_d);
   OCIERROR(errhp,
      OCIDateToText(errhp,&cr_date,(text*)FULLDATE,SIZ(FULLDATE),(text*)0,0,
                &datelen,o_entry_d));

  str->newout.terror = NOERR;
  str->newout.o_id = o_id;
  str->newout.o_ol_cnt = o_ol_cnt;
  strncpy (str->newout.c_last, c_last, 17);
  strncpy (str->newout.c_credit, c_credit, 3);
  str->newout.c_discount = c_discount;
  str->newout.w_tax = (float)(w_tax);
  str->newout.d_tax = (float)(d_tax);
  strncpy (str->newout.o_entry_d, (char*)o_entry_d, 20);
  str->newout.total_amount = total_amount;
  for (i = 0; i < o_ol_cnt; i++) {
    strncpy (str->newout.i_name[i], i_name[i], 25);
    str->newout.s_quantity[i] = s_quantity[i];
    str->newout.brand_generic[i] = brand_generic[i][0];
    str->newout.i_price[i] = (float)(i_price[i])/100;
    str->newout.ol_amount[i] = (float)(nol_amount[i])/100;
  }
```

```
#ifndef AVOID_DEADLOCK
    q_sort(indx, str, 0, ordl_cnt-1);
#endif

   if (status)
     strcpy (str->newout.status, "Item number is not valid");
   else
     str->newout.status[0] = '\0';
   str->newout.retry = retries;
#if defined(TOP)  || defined(TUX)    /* changed mjb 17 feb for tuxedo */
   return(1);
#else
   return (0);
#endif

}



TPCpay (str)

struct paystruct *str;

{

   w_id = str->payin.w_id;
   d_id = str->payin.d_id;
   c_w_id = str->payin.c_w_id;
   c_d_id = str->payin.c_d_id;
   h_amount = str->payin.h_amount;
   bylastname = str->payin.bylastname;

/*
   vgetdate(cr_date);  */
   OCIERROR(errhp,OCIDateSysDate(errhp,&cr_date));

   if (bylastname) {
     c_id = 0;
     strncpy (c_last, str->payin.c_last, 17);
   }
   else {
     c_id = str->payin.c_id;
     strcpy (c_last, " ");
   }
   retries = 0;

   if (str->payout.terror = tkvcp ()) {
     if (str->payout.terror != RECOVERR)
       str->payout.terror = IRRECERR;
     return (-1);
   }

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

/*
   cvtdmy(c_since,c_since_d);
*/
       sincelen=SIZ(c_since_d);
       OCIERROR(errhp,OCIDateToText(errhp,&c_since,
         (text*)SHORTDATE,strlen(SHORTDATE),(text*)0,0,&sincelen,c_since_d));



   str->payout.terror = NOERR;
   strncpy (str->payout.w_street_1, w_street_1, 21);
   strncpy (str->payout.w_street_2, w_street_2, 21);
   strncpy (str->payout.w_city, w_city, 21);
   strncpy (str->payout.w_state, w_state, 3);
   strncpy (str->payout.w_zip, w_zip, 10);
   strncpy (str->payout.d_street_1, d_street_1, 21);
   strncpy (str->payout.d_street_2, d_street_2, 21);
   strncpy (str->payout.d_city, d_city, 21);
   strncpy (str->payout.d_state, d_state, 3);
   strncpy (str->payout.d_zip, d_zip, 10);
   str->payout.c_id = c_id;
   strncpy (str->payout.c_first, c_first, 17);
   strncpy (str->payout.c_middle, c_middle, 3);
   strncpy (str->payout.c_last, c_last, 17);
   strncpy (str->payout.c_street_1, c_street_1, 21);
   strncpy (str->payout.c_street_2, c_street_2, 21);
   strncpy (str->payout.c_city, c_city, 21);
   strncpy (str->payout.c_state, c_state, 3);
   strncpy (str->payout.c_zip, c_zip, 10);
   strncpy (str->payout.c_phone, c_phone, 17);
   strncpy (str->payout.c_since, (char*)c_since_d, 11);
   strncpy (str->payout.c_credit, c_credit, 3);
   str->payout.c_credit_lim = (float)(c_credit_lim)/100;
   str->payout.c_discount = c_discount;
   str->payout.c_balance = (float)(c_balance)/100;
   strncpy (str->payout.c_data, c_data, 201);
   strncpy (str->payout.h_date, (char*)h_date, 20);
   str->payout.retry = retries;
#if defined(TOP)  || defined(TUX)   /* changed mjb 17 Feb */
   return(1);
#else
   return (0);
#endif

}



TPCord (str)

struct ordstruct *str;

{

   int i;
   w_id = str->ordin.w_id;
   d_id = str->ordin.d_id;
   bylastname = str->ordin.bylastname;
   if (bylastname) {
     c_id = 0;
     strncpy (c_last, str->ordin.c_last, 17);
   }
   else {
     c_id = str->ordin.c_id;
     strcpy (c_last, " ");
   }
   retries = 0;

   if (str->ordout.terror = tkvco ()) {
     if (str->ordout.terror != RECOVERR)
       str->ordout.terror = IRRECERR;
     return (-1);
   }

   datelen = sizeof(o_entry_d);
   OCIERROR(errhp,
       OCIDateToText(errhp,&o_entry_d_base,(text*)FULLDATE,SIZ(FULLDATE),(text*)0,0,
                 &datelen,o_entry_d));

   str->ordout.terror = NOERR;
   str->ordout.c_id = c_id;
   strncpy (str->ordout.c_last, c_last, 17);
   strncpy (str->ordout.c_first, c_first, 17);
   strncpy (str->ordout.c_middle, c_middle, 3);
   str->ordout.c_balance = c_balance/100;
   str->ordout.o_id = o_id;
   strncpy (str->ordout.o_entry_d, (char*)o_entry_d, 20);
   if ( o_carrier_id == 11 )
      str->ordout.o_carrier_id = 0;
   else
      str->ordout.o_carrier_id = o_carrier_id;
   str->ordout.o_ol_cnt = o_ol_cnt;
   for (i = 0; i < o_ol_cnt; i++) {
     ol_delivery_d[i][10] = '\0';
     if ( !strcmp((char*)ol_delivery_d[i],"15-09-1911") )
                 strncpy((char*)ol_delivery_d[i],"NOT DELIVR",10);
     str->ordout.ol_supply_w_id[i] = ol_supply_w_id[i];
     str->ordout.ol_i_id[i] = ol_i_id[i];
     str->ordout.ol_quantity[i] = ol_quantity[i];
     str->ordout.ol_amount[i] = (float)(ol_amount[i])/100;
     strncpy (str->ordout.ol_delivery_d[i], (char*)ol_delivery_d[i], 11);
   }
   str->ordout.retry = retries;
#if defined(TOP)  || defined(TUX)
   return(1);
#else
   return (0);
#endif

}



TPCdel (str)

struct delstruct *str;

{

   double tr_end;
   int i;

   w_id = str->delin.w_id;
   o_carrier_id = str->delin.o_carrier_id;
   retries = 0;
/*
   vgetdate(cr_date);  */
   OCIERROR(errhp,OCIDateSysDate(errhp,&cr_date));

   if (str->delout.terror = tkvcd (str->delin.plsqlflag)) {
```

```
    if(str->delout.terror == DEL_ERROR)
      return DEL_ERROR;
    if (str->delout.terror != RECOVERR)
      str->delout.terror = IRRECERR;
    return (-1);
  }

  /* Comment out for the HP kit.
  tr_end = gettime ();
  fprintf (lfp, "%d %d %f %f %d %d", str->delin.in_timing_int,
        (tr_end - str->delin.qtime) <= DELRT ? 1 : 0,
        str->delin.qtime, tr_end, w_id, o_carrier_id);
  for (i = 0; i < 10; i++) {
    fprintf (lfp, " %d %d", i + 1, del_o_id[i]);
    if (del_o_id[i] <= 0) {
#ifdef TUX
      userlog ("DELIVERY: no new order for w_id: %d, d_id %d\n",
          w_id, i + 1);
#else
      fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
          w_id, i + 1);
#endif
    }
  }
  fprintf (lfp, " %d\n", retries);
  */

  str->delout.terror = NOERR;
  str->delout.retry = retries;
#if defined(TOP) || defined(TUX)   /* changed mjb 17 feb */
  return(1);
#else
  return (0);
#endif

}


TPCsto (str)

struct stostruct *str;

{

  w_id = str->stoin.w_id;
  d_id = str->stoin.d_id;
  threshold = str->stoin.threshold;
  retries = 0;

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

  str->stoout.terror = NOERR;
  str->stoout.low_stock = low_stock;
  str->stoout.retry = retries;
#if defined(TOP) || defined(TUX) /* changed mjb 17 feb */
  return(1);
#else
  return (0);
#endif

}

#ifndef AVOID_DEADLOCK

void q_sort(int *arr,struct newstruct *str,int left, int right)
{
 int i, last;

 if(left >= right)
   return;
 swap(str,left,(left+right)/2);
 last = left;
 for(i=left+1;i<=right;i++)
  if(arr[i] < arr[left])
    swap(str,last,i);
 swap(str,left,last);
 q_sort(arr,str,left,last-1);
 q_sort(arr,str,last+1,right);
}

void swap(struct newstruct *str, int i, int j)
{
 int temp;
 float tempf;
 char tmpstr[25];
 char tmpch;

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

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

 tmpch = str->newout.brand_generic[i];
 str->newout.brand_generic[i] = str->newout.brand_generic[j];

 str->newout.brand_generic[j] = tmpch;

 tempf = str->newout.i_price[i];
 str->newout.i_price[i] = str->newout.i_price[j];
 str->newout.i_price[j] = tempf;

 tempf = str->newout.ol_amount[i];
 str->newout.ol_amount[i] = str->newout.ol_amount[j];
 str->newout.ol_amount[j] = tempf;

}

#endif
```

client/oracle/plnew.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: tkvcnew.c 21-apr-98.18:32:59 rdecker Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*=========================================================================+
 |      Copyright (c) 1996 , 1997, 1998 Oracle Corp, Redwood Shores, CA    |
 |                OPEN SYSTEMS PERFORMANCE GROUP                           |
 |                   All Rights Reserved                                   |
 +=========================================================================+
 | FILENAME
 |   plnew.c
 | DESCRIPTION
 |   OCI version (using PL/SQL stored procedure) of
 |   NEW ORDER transaction in TPC-C benchmark.
 +=========================================================================*/

#ifndef ORA_TPCC
# define ORA_TPCC
# include "ora_tpcc.h"
#endif

#ifdef TUX
#include <userlog.h>
#endif

#ifndef ORA_TPCCFLAGS
# define ORA_TPCCFLAGS
# include "tpccflags.h"
#endif

extern void userlog();

#define SQLTXT2 "BEGIN inittpcc.init_no(:idx1arr); END;"

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

struct newctx {

  ub2 nol_i_id_len[NITEMS];
  ub2 nol_supply_w_id_len[NITEMS];
  ub2 nol_quantity_len[NITEMS];
  ub2 nol_amount_len[NITEMS];
  ub2 s_quantity_len[NITEMS];
  ub2 i_name_len[NITEMS];
  ub2 i_price_len[NITEMS];
  ub2 s_dist_info_len[NITEMS];
  ub2 ol_o_id_len[NITEMS];
  ub2 ol_number_len[NITEMS];
  ub2 s_remote_len[NITEMS];
```

```
  ub2 s_quant_len[NITEMS];
  ub2 ol_dist_info_len[NITEMS];
  ub2 s_bg_len[NITEMS];

  int ol_o_id[NITEMS];
  int ol_number[NITEMS];

  int s_remote[NITEMS];
  char s_dist_info[NITEMS][25];
  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;
  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;
  OCIStmt *curn2;
  OCIBind *ol_quantity_bp;
  OCIBind *s_remote_bp;
  OCIBind *s_quantity_bp;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *c_id_bp;
  OCIBind *o_all_local_bp;
  OCIBind *o_all_cnt_bp;
  OCIBind *w_tax_bp;
  OCIBind *d_tax_bp;
  OCIBind *o_id_bp;
  OCIBind *c_discount_bp;
  OCIBind *c_credit_bp;
  OCIBind *c_last_bp;
  OCIBind *retries_bp;
  OCIBind *cr_date_bp;
  OCIBind *ol_o_id_bp;
  OCIBind *ol_amount_bp;

  sb2 w_id_len;
  ub2 d_id_len;
  ub2 c_id_len;
  ub2 o_all_local_len;
  ub2 o_ol_cnt_len;
  ub2 w_tax_len;
  ub2 d_tax_len;
  ub2 o_id_len;
  ub2 c_discount_len;
  ub2 c_credit_len;
  ub2 c_last_len;
  ub2 retries_len;
  ub2 cr_date_len;
};

typedef struct newctx newctx;

static newctx *nctx;

tkvcninit ()
{

  int i;
  text stmbuf[16*1024];

  nctx = (newctx *) malloc (sizeof(newctx));
  DISCARD memset(nctx,(char)0,sizeof(newctx));
  nctx->w_id_len = sizeof(w_id);
  nctx->d_id_len = sizeof(d_id);
  nctx->c_id_len = sizeof(c_id);
  nctx->o_all_local_len = sizeof(o_all_local);
  nctx->o_ol_cnt_len = sizeof(o_ol_cnt);
  nctx->w_tax_len = 0;
  nctx->d_tax_len = 0;
  nctx->o_id_len = sizeof(o_id);
  nctx->c_discount_len = 0;
  nctx->c_credit_len = 0;
  nctx->c_last_len = 0;
  nctx->retries_len = sizeof(retries);
  nctx->cr_date_len = sizeof(cr_date);

  /* open first cursor */
  DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&nctx->curn1),
        OCI_HTYPE_STMT, 0, (dvoid**)0));
  sqlfile("../blocks/tkvcpnew.sql",stmbuf);
  DISCARD OCIERROR(errhp,OCIStmtPrepare(nctx->curn1, errhp, stmbuf,
        strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

  /* bind variables */

  OCIBNDPL(nctx->curn1, nctx->w_id_bp, errhp, ":w_id",ADR(w_id),SIZ(w_id),
        SQLT_INT, &nctx->w_id_len);
  OCIBNDPL(nctx->curn1, nctx->d_id_bp, errhp, ":d_id",ADR(d_id),SIZ(d_id),
        SQLT_INT, &nctx->d_id_len);
  OCIBNDPL(nctx->curn1, nctx->c_id_bp, errhp, ":c_id",ADR(c_id),SIZ(c_id),
        SQLT_INT, &nctx->c_id_len);
  OCIBNDPL(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_len);
  OCIBNDPL(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_len);
  OCIBNDPL(nctx->curn1, nctx->w_tax_bp, errhp, ":w_tax",ADR(w_tax),SIZ(w_tax),
        SQLT_FLT, &nctx->w_tax_len);
  OCIBNDPL(nctx->curn1, nctx->d_tax_bp, errhp, ":d_tax",ADR(d_tax),SIZ(d_tax),
        SQLT_FLT, &nctx->d_tax_len);
  OCIBNDPL(nctx->curn1, nctx->o_id_bp, errhp, ":o_id",ADR(o_id),SIZ(o_id),
        SQLT_INT, &nctx->o_id_len);
  OCIBNDPL(nctx->curn1, nctx->c_discount_bp, errhp, ":c_discount",
        ADR(c_discount), SIZ(c_discount),SQLT_FLT, &nctx->c_discount_len);
  OCIBNDPL(nctx->curn1, nctx->c_credit_bp, errhp, ":c_credit",c_credit,
        SIZ(c_credit),SQLT_CHR, &nctx->c_credit_len);
  OCIBNDPL(nctx->curn1, nctx->c_last_bp, errhp, ":c_last",c_last,SIZ(c_last),
        SQLT_STR, &nctx->c_last_len);
  OCIBNDPL(nctx->curn1, nctx->retries_bp, errhp, ":retry",ADR(retries),
        SIZ(retries),SQLT_INT, &nctx->retries_len);
  OCIBNDPL(nctx->curn1, nctx->cr_date_bp, errhp, ":cr_date",&cr_date,
        SIZ(OCIDate), SQLT_ODT, &nctx->cr_date_len);

  OCIBNDPLA(nctx->curn1, nctx->ol_i_id_bp,errhp,":ol_i_id",nol_i_id,
        SIZ(int), SQLT_INT, nctx->nol_i_id_len,NITEMS,&nctx->nol_i_count);
  OCIBNDPLA(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_len,
        NITEMS, &nctx->nol_s_count);
  OCIBNDPLA(nctx->curn1, nctx->ol_quantity_bp,errhp,":ol_quantity",
        nol_quantity, SIZ(int),SQLT_INT,nctx->nol_quantity_len,
        NITEMS,&nctx->nol_q_count);
  OCIBNDPLA(nctx->curn1, nctx->i_price_bp,errhp,":i_price",i_price,SIZ(int),
        SQLT_INT, nctx->i_price_len, NITEMS, &nctx->nol_item_count);
  OCIBNDPLA(nctx->curn1, nctx->i_name_bp,errhp,":i_name",i_name,
        SIZ(i_name[0]),SQLT_STR, nctx->i_name_len,NITEMS,
        &nctx->nol_name_count);
  OCIBNDPLA(nctx->curn1, nctx->s_quantity_bp,errhp,":s_quantity",s_quantity,
        SIZ(int), SQLT_INT,nctx->s_quant_len,NITEMS,&nctx->nol_qty_count);
  OCIBNDPLA(nctx->curn1, nctx->s_bg_bp,errhp,":brand_generic",brand_generic,
        SIZ(char), SQLT_CHR,nctx->s_bg_len,NITEMS,&nctx->nol_bg_count);
  OCIBNDPLA(nctx->curn1, nctx->ol_amount_bp,errhp,":ol_amount",nol_amount,
        SIZ(int),SQLT_INT, nctx->nol_amount_len,NITEMS,&nctx->nol_am_count);
  OCIBNDPLA(nctx->curn1, nctx->s_remote_bp,errhp,":s_remote",nctx->s_remote,
        SIZ(int),SQLT_INT, nctx->s_remote_len,NITEMS,&nctx->s_remote_count);

  /* open second cursor */
  DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&nctx->curn2),
            OCI_HTYPE_STMT, 0, (dvoid**)0));
  DISCARD sprintf ((char *) stmbuf, SQLTXT2);
  DISCARD 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 */
   OCIBNDPLA(nctx->curn2, idx1arr_bp,errhp,":idx1arr",idx1arr,
         SIZ(int), SQLT_INT, idx1arr_len, NITEMS,&idx1arr_count);

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

  return (0);

}


tkvcn ()
```

```
{

  int i;
  int rcount;

retry:

  status = 0;                   /* number of invalid items */

  /* get number of order lines, and check if all are local */

  o_ol_cnt = NITEMS;
  o_all_local = 1;
  for (i = 0; i < NITEMS; i++) {
    if (nol_i_id[i] == 0) {
      o_ol_cnt = i;
      break;
    }
    if (nol_supply_w_id[i] != w_id) {
      nctx->s_remote[i] = 1;
      o_all_local = 0;
    }
    else
      nctx->s_remote[i] = 0;
  }

  nctx->w_id_len = sizeof(w_id);
  nctx->d_id_len = sizeof(d_id);
  nctx->c_id_len = sizeof(c_id);
  nctx->o_all_local_len = sizeof(o_all_local);
  nctx->o_ol_cnt_len = sizeof(o_ol_cnt);
  nctx->w_tax_len = 0;
  nctx->d_tax_len = 0;
  nctx->o_id_len = sizeof(o_id);
  nctx->c_discount_len = 0;
  nctx->c_credit_len = 0;
  nctx->c_last_len = 0;
  nctx->retries_len = sizeof(retries);
  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;

  /* initialization for array operations */
  for (i = 0; i < o_ol_cnt; i++) {
    nctx->ol_number[i] = i + 1;
    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_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->s_remote_len[i] = sizeof(int);
    nctx->s_quant_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_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_o_id_len[i] = 0;
    nctx->ol_number_len[i] = 0;
    nctx->ol_dist_info_len[i] = 0;
    nctx->s_remote_len[i] = 0;
    nctx->s_quant_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 if (errcode == SNAPSHOT_TOO_OLD) {
                  retries++;
                  goto retry;
    } else {
                  return -1;
    }
  }

  /* did the txn succeed ? */
  if (rcount != o_ol_cnt)
  {
    status = rcount - o_ol_cnt;
    o_ol_cnt = rcount;
  }

#ifdef DEBUG
  printf("w_id = %d, d_id = %d, c_id = %d\n",w_id, d_id, c_id);
#endif


  return (0);

}



void tkvcndone ()

{

  int i;

  if (nctx)
  {
    DISCARD OCIHandleFree((dvoid *)nctx->curn1,OCI_HTYPE_STMT);
    DISCARD OCIHandleFree((dvoid *)nctx->curn2,OCI_HTYPE_STMT);
    free (nctx);
  }
}
```

client/oracle/plpay.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: plpay.c 7030100.1 95/07/19 14:44:59 plai Generic<base> $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

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

#include "ora_tpcc.h"
#include "tpccflags.h"

#ifdef TUX
#include <userlog.h>
#endif


#define SQLTXT_INIT "BEGIN inittpcc.init_pay; END;"

struct payctx {
  OCIStmt *curpi;
  OCIStmt *curp0;
  OCIStmt *curp1;
  OCIBind *w_id_bp[2];
  ub2 w_id_len;

  OCIBind *d_id_bp[2];
  ub2 d_id_len;

  OCIBind *c_w_id_bp[2];
  ub2 c_w_id_len;

  OCIBind *c_d_id_bp[2];
  ub2 c_d_id_len;

  OCIBind *c_id_bp[2];
  ub2 c_id_len;

  OCIBind *h_amount_bp[2];
  ub2 h_amount_len;
```

```
  OCIBind *c_last_bp[2];
  ub2 c_last_len;

  OCIBind *w_street_1_bp[2];
  ub2 w_street_1_len;

  OCIBind *w_street_2_bp[2];
  ub2 w_street_2_len;

  OCIBind *w_city_bp[2];
  ub2 w_city_len;

  OCIBind *w_state_bp[2];
  ub2 w_state_len;

  OCIBind *w_zip_bp[2];
  ub2 w_zip_len;

  OCIBind *d_street_1_bp[2];
  ub2 d_street_1_len;

  OCIBind *d_street_2_bp[2];
  ub2 d_street_2_len;

  OCIBind *d_city_bp[2];
  ub2 d_city_len;

  OCIBind *d_state_bp[2];
  ub2 d_state_len;

  OCIBind *d_zip_bp[2];
  ub2 d_zip_len;

  OCIBind *c_first_bp[2];
  ub2 c_first_len;

  OCIBind *c_middle_bp[2];
  ub2 c_middle_len;

  OCIBind *c_street_1_bp[2];
  ub2 c_street_1_len;

  OCIBind *c_street_2_bp[2];
  ub2 c_street_2_len;

  OCIBind *c_city_bp[2];
  ub2 c_city_len;

  OCIBind *c_state_bp[2];
  ub2 c_state_len;

  OCIBind *c_zip_bp[2];
  ub2 c_zip_len;

  OCIBind *c_phone_bp[2];
  ub2 c_phone_len;

  OCIBind *c_since_bp[2];
  ub2 c_since_len;

  OCIBind *c_credit_bp[2];
  ub2 c_credit_len;

  OCIBind *c_credit_lim_bp[2];
  ub2 c_credit_lim_len;

  OCIBind *c_discount_bp[2];
  ub2 c_discount_len;

  OCIBind *c_balance_bp[2];
  ub2 c_balance_len;

  OCIBind *c_data_bp[2];
  ub2 c_data_len;

  OCIBind *h_date_bp[2];
  ub2 h_date_len;

  OCIBind *retries_bp[2];
  ub2 retries_len;

  OCIBind *cr_date_bp[2];
  ub2 cr_date_len;

  OCIBind *byln_bp[2];
  ub2 byln_len;
};

typedef struct payctx payctx;



payctx *pctx;
```

```
int tkvcpinit (void)

{

  text stmbuf[SQL_BUF_SIZE];
  pctx = (payctx *)malloc(sizeof(payctx));
  memset(pctx,(char)0,sizeof(payctx));

/* cursor for init */
  DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curpi)),
        OCI_HTYPE_STMT,0,(dvoid**)0));

  DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curp0)),
        OCI_HTYPE_STMT,0,(dvoid**)0));
  DISCARD 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);
  DISCARD OCIERROR(errhp,OCIStmtPrepare(pctx->curpi, errhp, stmbuf,
        strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
  DISCARD OCIERROR(errhp, OCIStmtExecute(tpcsvc,pctx->curpi,errhp,1,0,
              NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT));

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

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

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

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

  pctx->w_id_len = SIZ(w_id);
  pctx->d_id_len = SIZ(d_id);
  pctx->c_w_id_len = SIZ(c_w_id);
  pctx->c_d_id_len = SIZ(c_d_id);
  pctx->c_id_len = 0;
  pctx->h_amount_len = SIZ(h_amount);
  pctx->c_last_len = 0;
  pctx->w_street_1_len = 0;
  pctx->w_street_2_len = 0;
  pctx->w_city_len = 0;
  pctx->w_state_len = 0;
  pctx->w_zip_len = 0;
  pctx->d_street_1_len = 0;
  pctx->d_street_2_len = 0;
  pctx->d_city_len = 0;
  pctx->d_state_len = 0;
  pctx->d_zip_len = 0;
  pctx->c_first_len = 0;
  pctx->c_middle_len = 0;
  pctx->c_street_1_len = 0;
  pctx->c_street_2_len = 0;
  pctx->c_city_len = 0;
  pctx->c_state_len = 0;
  pctx->c_zip_len = 0;
  pctx->c_phone_len = 0;
  pctx->c_since_len = 0;
  pctx->c_credit_len = 0;
  pctx->c_credit_lim_len = 0;
  pctx->c_discount_len = 0;
  pctx->c_balance_len = sizeof(double);
  pctx->c_data_len = 0;
  pctx->h_date_len = 0;
  pctx->retries_len =SIZ(retries) ;
  pctx->cr_date_len = 7;



  /* bind variables */


  OCIBNDPL(pctx->curp0, pctx->w_id_bp[0], errhp,":w_id",ADR(w_id),SIZ(int),
        SQLT_INT, NULL);
  OCIBNDPL(pctx->curp0, pctx->d_id_bp[0], errhp,":d_id",ADR(d_id),SIZ(int),
        SQLT_INT, NULL);
  OCIBND(pctx->curp0, pctx->c_w_id_bp[0], errhp,":c_w_id",ADR(c_w_id),SIZ(int),
        SQLT_INT);
  OCIBND(pctx->curp0, pctx->c_d_id_bp[0], errhp,":c_d_id",ADR(c_d_id),SIZ(int),
        SQLT_INT);
  OCIBND(pctx->curp0, pctx->c_id_bp[0], errhp,":c_id",ADR(c_id),SIZ(int),
        SQLT_INT);
  OCIBNDPL(pctx->curp0, pctx->h_amount_bp[0], errhp,":h_amount",ADR(h_amount),
        SIZ(int),SQLT_INT,  &pctx->h_amount_len);
  OCIBNDPL(pctx->curp0, pctx->c_last_bp[0], errhp,":c_last",c_last,SIZ(c_last),
        SQLT_STR, &pctx->c_last_len);
  OCIBNDPL(pctx->curp0, pctx->w_street_1_bp[0], errhp,":w_street_1",w_street_1,
        SIZ(w_street_1),SQLT_STR, &pctx->w_street_1_len);
  OCIBNDPL(pctx->curp0, pctx->w_street_2_bp[0], errhp,":w_street_2",w_street_2,
```

```
        SIZ(w_street_2),SQLT_STR, &pctx->w_street_2_len);
    OCIBNDPL(pctx->curp0, pctx->w_city_bp[0], errhp,":w_city",w_city,SIZ(w_city),
        SQLT_STR, &pctx->w_city_len);
    OCIBNDPL(pctx->curp0, pctx->w_state_bp[0], errhp,":w_state",w_state,
        SIZ(w_state), SQLT_STR, &pctx->w_state_len);
    OCIBNDPL(pctx->curp0, pctx->w_zip_bp[0], errhp,":w_zip",w_zip,SIZ(w_zip),
        SQLT_STR, &pctx->w_zip_len);
    OCIBNDPL(pctx->curp0, pctx->d_street_1_bp[0], errhp,":d_street_1",d_street_1,
        SIZ(d_street_1),SQLT_STR, &pctx->d_street_1_len);
    OCIBNDPL(pctx->curp0, pctx->d_street_2_bp[0], errhp,":d_street_2",d_street_2,
        SIZ(d_street_2),SQLT_STR, &pctx->d_street_2_len);
    OCIBNDPL(pctx->curp0, pctx->d_city_bp[0], errhp,":d_city",d_city,SIZ(d_city),
        SQLT_STR, &pctx->d_city_len);
    OCIBNDPL(pctx->curp0, pctx->d_state_bp[0], errhp,":d_state",d_state,
        SIZ(d_state), SQLT_STR, &pctx->d_state_len);
    OCIBNDPL(pctx->curp0, pctx->d_zip_bp[0], errhp,":d_zip",d_zip,SIZ(d_zip),
        SQLT_STR, &pctx->d_zip_len);
    OCIBNDPL(pctx->curp0, pctx->c_first_bp[0], errhp,":c_first",c_first,
        SIZ(c_first), SQLT_STR, &pctx->c_first_len);
    OCIBNDPL(pctx->curp0, pctx->c_middle_bp[0], errhp,":c_middle",c_middle,2,
        SQLT_AFC, &pctx->c_middle_len);
    OCIBNDPL(pctx->curp0, pctx->c_street_1_bp[0], errhp,":c_street_1",c_street_1,
        SIZ(c_street_1),SQLT_STR, &pctx->c_street_1_len);
    OCIBNDPL(pctx->curp0, pctx->c_street_2_bp[0], errhp,":c_street_2",c_street_2,
        SIZ(c_street_2),SQLT_STR, &pctx->c_street_2_len);
    OCIBNDPL(pctx->curp0, pctx->c_city_bp[0], errhp,":c_city",c_city,SIZ(c_city),
        SQLT_STR, &pctx->c_city_len);
    OCIBNDPL(pctx->curp0, pctx->c_state_bp[0], errhp,":c_state",c_state,
        SIZ(c_state), SQLT_STR, &pctx->c_state_len);
    OCIBNDPL(pctx->curp0, pctx->c_zip_bp[0], errhp,":c_zip",c_zip,SIZ(c_zip),
        SQLT_STR,&pctx->c_zip_len);
    OCIBNDPL(pctx->curp0, pctx->c_phone_bp[0], errhp,":c_phone",c_phone,
        SIZ(c_phone), SQLT_STR, &pctx->c_phone_len);
    OCIBNDPL(pctx->curp0, pctx->c_since_bp[0], errhp,":c_since",&c_since,
        SIZ(OCIDate), SQLT_ODT, &pctx->c_since_len);
    OCIBNDPL(pctx->curp0, pctx->c_credit_bp[0], errhp,":c_credit",c_credit,
        SIZ(c_credit),SQLT_CHR, &pctx->c_credit_len);
    OCIBNDPL(pctx->curp0, pctx->c_credit_lim_bp[0], errhp,":c_credit_lim",
        ADR(c_credit_lim),SIZ(int), SQLT_INT, &pctx->c_credit_lim_len);
    OCIBNDPL(pctx->curp0, pctx->c_discount_bp[0], errhp,":c_discount",
        ADR(c_discount),SIZ(c_discount), SQLT_FLT, &pctx->c_discount_len);
    OCIBNDPL(pctx->curp0, pctx->c_balance_bp[0], errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_FLT, &pctx->c_balance_len);
    OCIBNDPL(pctx->curp0, pctx->c_data_bp[0], errhp,":c_data",c_data,SIZ(c_data),
        SQLT_STR, &pctx->c_data_len);
/*
    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);
*/
    OCIBNDPL(pctx->curp0, pctx->retries_bp[0], errhp,":retry",ADR(retries),
        SIZ(int), SQLT_INT, &pctx->retries_len);
    OCIBNDPL(pctx->curp0, pctx->cr_date_bp[0], errhp,":cr_date",ADR(cr_date),
        SIZ(OCIDate),SQLT_ODT, &pctx->cr_date_len);

/* ---- Binds for  the second cursor   */


    OCIBNDPL(pctx->curp1, pctx->w_id_bp[1], errhp,":w_id",ADR(w_id),SIZ(int),
        SQLT_INT, &pctx->w_id_len);
    OCIBNDPL(pctx->curp1, pctx->d_id_bp[1], errhp,":d_id",ADR(d_id),SIZ(int),
        SQLT_INT, &pctx->d_id_len);
    OCIBND(pctx->curp1, pctx->c_w_id_bp[1], errhp,":c_w_id",ADR(c_w_id),SIZ(int),
        SQLT_INT);
    OCIBND(pctx->curp1, pctx->c_d_id_bp[1], errhp,":c_d_id",ADR(c_d_id),SIZ(int),
        SQLT_INT);
    OCIBNDPL(pctx->curp1, pctx->c_id_bp[1], errhp,":c_id",ADR(c_id),SIZ(int),
        SQLT_INT, &pctx->c_id_len);
    OCIBNDPL(pctx->curp1, pctx->h_amount_bp[1], errhp,":h_amount",ADR(h_amount),
        SIZ(int),SQLT_INT, &pctx->h_amount_len);
    OCIBND(pctx->curp1, pctx->c_last_bp[1], errhp,":c_last",c_last,SIZ(c_last),
        SQLT_STR);
    OCIBNDPL(pctx->curp1, pctx->w_street_1_bp[1], errhp,":w_street_1",w_street_1,
        SIZ(w_street_1),SQLT_STR, &pctx->w_street_1_len);
    OCIBNDPL(pctx->curp1, pctx->w_street_2_bp[1], errhp,":w_street_2",w_street_2,
        SIZ(w_street_2),SQLT_STR, &pctx->w_street_2_len);
    OCIBNDPL(pctx->curp1, pctx->w_city_bp[1], errhp,":w_city",w_city,SIZ(w_city),
        SQLT_STR, &pctx->w_city_len);
    OCIBNDPL(pctx->curp1, pctx->w_state_bp[1], errhp,":w_state",w_state,
        SIZ(w_state), SQLT_STR, &pctx->w_state_len);
    OCIBNDPL(pctx->curp1, pctx->w_zip_bp[1], errhp,":w_zip",w_zip,SIZ(w_zip),
        SQLT_STR, &pctx->w_zip_len);
    OCIBNDPL(pctx->curp1, pctx->d_street_1_bp[1], errhp,":d_street_1",d_street_1,
        SIZ(d_street_1),SQLT_STR, &pctx->d_street_1_len);
    OCIBNDPL(pctx->curp1, pctx->d_street_2_bp[1], errhp,":d_street_2",d_street_2,
        SIZ(d_street_2),SQLT_STR, &pctx->d_street_2_len);
    OCIBNDPL(pctx->curp1, pctx->d_city_bp[1], errhp,":d_city",d_city,SIZ(d_city),
        SQLT_STR, &pctx->d_city_len);
    OCIBNDPL(pctx->curp1, pctx->d_state_bp[1], errhp,":d_state",d_state,
        SIZ(d_state), SQLT_STR, &pctx->d_state_len);
    OCIBNDPL(pctx->curp1, pctx->d_zip_bp[1], errhp,":d_zip",d_zip,SIZ(d_zip),
        SQLT_STR, &pctx->d_zip_len);
    OCIBNDPL(pctx->curp1, pctx->c_first_bp[1], errhp,":c_first",c_first,
        SIZ(c_first), SQLT_STR, &pctx->c_first_len);
    OCIBNDPL(pctx->curp1, pctx->c_middle_bp[1], errhp,":c_middle",c_middle,2,
        SQLT_AFC, &pctx->c_middle_len);
    OCIBNDPL(pctx->curp1, pctx->c_street_1_bp[1], errhp,":c_street_1",c_street_1,
        SIZ(c_street_1),SQLT_STR, &pctx->c_street_1_len);
    OCIBNDPL(pctx->curp1, pctx->c_street_2_bp[1], errhp,":c_street_2",c_street_2,
        SIZ(c_street_2),SQLT_STR, &pctx->c_street_2_len);
    OCIBNDPL(pctx->curp1, pctx->c_city_bp[1], errhp,":c_city",c_city,
        SIZ(c_city),SQLT_STR, &pctx->c_city_len);
    OCIBNDPL(pctx->curp1, pctx->c_state_bp[1], errhp,":c_state",c_state,
        SIZ(c_state), SQLT_STR, &pctx->c_state_len);
    OCIBNDPL(pctx->curp1, pctx->c_zip_bp[1], errhp,":c_zip",c_zip,SIZ(c_zip),
        SQLT_STR, &pctx->c_zip_len);
    OCIBNDPL(pctx->curp1, pctx->c_phone_bp[1], errhp,":c_phone",c_phone,
        SIZ(c_phone), SQLT_STR, &pctx->c_phone_len);
    OCIBNDPL(pctx->curp1, pctx->c_since_bp[1], errhp,":c_since",&c_since,
        SIZ(OCIDate), SQLT_ODT, &pctx->c_since_len);
    OCIBNDPL(pctx->curp1, pctx->c_credit_bp[1], errhp,":c_credit",c_credit,
        SIZ(c_credit),SQLT_CHR, &pctx->c_credit_len);
    OCIBNDPL(pctx->curp1, pctx->c_credit_lim_bp[1], errhp,":c_credit_lim",
        ADR(c_credit_lim),SIZ(int), SQLT_INT, &pctx->c_credit_lim_len);
    OCIBNDPL(pctx->curp1, pctx->c_discount_bp[1], errhp,":c_discount",
        ADR(c_discount),SIZ(c_discount), SQLT_FLT, &pctx->c_discount_len);
    OCIBNDPL(pctx->curp1, pctx->c_balance_bp[1], errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_FLT, &pctx->c_balance_len);
    OCIBNDPL(pctx->curp1, pctx->c_data_bp[1], errhp,":c_data",c_data,SIZ(c_data),
        SQLT_STR, &pctx->c_data_len);
/*
    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);
*/
    OCIBNDPL(pctx->curp1, pctx->retries_bp[1], errhp,":retry",ADR(retries),
        SIZ(int), SQLT_INT, &pctx->retries_len);
    OCIBNDPL(pctx->curp1, pctx->cr_date_bp[1], errhp,":cr_date",ADR(cr_date),
        SIZ(OCIDate),SQLT_ODT, &pctx->cr_date_len);

    return (0);

}



tkvcp ()

{

retry:

    pctx->w_id_len = SIZ(w_id);
    pctx->d_id_len = SIZ(d_id);
    pctx->c_w_id_len = 0;
    pctx->c_d_id_len = 0;
    pctx->c_id_len = 0;
    pctx->h_amount_len = SIZ(h_amount);
    pctx->c_last_len = SIZ(c_last);
    pctx->w_street_1_len = 0;
    pctx->w_street_2_len = 0;
    pctx->w_city_len = 0;
    pctx->w_state_len = 0;
    pctx->w_zip_len = 0;
    pctx->d_street_1_len = 0;
    pctx->d_street_2_len = 0;
    pctx->d_city_len = 0;
    pctx->d_state_len = 0;
    pctx->d_zip_len = 0;
    pctx->c_first_len = 0;
    pctx->c_middle_len = 0;
    pctx->c_street_1_len = 0;
    pctx->c_street_2_len = 0;
    pctx->c_city_len = 0;
    pctx->c_state_len = 0;
    pctx->c_zip_len = 0;
    pctx->c_phone_len = 0;
    pctx->c_since_len = 0;
    pctx->c_credit_len = 0;
    pctx->c_credit_lim_len = 0;
    pctx->c_discount_len = 0;
    pctx->c_balance_len = sizeof(double);
    pctx->c_data_len = 0;
    pctx->h_date_len = 0;
    pctx->retries_len = SIZ(retries);
    pctx->cr_date_len = 7;

    if(bylastname) {
     execstatus=OCIStmtExecute(tpcsvc,pctx->curp1,errhp,1,0,
                NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
                OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
    } else {
     execstatus=OCIStmtExecute(tpcsvc,pctx->curp0,errhp,1,0,
                NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
                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 if (errcode == SNAPSHOT_TOO_OLD) {
      retries++;
      goto retry;
    } else {
      return -1;
    }
  }
  return 0;
}



void tkvcpdone ()

{
 if(pctx) {
   free(pctx);
  }
}
```

client/oracle/plord.c

```
/* Copyright (c) 2002, Oracle Corporation.  All rights reserved.  */

/*

  NAME
    tkvcordq.c - OCI version  using queues of ORDER STATUS
                 transaction in TPC-C benchmark.

  DESCRIPTION
    <short description of facility this file declares/defines>

  EXPORT FUNCTION(S)

  INTERNAL FUNCTION(S)
    <other external functions defined - one-line descriptions>

  STATIC FUNCTION(S)
    <static functions defined - one-line descriptions>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  xnie       06/25/02 - queue open cluster join.
  heri       05/07/02 - Fix error in cursor.
  heri       02/01/02 - Cleanup, remove indicator values and return codes.
  lwang      07/25/01 - Merged lwang_tpccitrc
  lwang      07/23/01 - fix include
  lwang      07/23/01 - Creation

*/


#include "ora_tpcc.h"
#include "tpccflags.h"

/*---------------------------------------------------------------------------
                    PRIVATE TYPES AND CONSTANTS
  ---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------
                    STATIC FUNCTION DECLARATIONS
  ---------------------------------------------------------------------------*/



#define SQLCUR0 "SELECT rowid FROM cust \
                 WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last = :c_last \
                 ORDER BY c_last, c_d_id, c_w_id, c_first"

#define SQLCUR1 "SELECT /*+ USE_NL(cust) INDEX_DESC(ordr iordr2) */  \
                 c_id, c_balance, c_first, c_middle, c_last, \
                 o_id, o_entry_d, o_carrier_id, o_ol_cnt, ordr.rowid \
                 FROM cust, ordr \
                 WHERE cust.rowid = :cust_rowid \
                 AND   o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id = c_id \
                 ORDER BY o_c_id DESC, o_d_id DESC, o_w_id DESC, o_id DESC"


#define SQLCUR2 "SELECT /*+ USE_NL(cust)  INDEX_DESC (ordr iordr2) */   \
                 c_balance, c_first, c_middle, c_last, \
                 o_id, o_entry_d, o_carrier_id, o_ol_cnt, ordr.rowid \
                 FROM cust, ordr \
                 WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id \
                 AND o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id = c_id \
                 ORDER BY o_c_id DESC, o_d_id DESC, o_w_id DESC , o_id DESC"


#define SQLCUR3 "SELECT  /*+ ORDERED USE_NL(ordl) CLUSTER(ordl) */ \
                 ol_i_id, ol_supply_w_id, ol_quantity, ol_amount, ol_delivery_d \
                 FROM ordr, ordl \
                 WHERE ordr.rowid = :ordr_rowid \
                  AND o_id = ol_o_id AND  ol_d_id = o_d_id AND ol_w_id = o_w_id"

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


struct ordctx {

  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;


  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;
  OCIStmt *curo4;
  OCIBind *c_id_bp;
  OCIBind *w_id_bp[4];
  OCIBind *d_id_bp[4];
  OCIBind *c_last_bp[2];
  OCIBind *o_id_bp;
  OCIBind *c_rowid_bp;
  OCIBind *o_rowid_bp;
  OCIDefine *c_rowid_dp;
  OCIDefine *c_last_dp[2];
  OCIDefine *c_id_dp;
  OCIDefine *c_first_dp[2];
  OCIDefine *c_middle_dp[2];
  OCIDefine *c_balance_dp[2];
  OCIDefine *o_rowid_dp[2];
  OCIDefine *o_id_dp[2];
  OCIDefine *o_entry_d_dp[2];
  OCIDefine *o_cr_id_dp[2];
  OCIDefine *o_ol_cnt_dp[2];
  OCIDefine *ol_d_d_dp;
  OCIDefine *ol_i_id_dp;
  OCIDefine *ol_supply_w_id_dp;
  OCIDefine *ol_quantity_dp;
  OCIDefine *ol_amount_dp;
  OCIDefine *ol_d_base_dp;
  OCIDefine *c_count_dp;
  OCIRowid *c_rowid_ptr[100];
  OCIRowid *c_rowid_cust;
  OCIRowid *o_rowid;
  int cs;
  int cust_idx;
  int norow;
  int rcount;
  int somerows;
};

typedef struct ordctx ordctx;

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

static ordctx *octx;

static defctx cbctx;


tkvcoinit ()

{

  int i;
```

```
  text stmbuf[SQL_BUF_SIZE];

  octx = (ordctx *) malloc (sizeof(ordctx));
  DISCARD memset(octx,(char)0,sizeof(ordctx));
  octx->cs = 1;
  octx->norow = 0;
  octx->somerows = 10;
/* get the rowid handles */
  OCIERROR(errhp, OCIDescriptorAlloc((dvoid *)tpcenv,(dvoid **)&octx->o_rowid,
              (ub4)OCI_DTYPE_ROWID, (size_t) 0, (dvoid **)0));
  for(i=0;i<100;i++) {
   DISCARD OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,
            (dvoid**)&octx->c_rowid_ptr[i], OCI_DTYPE_ROWID,0,(dvoid**)0));
  }


  DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
  DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
  DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo1,OCI_HTYPE_STMT,0,(dvoid**)0));
  DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo2,OCI_HTYPE_STMT,0,(dvoid**)0));
  DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo3,OCI_HTYPE_STMT,0,(dvoid**)0));
  DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo4,OCI_HTYPE_STMT,0,(dvoid**)0));

/*  c_id = 0, use find customer by lastname. Get an array or rowid's back*/
  DISCARD sprintf(char *) stmbuf, SQLCUR0);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo0,errhp,stmbuf,(ub4)strlen((char *)stmbuf),
                    OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo0,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));
/* get order/customer info back based on rowid */
  DISCARD sprintf(char *) stmbuf, SQLCUR1);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo1,errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo1,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));

/*  c_id == 0, use lastname to find customer */
  DISCARD sprintf(char *) stmbuf, SQLCUR2);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo2,errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo2,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));

  DISCARD sprintf(char *) stmbuf, SQLCUR3);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo3,errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo3,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));

  DISCARD sprintf(char *) stmbuf, SQLCUR4);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo4,errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo4,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));

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

    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_len = sizeof(int);
  octx->ol_d_id_len = sizeof(int);
  octx->ol_o_id_len = sizeof(int);

  /* bind variables */

  /* c_id (customer id) is not known */
  OCIBND(octx->curo0,octx->w_id_bp[0],errhp,":w_id",ADR(w_id),
          SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->d_id_bp[0],errhp,":d_id",ADR(d_id),
          SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->c_last_bp[0],errhp,":c_last",c_last,
          SIZ(c_last), SQLT_STR);
  OCIDFNRA(octx->curo0,octx->c_rowid_dp,errhp,1,octx->c_rowid_ptr,
            SIZ(OCIRowid*), SQLT_RDD, NULL, octx->c_rowid_len, NULL);

  OCIBND(octx->curo1,octx->c_rowid_bp,errhp,":cust_rowid", &octx->c_rowid_cust,
          sizeof( octx->c_rowid_ptr[0]),SQLT_RDD);
  OCIDEF(octx->curo1,octx->c_id_dp,errhp,1,ADR(c_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->c_balance_dp[0],errhp,2,ADR(c_balance),
          SIZ(double),SQLT_FLT);
  OCIDEF(octx->curo1,octx->c_first_dp[0],errhp,3,c_first,SIZ(c_first)-1,
          SQLT_CHR);
  OCIDEF(octx->curo1,octx->c_middle_dp[0],errhp,4,c_middle,
          SIZ(c_middle)-1,SQLT_AFC);
  OCIDEF(octx->curo1,octx->c_last_dp[0],errhp,5,c_last,SIZ(c_last)-1,
          SQLT_CHR);
  OCIDEF(octx->curo1,octx->o_id_dp[0],errhp,6,ADR(o_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_entry_d_dp[0],errhp,7,
          &o_entry_d_base,SIZ(OCIDate),SQLT_ODT);
  OCIDEF(octx->curo1,octx->o_cr_id_dp[0],errhp,8,ADR(o_carrier_id),
          SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_ol_cnt_dp[0],errhp,9,ADR(o_ol_cnt),
          SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_rowid_dp[0],errhp,10,ADR(octx->o_rowid),
          SIZ(OCIRowid*),SQLT_RDD);


/* Bind for third cursor , no-zero customer id */
  OCIBND(octx->curo2,octx->w_id_bp[1],errhp,":w_id",ADR(w_id),
          SIZ(int),SQLT_INT);
  OCIBND(octx->curo2,octx->d_id_bp[1],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_dp[1],errhp,1,ADR(c_balance),
                  SIZ(double),SQLT_FLT);
  OCIDEF(octx->curo2,octx->c_first_dp[1],errhp,2,c_first,SIZ(c_first)-1,
          SQLT_CHR);
  OCIDEF(octx->curo2,octx->c_middle_dp[1],errhp,3,c_middle,
          SIZ(c_middle)-1,SQLT_AFC);
  OCIDEF(octx->curo2,octx->c_last_dp[1],errhp,4,c_last,SIZ(c_last)-1,
          SQLT_CHR);
  OCIDEF(octx->curo2,octx->o_id_dp[1],errhp,5,ADR(o_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo2,octx->o_entry_d_dp[1],errhp,6, &o_entry_d_base,
          SIZ(OCIDate),SQLT_ODT);
  OCIDEF(octx->curo2, octx->o_cr_id_dp[1],errhp,7,ADR(o_carrier_id),
          SIZ(int), SQLT_INT);
  OCIDEF(octx->curo2,octx->o_ol_cnt_dp[1],errhp,8,ADR(o_ol_cnt),
          SIZ(int),SQLT_INT);
  OCIDEF(octx->curo2,octx->o_rowid_dp[1],errhp,9,ADR(octx->o_rowid),
          SIZ(OCIRowid*),SQLT_RDD);

/* Bind for last cursor */

/*
  OCIBND(octx->curo3,octx->w_id_bp[2],errhp,":w_id",ADR(w_id), SIZ(int),SQLT_INT);
  OCIBND(octx->curo3,octx->d_id_bp[2],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);
  OCIBND(octx->curo3,octx->c_id_bp,errhp,":c_id",ADR(c_id), SIZ(int),SQLT_INT);
 */
  OCIBND(octx->curo3,octx->o_rowid_bp,errhp,":ordr_rowid",
        &octx->o_rowid, SIZ(OCIRowid*),SQLT_RDD);

  OCIDFNRA(octx->curo3, octx->ol_i_id_dp, errhp, 1, ol_i_id,SIZ(int),SQLT_INT,
                  NULL,octx->ol_i_id_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_supply_w_id_dp,errhp,2, ol_supply_w_id,
          SIZ(int),SQLT_INT, NULL,
          octx->ol_supply_w_id_len, NULL);
  OCIDFNRA(octx->curo3, octx->ol_quantity_dp,errhp,3, ol_quantity,SIZ(int),
          SQLT_INT, NULL,octx->ol_quantity_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_amount_dp,errhp,4,ol_amount, SIZ(int),
          SQLT_INT,NULL, octx->ol_amount_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_d_base_dp,errhp,5,ol_d_base,SIZ(OCIDate),
          SQLT_ODT, NULL,octx->ol_delivery_d_len,NULL);

  OCIBND(octx->curo4,octx->w_id_bp[3],errhp,":w_id",ADR(w_id),
          SIZ(int),SQLT_INT);
  OCIBND(octx->curo4,octx->d_id_bp[3],errhp,":d_id",ADR(d_id),
          SIZ(int),SQLT_INT);
  OCIBND(octx->curo4,octx->c_last_bp[1],errhp,":c_last",c_last,
          SIZ(c_last), SQLT_STR);
  OCIDEF(octx->curo4,octx->c_count_dp,errhp,1,ADR(octx->rcount),SIZ(int),
          SQLT_INT);


  return (0);

}


tkvco ()
```

```
{

  int i;
  int rcount;

  for (i = 0; i < NITEMS; i++) {
    octx->ol_supply_w_id_len[i] = sizeof(int);
    octx->ol_i_id_len[i] = sizeof(int);
    octx->ol_quantity_len[i] = sizeof(int);
    octx->ol_amount_len[i] = sizeof(int);
    octx->ol_delivery_d_len[i] = sizeof(OCIDate);
  }
  octx->ol_supply_w_id_csize = NITEMS;
  octx->ol_i_id_csize = NITEMS;
  octx->ol_quantity_csize = NITEMS;
  octx->ol_amount_csize = NITEMS;
  octx->ol_delivery_d_csize = NITEMS;
retry:
  if(bylastname)
  {
   cbctx.reexec = FALSE;
   execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,100,0,
           NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
   /* will get OCI_NO_DATA if <100 found */
   if ((execstatus != OCI_NO_DATA) && (execstatus != OCI_SUCCESS))
   {
    errcode=OCIERROR(errhp, execstatus);
    if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
    {
     DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
     retries++;
     goto retry;
    } else {
                userlog("ORDERSTATUS curo0 ((execstatus != OCI_NO_DATA) &&
(execstatus != OCI_SUCCESS)) error = %d \n", errcode);
     return -1;
    }
   }
   if (execstatus == OCI_NO_DATA) /* there are no more rows */
   {
    /* get rowcount, find middle one */
    DISCARD OCIAttrGet(octx->curo0,OCI_HTYPE_STMT,&rcount,NULL,
                OCI_ATTR_ROW_COUNT,errhp);
    if (rcount <1)
    {
     userlog("ORDERSTATUS  curo0 rcount=%d\n",rcount);
     return (-1);
    }
    octx->cust_idx=(rcount)/2 ;
   }
   else
   {
    /* count the number of rows */
    execstatus=OCIStmtExecute(tpcsvc,octx->curo4,errhp,1,0,
           NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if ((execstatus != OCI_NO_DATA) && (execstatus != OCI_SUCCESS))
    {
     if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
     {
      DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
      retries++;
      goto retry;
     } else {
                userlog("ORDERSTATUS curo4 ((execstatus != OCI_NO_DATA) &&
(execstatus != OCI_SUCCESS)) error = %d \n", errcode);
      return -1;
     }
    }
    if (octx->rcount+1 < 2*10 )
      octx->cust_idx=(octx->rcount+1)/2 ;
    else          /* */
    {
      cbctx.reexec = TRUE;
      cbctx.count = (octx->rcount+1)/2 ;
      execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,cbctx.count,
                        0,NULLP(CONST OCISnapshot),
                        NULLP(OCISnapshot),OCI_DEFAULT);
      /* will get OCI_NO_DATA if <100 found */
      if (cbctx.count > 0)
      {
        userlog ("ORDERSTATUS curo0 did not get all rows ");
        return (-1);
      }

      if ((execstatus != OCI_NO_DATA) && (execstatus != OCI_SUCCESS))
      {
       errcode=OCIERROR(errhp, execstatus);
       if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
       {
        DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
        retries++;
        goto retry;
       } else {
                userlog ("ORDERSTATUS curo0 ((execstatus != OCI_NO_DATA) &&
(execstatus != OCI_SUCCESS)) error = %d\n", errcode);
         return -1;
       }
      }
      octx->cust_idx=0 ;
     }
    }

    octx->c_rowid_cust = octx->c_rowid_ptr[octx->cust_idx];
    execstatus=OCIStmtExecute(tpcsvc,octx->curo1,errhp,1,0,
              NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if (execstatus |= OCI_SUCCESS)
    {
      errcode=OCIERROR(errhp,execstatus);
      DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
      if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
                || (errcode == SNAPSHOT_TOO_OLD))
      {
        retries++;
        goto retry;
        } else {
                userlog ("ORDERSTATUS curo1 (execstatus |= OCI_SUCCESS) error=%d\n",
errcode);
        return -1;
      }
     }
    }
    else
    {
     execstatus=OCIStmtExecute(tpcsvc,octx->curo2,errhp,1,0,
                NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
                OCI_DEFAULT);
     if (execstatus |= OCI_SUCCESS)
     {
      errcode=OCIERROR(errhp,execstatus);
      DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
      if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
                || (errcode == SNAPSHOT_TOO_OLD))
      {
       retries++;
       goto retry;
      }
      else
      {
                userlog ("ORDERSTATUS curo2 (execstatus |= OCI_SUCCESS) error=%d\n",
errcode);
       return -1;
      }
     }
    }
    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,
                  NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
                  OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
    if (execstatus |= OCI_SUCCESS )
    {
     errcode=OCIERROR(errhp,execstatus);
     DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
     if(errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
                || (errcode == SNAPSHOT_TOO_OLD))
     {
      retries++;
      goto retry;
     }
     else
     {
                userlog ("ORDERSTATUS curo3 (execstatus |= OCI_SUCCESS) error=%d\n",
errcode);
      return -1;
     }

    }
   /* clean up and convert the delivery dates */
    for (i = 0; i < o_ol_cnt; i++)
    {
     ol_del_len[i]=sizeof(ol_delivery_d[i]);
     DISCARD OCIERROR(errhp,OCIDateToText(errhp,&ol_d_base[i],
       (const text*)SHORTDATE,(ub1)strlen(SHORTDATE),(text*)0,0,
        &ol_del_len[i], ol_delivery_d[i]));
/*
                cvtdmy(ol_d_base[i],ol_delivery_d[i]);
*/
    }

   return (0);

}



void tkvcodone ()
```

```
{

  if (octx)
    free (octx);

}

/* end of file tkvcord.c */
```

client/oracle/plsto.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: plsto.c 7010000.3 95/02/14 12:48:03 plai Generic<base> $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*=============================================================+
|      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA     |
|              OPEN SYSTEMS PERFORMANCE GROUP                  |
|                  All Rights Reserved                         |
+=============================================================+
| FILENAME
|   plsto.c
| DESCRIPTION
|   OCI version of STOCK LEVEL transaction in TPC-C benchmark.
+=============================================================*/

#include "ora_tpcc.h"
#include "tpccflags.h"

#ifdef PLSQLSTO
#define SQLTXT "BEGIN stocklevel.getstocklevel (:w_id, :d_id, :threshold, \
  :low_stock); END;"
#else
#define SQLTXT "SELECT count (DISTINCT s_i_id) \
        FROM ordl, stok, dist \
        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) \
                                order by ol_o_id desc"
/* query using functional index */
/*
#define SQLTXT "SELECT count (DISTINCT s_i_id) \
        FROM ordl, stok, dist \
        WHERE d_id = :d_id AND d_w_id = :w_id AND \
            d_id = ol_d_id AND d_w_id = ol_w_id AND \
            ol_o_id BETWEEN (d_next_o_id - 20) AND (d_next_o_id - 1) AND \
            decode(SIGN(s_quantity -21) , -1, s_w_id*100000 + s_i_id, NULL) \
                                = ol_w_id*100000 + ol_i_id  AND \
            s_quantity < :threshold;"

*/

#endif

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

typedef struct stoctx stoctx;

stoctx *sctx;


tkvcsinit ()

{

  text stmbuf[SQL_BUF_SIZE];
  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));
#ifndef PLSQLSTO
  OCIERROR(errhp,
    OCIAttrSet(sctx->curs,OCI_HTYPE_STMT,(dvoid*)&sctx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));
#endif

  /* bind variables */

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

  return (0);

}



tkvcs ()

{

retry:
   execstatus= OCIStmtExecute(tpcsvc,sctx->curs,errhp,1,0,0,0,
                        OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
   if (execstatus != OCI_SUCCESS)
   {
    errcode=OCIERROR(errhp,execstatus);
    OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
    if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
       || (errcode == SNAPSHOT_TOO_OLD))
    {
      retries++;
      goto retry;
    } else {
      return -1;
    }
   }

  return (0);
}



void tkvcsdone ()

{
 if(sctx) free(sctx);
}
```

client/oracle/pldel.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: pldel.c 7030100.5 96/06/24 16:26:06 plai Generic<base> $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*=============================================================+
|      Copyright (c) 1996  Oracle Corp, Redwood Shores, CA     |
|              OPEN SYSTEMS PERFORMANCE GROUP                  |
|                  All Rights Reserved                         |
+=============================================================+
| FILENAME
|   pldel.c
| DESCRIPTION
|   OCI version of DELIVERY transaction in TPC-C benchmark.
+=============================================================*/

#include "ora_tpcc.h"
#ifdef TUX
#include <userlog.h>
#endif

/*
extern int userlog();
*/

#include "tpccflags.h"

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

```
#endif

#define SQLTXT "BEGIN inittpcc.init_del ; END;"

#define SQLTXT1 "DELETE FROM nord WHERE no_d_id = :d_id \
                 AND no_w_id = :w_id and rownum <= 1 \
                RETURNING no_o_id into :o_id "

#define SQLTXT3 "UPDATE ordr 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"

#define SQLTXT4 "UPDATE ordl \
   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 sum(ol_amount) into :ol_amount "

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

#define NDISTS 10
#define ROWIDLEN 20


struct delctx {
   sb2 del_o_id_ind[NDISTS];
   sb2 d_id_ind[NDISTS];
   sb2 c_id_ind[NDISTS];
   sb2 del_date_ind[NDISTS];
   sb2 carrier_id_ind[NDISTS];
   sb2 amt_ind[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   sb2 inum_ind;
#endif

   ub4 del_o_id_len[NDISTS];
   ub4 c_id_len[NDISTS];
   int oid_ctx;
   int cid_ctx;
   OCIBind *olamt_bp;

   ub2 w_id_len[NDISTS];
   ub2 d_id_len[NDISTS];
   ub2 del_date_len[NDISTS];
   ub2 carrier_id_len[NDISTS];
   ub2 amt_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];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   ub2 inum_rcode;
#endif

   int del_o_id[NDISTS];
   int del_d_id[NDISTS];
   int cons[NDISTS];
   int w_id[NDISTS];
   int d_id[NDISTS];
   int c_id[NDISTS];
   int carrier_id[NDISTS];
   int amt[NDISTS];
   ub4 del_o_id_rcnt;
   int retry;
   OCIRowid *no_rowid_ptr[NDISTS];
   OCIRowid *o_rowid_ptr[NDISTS];
   OCIDate del_date[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   char inum[10];
#endif
   OCIStmt *curd0;
   OCIStmt *curd1;
   OCIStmt *curd2;
   OCIStmt *curd3;
   OCIStmt *curd4;
   OCIStmt *curd5;
   OCIStmt *curd6;
   OCIStmt *curdtest;

   OCIBind *w_id_bp;
   OCIBind *w_id_bp3;
   OCIBind *w_id_bp4;
   OCIBind *w_id_bp5;
   OCIBind *w_id_bp6;
   OCIBind *d_id_bp;
   OCIBind *d_id_bp3;
   OCIBind *d_id_bp4;
   OCIBind *d_id_bp6;
   OCIBind *o_id_bp;
   OCIBind *cr_date_bp;
   OCIBind *c_id_bp;
   OCIBind *c_id_bp3;
   OCIBind *no_rowid_bp;
   OCIBind *carrier_id_bp;
   OCIBind *o_rowid_bp;
   OCIBind *del_o_id_bp;
   OCIBind *del_o_id_bp3;
   OCIBind *amt_bp;
   OCIBind *bstr1_bp[10];
   OCIBind *bstr2_bp[10];
   OCIBind *retry_bp;
   OCIDefine *inum_dp;
   OCIDefine *d_id_dp;
   OCIDefine *del_o_id_dp;
   OCIDefine *no_rowid_dp;
   OCIDefine *c_id_dp;
   OCIDefine *o_rowid_dp;
   OCIDefine *cons_dp;
   OCIDefine *amt_dp;

   int norow;
};

typedef struct delctx delctx;
struct pldelctx {

   ub2 del_d_id_len[NDISTS];
   ub2 del_o_id_len[NDISTS];


   ub2 w_id_len;
   ub2 d_id_len[NDISTS];
   ub2 o_c_id_len[NDISTS];
   ub2 sums_len[NDISTS];
   ub2 carrier_id_len;
   ub2 ordcnt_len;
   ub2 del_date_len;
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   ub2 inum_len;
#endif

   int del_o_id[NDISTS];
   int del_d_id[NDISTS];
   int o_c_id[NDISTS];
   int sums[NDISTS];
   OCIDate del_date;
   int carrier_id;
   int ordcnt;

   ub4 del_o_id_rcnt;
   ub4 del_d_id_rcnt;
   ub4 o_c_id_rcnt;
   ub4 sums_rcnt;

   int retry;
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   char inum[10];
#endif
   OCIStmt *curp1;
   OCIStmt *curp2;
   OCIBind *w_id_bp;
   OCIBind *d_id_bp;
   OCIBind *o_id_bp;
   OCIBind *o_c_id_bp;
   OCIBind *ordcnt_bp;
   OCIBind *sums_bp;
   OCIBind *del_date_bp;
   OCIBind *carrier_id_bp;
   OCIBind *retry_bp;

   int norow;

};
typedef struct pldelctx pldelctx;

static pldelctx *pldctx;


static delctx *dctx;

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


# ifdef OLD
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=actx->ol_cnt+1;
  *bufpp = &actx->ol_amt[index];
  *indpp= &actx->ol_amt_ind[index];
  actx->ol_amt_len[index]=sizeof(actx->ol_amt[0]);
  *alenp= &actx->ol_amt_len[index];
  *rcodepp = &actx->ol_amt_rcode[index];
  *piecep =OCI_ONE_PIECE;
  if (iter ==1 )
    return (OCI_CONTINUE);
  else
    return (OCI_ERROR);
}
# else
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;
  *bufpp = &actx->ol_amt[index];
  *indpp= &actx->ol_amt_ind[index];
  actx->ol_amt_len[index]=sizeof(actx->ol_amt[0]);
  *alenp= &actx->ol_amt_len[index];
  *rcodepp = &actx->ol_amt_rcode[index];
  *piecep =OCI_ONE_PIECE;
  return (OCI_CONTINUE);
}
# endif

#endif


tkvcdinit (int plsqlflag)

{

  text stmbuf[SQL_BUF_SIZE];

  if (plsqlflag)
  {
    pldctx = (pldelctx *) malloc (sizeof(pldelctx));
    DISCARD memset(pldctx,(char)0,(ub4)sizeof(pldelctx));
    /* Initialize */
    DISCARD OCIHandleAlloc(tpcenv, (dvoid**)&pldctx->curp1, OCI_HTYPE_STMT, 0,
                (dvoid**)0);
    DISCARD sprintf ((char *) stmbuf, SQLTXT);
    DISCARD OCIStmtPrepare(pldctx->curp1, errhp, stmbuf,
                (ub4) strlen((char *)stmbuf),
              OCI_NTV_SYNTAX, OCI_DEFAULT);
    DISCARD OCIERROR(errhp,
      OCIStmtExecute(tpcsvc,pldctx->curp1,errhp,1,0,NULLP(OCISnapshot),
                NULLP(OCISnapshot), OCI_DEFAULT));


    DISCARD OCIHandleAlloc(tpcenv,(dvoid**) &pldctx->curp2, OCI_HTYPE_STMT,
                0, (dvoid**)0);
    sqlfile("../blocks/tkvcpdel.sql",stmbuf);
    DISCARD OCIStmtPrepare(pldctx->curp2, errhp, stmbuf,
                (ub4)strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIBNDPL(pldctx->curp2, pldctx->w_id_bp , errhp,":w_id",
            ADR(w_id), SIZ(int), SQLT_INT,&pldctx->w_id_len);
    OCIBNDPL(pldctx->curp2, pldctx->ordcnt_bp , errhp,":ordcnt",
            ADR(pldctx->ordcnt), SIZ(int), SQLT_INT,&pldctx->ordcnt_len);
    OCIBNDPL(pldctx->curp2, pldctx->del_date_bp,errhp,":now",
            dctx->del_date, SIZ(OCIDate), SQLT_ODT,&pldctx->del_date_len);
    OCIBNDPL(pldctx->curp2, pldctx->carrier_id_bp , errhp,
            ":carrier_id", ADR(o_carrier_id), SIZ(int),
            SQLT_INT, &pldctx->carrier_id_len);

    OCIBNDPLA(pldctx->curp2, pldctx->d_id_bp, errhp,":d_id",
             pldctx->del_d_id, SIZ(int),SQLT_INT, pldctx->del_d_id_len,
             NDISTS, &pldctx->del_d_id_rcnt);
    OCIBNDPLA(pldctx->curp2, pldctx->o_id_bp, errhp,":order_id",
             pldctx->del_o_id,SIZ(int),SQLT_INT, pldctx->del_o_id_len,NDISTS,
             &pldctx->del_o_id_rcnt);
    OCIBNDPLA(pldctx->curp2, pldctx->sums_bp, errhp,"sums",
             pldctx->sums,SIZ(int),SQLT_INT, pldctx->sums_len,NDISTS,
             &pldctx->sums_rcnt);
    OCIBNDPLA(pldctx->curp2, pldctx->o_c_id_bp, errhp,":o_c_id",
             pldctx->o_c_id,SIZ(int),SQLT_INT, pldctx->o_c_id_len,NDISTS,
             &pldctx->o_c_id_rcnt);
    OCIBND(pldctx->curp2, pldctx->retry_bp , errhp,":retry",
           ADR(pldctx->retry), SIZ(int),SQLT_INT);

  }
  else
  {

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

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
    OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd0), OCI_HTYPE_STMT, 0,
                (dvoid**)0);
    sprintf ((char *) stmbuf, SQLTXT0);
    OCIStmtPrepare(dctx->curd0, errhp, stmbuf, strlen((char *)stmbuf),
                OCI_NTV_SYNTAX, OCI_DEFAULT);

    OCIDFNRA(dctx->curd0, dctx->inum_dp,errhp,1,dctx->inum,SIZ(dctx->inum),
            SQLT_STR,&(dctx->inum_ind),&(dctx->inum_len),&(dctx->inum_rcode));
#endif

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

    OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd1), OCI_HTYPE_STMT, 0,
                (dvoid**)0);
    DISCARD sprintf ((char *) stmbuf, "%s",    SQLTXT1);
    DISCARD 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);

  /* open third cursor */

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


  /* bind variables */

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

    OCIBNDRA(dctx->curd3, dctx->w_id_bp3, errhp, ":w_id", dctx->w_id,SIZ(int),
            SQLT_INT, NULL, NULL, NULL);
    OCIBNDRA(dctx->curd3, dctx->d_id_bp3, errhp, ":d_id", dctx->d_id,SIZ(int),
```

```
        SQLT_INT,NULL, NULL, NULL);
  OCIBNDRA(dctx->curd3, dctx->del_o_id_bp3, errhp, ":o_id", dctx->del_o_id,
        SIZ(int), SQLT_INT,NULL,NULL,NULL);
  OCIBNDRAD(dctx->curd3, dctx->c_id_bp3, errhp, ":o_c_id", SIZ(int),
        SQLT_INT,NULL,&dctx->cid_ctx,no_data, cid_data);

 /* open fourth cursor */

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

 /* bind variables */

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



 /* open sixth cursor */

  DISCARD OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd6), OCI_HTYPE_STMT,
                  0, (dvoid**)0);
  DISCARD sprintf ((char *) stmbuf, SQLTXT6);
  DISCARD 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);
 }
 return (0);

}


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


tkvcd (int plsqlflag)

{

  int i, j;
  int rpc,rcount,count;
  int invalid;

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

  OCIStmtExecute(tpcsvc,dctx->curd0,errhp,1,0,0,0,OCI_DEFAULT);
  sysdate (sdate);
  printf ("Delivery started at %s on node %s\n", sdate, dctx->inum);
#endif
  if (plsqlflag)
  {

    pldctx->w_id_len = sizeof (int);
    pldctx->carrier_id_len = sizeof (int);
    for (i = 0; i < NDISTS; i++)
    {
      pldctx->del_o_id_len[i] = sizeof(int);
      del_o_id[i] = 0;
    }
    pldctx->del_date_len = DEL_DATE_LEN;
    DISCARD memcpy(&pldctx->del_date,&cr_date,sizeof(OCIDate));

    pldctx->retry=0;

    DISCARD OCIERROR(errhp,
      OCIStmtExecute(tpcsvc,pldctx->curp2,errhp,1,0,NULLP(CONST OCISnapshot),
              NULLP(OCISnapshot),OCI_DEFAULT));
    for (i = 0; i < NDISTS; i++)
    {
      del_o_id[i] = 0;
    }
    for (i = 0; i < pldctx->del_o_id_rcnt; i++)
      del_o_id[pldctx->del_d_id[i] - 1] = pldctx->del_o_id[i];
  }
  else
  {


retry:

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
    reread = 1;
#endif

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
iso:
#endif

    invalid = 0;

   /* initialization for array operations */

    for (i = 0; i < NDISTS; i++)
    {
     dctx->del_o_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->del_o_id_len[i] = SIZ(dctx->del_o_id[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->w_id[i] = w_id;
     dctx->d_id[i] = i+1;
     dctx->carrier_id[i] = o_carrier_id;
     memcpy(&dctx->del_date[i],&cr_date,sizeof(OCIDate));
    }

    memset(actx,(char)0,sizeof(amtctx));

   /* array select from new_order and orders tables */

    execstatus=OCIStmtExecute(tpcsvc,dctx->curd1,errhp,NDISTS,0,
             NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if((execstatus != OCI_SUCCESS) && (execstatus != OCI_NO_DATA))
    {
     DISCARD OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     errcode = OCIERROR(errhp,execstatus);
     if(errcode == NOT_SERIALIZABLE)
     {
      retries++;
      goto retry;
     }
     else if (errcode == RECOVERR)
     {
      retries++;
      goto retry;
     }
     else if (errcode == SNAPSHOT_TOO_OLD)
     {
      retries++;
      goto retry;
     }
     else
     {
      return -1;
     }
    }
    /* mark districts with no new order */
    DISCARD OCIAttrGet(dctx->curd1,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
           OCI_ATTR_ROW_COUNT,errhp);
    rpc = rcount;
    if (rcount != NDISTS )
    {
     int j = 0;
     for (i=0;i < NDISTS; i++)
```

```
      {
       if (dctx->del_o_id_ind[j] == 0) /* there is data here */
         j++;
       else
         shiftdata(j);
      }
    }

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
    if (invalid) {
      sysdate (sdate);
      for (i = 1; i <= NDISTS; i++) {
        hasno = 0;
        for (j = 0; j < rpc; j++) {
          if (dctx->d_id[j] == i) {
            hasno = 1;
            break;
          }
        }
        if (!hasno)
          printf ("Delivery [dist %d] found no new order at %s\n", i, sdate);
      }
      if (reread) {
        sleep (60);
        sysdate (sdate);
        printf ("Delivery wake up at %s\n", sdate);
        reread = 0;
        goto iso;
      }
    }
#endif

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

    DISCARD OCIAttrGet(dctx->curd3,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
          OCI_ATTR_ROW_COUNT,errhp);

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

    /* array update of order_line table */
    execstatus=OCIStmtExecute(tpcsvc,dctx->curd4,errhp,rpc,0,
            NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if(execstatus != OCI_SUCCESS)
    {
      DISCARD OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      errcode = OCIERROR(errhp,execstatus);
      if(errcode == NOT_SERIALIZABLE)
      {
        retries++;
        goto retry;
      }
      else if (errcode == RECOVERR)
      {
        retries++;
        goto retry;
      }
      else if (errcode == SNAPSHOT_TOO_OLD)
      {
        retries++;
        goto retry;
      }
      else
      {
        return -1;
      }
    }
    DISCARD OCIAttrGet(dctx->curd4,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
          OCI_ATTR_ROW_COUNT,errhp);
/* transfer amounts  */
    for (i=0;i<rpc;i++)
    {
      dctx->amt[i]=0;
      if ( actx->ol_amt_rcode[i] == 0)
      {
        dctx->amt[i] = actx->ol_amt[i];
      }
    }
#ifdef OLD
    if (rcount > rpc) {
      userlog
        ("Error in TPC-C server %d: %d ordnrs updated, %d ordl updated\n",
            proc_no, rpc, rcount);
    }
#endif
#if defined(ISO5) || defined(ISO6)
    printf ("d_id:amount\n");
    for (i = 0; i < rpc; i++)
      printf ("%d:%.2f ", dctx->d_id[i], (float)dctx->amt[i]/100);
    printf ("\n");
#endif

    /* array update of customer table */
#if defined(ISO5) || defined (ISO6)
    execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,0,0,
                                                        OCI_DEFAULT);
#else
    execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,
            NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
            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 if (errcode == SNAPSHOT_TOO_OLD)
      {
        retries++;
        goto retry;
      }
      else
      {
        return -1;
      }
    }

    DISCARD OCIAttrGet(dctx->curd6,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
          OCI_ATTR_ROW_COUNT,errhp);

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

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

```
#ifdef ISO6
    DISCARD  printf("Delivery ISO6 Rolling back.\n");
    DISCARD  OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
#endif

#ifdef ISO5
    DISCARD OCITransCommit(tpcsvc, errhp, OCI_DEFAULT);
#endif

#if defined(ISO5) || defined(ISO6)
    sysdate (sdate);
    DISCARD printf ("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];
  }
  return (0);

}



void tkvcddone (int plsqlflag)

{

  if (plsqlflag)
  {
   if (pldctx)
   {
    DISCARD OCIHandleFree((dvoid *)dctx->curd0,OCI_HTYPE_STMT);
    DISCARD free(pldctx);
   }
  }
  else
  {
   if (dctx)
   {
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
    OCIHandleFree((dvoid *)dctx->curd0,OCI_HTYPE_STMT);
#endif
    /* Again the above will cause a problem if both PSQLDEL and ISO are
                defined - VMM 12/30/97 */
    OCIHandleFree((dvoid *)dctx->curd1,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd2,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd3,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd4,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd5,OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)dctx->curd6,OCI_HTYPE_STMT);
    DISCARD free (dctx);
   }
  }

}
```

client/oracle/ora_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>

#ifndef boolean
#define boolean int
#endif

#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 void TPCexit ();
extern int TPCdumpinit ();
extern void TPCdumpnew ();
extern void TPCdumppay ();
extern void TPCdumpord ();
extern void TPCdumpdel ();
extern void TPCdumpsto ();
extern void TPCdumpexit ();
extern void userlog(char* fmtp, ...);


/* 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.hh24:mi:ss"
#define SHORTDATE "dd-mm-yyyy"



#define DELRT 80.0

extern int tkvcninit ();
extern int tkvcpinit ();
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 void 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];
/* xnie - begin */
extern OCIRowid *o_rowid;
/* xnie - end */

/* 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(x)  (x  *)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));



/* bind arrays for sql */
#define OCIBNDRA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
     DISCARD ocierror(__FILE__,__LINE__,(errp), \
      OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
     DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\

(progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));


/* use with callback data */
#define OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,\
            cbf_nodata,cbf_data) \
     DISCARD ocierror(__FILE__,__LINE__,(errp), \
      OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
     DISCARD ocierror(__FILE__,__LINE__,(errp), \
        OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar), \
                strlen((sqlvar)),0,(progvl),(ftype), \
                indp,0,0,0,0,OCI_DATA_AT_EXEC)); \
     DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)));



/* bind in/out for plsql without indicator and rcode */
#define OCIBNDPL(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen) \
   DISCARD ocierror(__FILE__,__LINE__,(errp), \
     OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
   DISCARD ocierror(__FILE__,__LINE__,(errp), \
     OCIBindByName((stmp),&(bndp),(errp),(CONST text *)(sqlvar), \
       (sb4)strlen((CONST char *)(sqlvar)), (dvoid*)(progv),(progvl),(ftype),\
            NULLP(dvoid),(alen), NULLP(ub2), 0,NULLP(ub4),OCI_DEFAULT));

/* bind  in values for plsql with indicator and rcode */
#define OCIBNDR(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
     DISCARD ocierror(__FILE__,__LINE__,(errp), \
      OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
     DISCARD ocierror(__FILE__,__LINE__,(errp), \
      OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
                                (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0, \
            OCI_DEFAULT));

/* bind in/out for plsql arrays witout indicator and rcode */
```

```
#define OCIBNDPLA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen,ms,cu) \
      DISCARD ocierror(__FILE__,__LINE__, (errp), \
        OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));\
      DISCARD ocierror(__FILE__,__LINE__,(errp),\
        OCIBindByName((stmp),&(bndp),(errp),(CONST text *)(sqlvar), \
            (sb4)strlen((CONST char *) (sqlvar)),(void *)(progv), \
            (progvl),(ftype),NULL,(alen),NULL,(ms),(cu),OCI_DEFAULT));

/* bind in/out values for plsql with indicator and rcode */
#define OCIBNDRAA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode,\
            ms,cu) \
      ocierror(__FILE__,__LINE__, (errp), \
        OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));\
      ocierror(__FILE__,__LINE__,(errp),\
        OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
      (progv),(progvl),(ftype),(indp),(alen),(arcode),(ms),(cu),OCI_DEFAULT));

#define OCIDEFINE(stmp,dfnp,errp,pos,progv,progvl,ftype)\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),(ftype),\
                                          0,0,0,OCI_DEFAULT);


#define OCIDEF(stmp,dfnp,errp,pos,progv,progvl,ftype) \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                                                            (dvoid**)0);\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),\
                  (ftype),NULL,NULL,NULL,OCI_DEFAULT); \


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

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



/* New order */

struct newinstruct {
   int w_id;
   int d_id;
   int c_id;
   int ol_i_id[15];
   int ol_supply_w_id[15];
   int ol_quantity[15];
};

struct newoutstruct {
   int terror;
   int o_id;
   int o_ol_cnt;
   char c_last[17];
   char c_credit[3];
   float c_discount;
   float w_tax;
   float d_tax;
   char o_entry_d[20];
   float total_amount;
   char i_name[15][25];
   int s_quantity[15];
   char brand_generic[15];
   float i_price[15];
   float ol_amount[15];
   char status[26];
   int retry;
};

struct newstruct {
   struct newinstruct newin;
   struct newoutstruct newout;
};


/* Payment */

struct payinstruct {
   int w_id;
   int d_id;
   int c_w_id;
   int c_d_id;
   int c_id;
   int bylastname;
   int h_amount;
   char c_last[17];
};

struct payoutstruct {
   int terror;
   char w_street_1[21];
   char w_street_2[21];
   char w_city[21];
   char w_state[3];
   char w_zip[10];
   char d_street_1[21];
   char d_street_2[21];
   char d_city[21];
   char d_state[3];
   char d_zip[10];
   int  c_id;
   char c_first[17];
   char c_middle[3];
   char c_last[17];
   char c_street_1[21];
   char c_street_2[21];
   char c_city[21];
   char c_state[3];
   char c_zip[10];
   char c_phone[17];
   char c_since[11];
   char c_credit[3];
   double c_credit_lim;
   float  c_discount;
   double c_balance;
   char c_data[201];
   char h_date[20];
   int retry;
};

struct paystruct {
   struct payinstruct payin;
   struct payoutstruct payout;
};


/* Order status */

struct ordinstruct {
   int w_id;
   int d_id;
   int c_id;
   int bylastname;
   char c_last[17];
};

struct ordoutstruct {
   int terror;
   int c_id;
   char c_last[17];
   char c_first[17];
   char c_middle[3];
   double c_balance;
   int o_id;
   char o_entry_d[20];
   int o_carrier_id;
   int o_ol_cnt;
   int ol_supply_w_id[15];
   int ol_i_id[15];
   int ol_quantity[15];
   float ol_amount[15];
   char ol_delivery_d[15][11];
   int retry;
};

struct ordstruct {
   struct ordinstruct ordin;
   struct ordoutstruct ordout;
};


/* Delivery */

struct delinstruct {
   int w_id;
   int o_carrier_id;
   double qtime;
   int in_timing_int;
   int plsqlflag;
};

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
```

client/oracle/tpccflags.h

```
#define PLSQLNO
#define DMLRETDEL
```

A.4 Server Stored Procedures

ACID/blocks/new.sql

```
DECLARE   /* new order */
   not_serializable        EXCEPTION;
   PRAGMA EXCEPTION_INIT(not_serializable,-8177);
   deadlock                EXCEPTION;
   PRAGMA EXCEPTION_INIT(deadlock,-60);
   snapshot_too_old        EXCEPTION;
   PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
 BEGIN
   LOOP BEGIN
     SELECT c_discount, c_last, c_credit
      INTO :c_discount, :c_last, :c_credit
      FROM cust
     WHERE c_id = :c_id
       AND c_d_id = :d_id
       AND c_w_id = :w_id;

     UPDATE wh_dist SET d_next_o_id = d_next_o_id + 1, d_tax=d_tax+0
      WHERE  d_id = :d_id
        AND  w_id = :w_id
   RETURNING d_tax, d_next_o_id-1, w_tax
      INTO :d_tax, :o_id, :w_tax;

     INSERT INTO nord  (no_o_id, no_d_id, no_w_id)
         VALUES (:o_id , :d_id, :w_id);
     INSERT INTO ordr (o_id, o_w_id, o_d_id, o_c_id, o_carrier_id,
                 o_ol_cnt, o_all_local,o_entry_d)
         VALUES (:o_id, :w_id, :d_id, :c_id, 11,
               :o_ol_cnt, :o_all_local, :cr_date);
     EXIT;

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

ACID/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 ware
       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 inittpcc.ware_name,
           :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


     SELECT rowid
     BULK COLLECT INTO inittpcc.row_id
     FROM cust
     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;

    inittpcc.c_num := sql%rowcount;
    inittpcc.cust_rowid := inittpcc.row_id((inittpcc.c_num) / 2);

     UPDATE cust
      SET c_balance = c_balance - :h_amount,
          c_ytd_payment = c_ytd_payment+ :h_amount,
          c_payment_cnt = c_payment_cnt+1
     WHERE rowid = inittpcc.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 cust
         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 = inittpcc.cust_rowid
         RETURNING substr(c_data,1, 200)
         INTO :c_data;

     END IF;

     UPDATE dist
       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 inittpcc.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 hist (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);

     EXIT;

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

   END LOOP;
 END;
```

ACID/blocks/paynz.sql

```
DECLARE /* paynz */
   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 ware
         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 inittpcc.ware_name, :w_street_1, :w_street_2, :w_city,
          :w_state, :w_zip;

     UPDATE  cust
       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 inittpcc.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;


    IF :c_credit = 'BC' THEN
      UPDATE cust
                  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 = inittpcc.cust_rowid
    RETURNING substr(c_data,1, 200)
       INTO :c_data;

    END IF;

      UPDATE dist
        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 inittpcc.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 hist  (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);
    EXIT;

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

  END LOOP;
END;
```

delay.c

```
/*********************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*********************************************************************************/
#include <sys/time.h>
#include <errno.h>
#ifndef HPUX9
#include <time.h>
#endif
#include "tpcc.h"
#include "shm.h"

delay(sec)
/*************************************************************************
delay sleeps for the specified number of seconds.  (to closest 1/100'th second)
*************************************************************************/
   double sec;
   {
#ifdef HPUX9
   struct timeval delay;
#else
   struct timespec delay;
#endif

   /* if no delay,  done */
   if (sec <= 0.0) return;

   /* add a portion of a clock tick to keep averages correct */
   sec += 1.0 / CLK_TCK;

   /* convert the delay to seconds and nanoseconds */
   delay.tv_sec = sec;
#ifdef HPUX9
   delay.tv_usec = (sec - delay.tv_sec) * 1000000;
#else
   delay.tv_nsec = (sec - delay.tv_sec) * 1000000000;
#endif

   /* sleep on a select call */
#ifdef HPUX9
   if (select(0, NULL, NULL, NULL, &delay) < 0) {
      syserror("delay:  select() call failed\n");
   }
#else
   if (nanosleep(&delay,NULL) == -1) {
                  if (errno != EINTR) {
         syserror("delay:  nanosleep() call failed, errno = %d\n",errno);
                  }
   }
#endif
}



struct timeval start_time;

initclock()
   {
   gettimeofday(&start_time, NULL);
   }


TIME getclock()
/*************************************************************************
getclock returns the current time, expressed in seconds from start of run
*************************************************************************/
   {
   struct timeval current;
   gettimeofday(&current, NULL);

   return elapsed_time(&current);
   }
```

random.h

```
/*********************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*********************************************************************************/
#ifndef TPCC_RANDOM
#define TPCC_RANDOM

#ifdef USE_DRAND48
double drand48();
#else
double randy();
#endif

extern int    MakeNumberString();
extern ID     RandomWarehouse();
extern int    MakeAlphaString();
extern void   RandomPermutation();
extern void   RandomDelay();
extern double exponential();
extern void   Randomize();
extern void   SetRandomSeed();

extern char lastNames[1000][16];
extern char customerData1[10][301];
extern char customerData2[10][201];
extern char stockData1[10][27];
extern char stockData2[10][25];
extern char historyData1[10][13];
extern char historyData2[10][13];
extern char citystreetData1[10][11];
extern char citystreetData2[10][11];
extern char firstNameData1[10][9];
extern char firstNameData2[10][9];
extern char StockDistrict[10][25];
extern char phoneData[10][17];

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

```
RandomNumber selects a uniform random number from min to max inclusive
*****************************************************************************/
#ifdef USE_DRAND48
#define RandomNumber(min,max)  \
                ((int)(drand48() * ((int)(max) - (int)(min) + 1)) + (int)(min))
#else
#define RandomNumber(min,max)  \
                ((int)(randy() * ((int)(max) - (int)(min) + 1)) + (int)(min))
#endif

/*****************************************************************************
NURandomNumber selects a non-uniform random number
*****************************************************************************/
#define NURandomNumber(a, min, max, c) \
   ((RandomNumber(0, a) | RandomNumber(min, max)) + (c)) % \
                        ((max) - (min) + 1) + (min)


#define SelectCityStreetData(data) \
{ \
   strcpy(data,citystreetData1[RandomNumber(0,9)]); \
   strcat(data,citystreetData2[RandomNumber(0,9)]); \
}

#define SelectFirstName(data) \
{ \
   strcpy(data,firstNameData1[RandomNumber(0,9)]); \
   strcat(data,firstNameData2[RandomNumber(0,9)]); \
}

#define SelectHistoryData(data) \
{ \
   strcpy(data,historyData1[RandomNumber(0,9)]); \
   strcat(data,historyData2[RandomNumber(0,9)]); \
}

#define SelectStockData(data) \
{ \
   strcpy(data,stockData1[RandomNumber(0,9)]); \
   strcat(data,stockData2[RandomNumber(0,9)]); \
}

#define SelectClientData(data) \
{ \
   strcpy(data,customerData1[RandomNumber(0,9)]); \
   strcat(data,customerData2[RandomNumber(0,9)]); \
}

#define SelectPhoneData(data) strcpy(data,phoneData[RandomNumber(0,9)])
#define SelectStockDistrict(data) strcpy(data,StockDistrict[RandomNumber(0,9)])

#define MakeZip(zip) \
{ \
  MakeNumberString(4, 4, zip); \
  zip[4] = '1'; \
  zip[5] = '1'; \
  zip[6] = '1'; \
  zip[7] = '1'; \
  zip[8] = '1'; \
  zip[9] = '\0'; \
}

#define MakeAddress(str1, str2, city, state, zip) \
{ \
   SelectCityStreetData(str1); \
   SelectCityStreetData(str2); \
   SelectCityStreetData(city); \
   MakeAlphaString(2,2,state); \
   MakeZip(zip); \
}

#define LastName(num, name) strcpy(name, lastNames[(num)])

#define Original(str) \
{ \
   int len = strlen(str); \
   if (len >= 8) { \
      int pos = RandomNumber(0,(len-8)); \
      str[pos+0] = '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'; \
   } \
}


#endif
```

results_file.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2002/07/18 22:07:44 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include "tpcc.h"



static FILE *rfile;

results_open(id)
   int id;
   {
   char fullname[128];
   char *basename;

   /* get the base file name for the deferred results */
   /*
    * Make it a directory under /tmp so at least we can set it to a
    * symbolic link in case /tmp doesn't have enough room.
    */
   basename = getenv("TPCC_RESULTS_FILE");
   if (basename == NULL)
      basename = "/tmp/TPCC_RESULTS_FILE";

   /* create the full file name */
   sprintf(fullname, "%s.%d", basename, id);

   /* open the file */
   unlink(fullname);
   rfile = fopen(fullname, "wb");
   if (rfile == NULL)
      syserror("Delivery server %d can't open file %s\n", id, fullname);

   /* allocate a larger buffer */
   }



results(t)
   delivery_trans *t;
   {
   if (fwrite(t, sizeof(*t), 1, rfile) != 1)
      syserror("Delivery server: Can't post results\n");
   }


results_close()
   {
   if (fclose(rfile) < 0)
      syserror("Delivery server can't close file\n");
   }
```

/source/bin/count_all_users.sh

```
#!/bin/csh

#*****************************************************************************
*
#@(#) Version: A.10.10 $Date: 2001/12/06 11:22:59 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************
*

sleep $1
~tpcc/bin/countusers.sh > /tmp/countusers.before

sleep $2
~tpcc/bin/countusers.sh > /tmp/countusers.after
```

/source/bin/countusers.sh

```
#!/bin/csh

#*****************************************************************************
*
#@(#) Version: A.10.10 $Date: 2002/07/18 21:14:54 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
```

```
#*****************************************************************************
*

echo "Number of clients on `hostname` at `date`:"
setenv UNIX95 1
ps -L -ef

#echo "Number of clients on `hostname` at `date`:  `ps -e | fgrep client | wc -l`"
```

/source/bin/numproc.aux

```
#!/usr/bin/ksh

# Create a string that contains the names of all drivers, clients, or servers.
construct_string()
{
if [ $# = 2 ]
then
   systems=$1
else
   systems=""
   this_system=1
   max=`expr $3 + 1`
   while [ ${this_system} -lt ${max} ]
   do
                systems="${systems} ${2}${this_system}"
                (( this_system=${this_system}+1 ))
   done
fi

echo $systems
}


if [ XXX${BATCH_TPCC} = XXX ]
   then
                BATCH_TPCC="0"
fi
if [ ${BATCH_TPCC} = "1" ]
   then
                DRVR_PROG="client_batch"
   else
                DRVR_PROG="driver"
fi


drivers=$(construct_string ${DRIVER} ${DRIVER} ${NR_DRIVER})
clients=$(construct_string ${CLIENT} ${CLIENT} ${NR_CLIENT})
servers=$(construct_string ${SERVER} ${SERVER} ${NR_SERVER})


SLEEP=30
N=`expr $TRANS_TIME \* 60`
N=`expr $N / $SLEEP `
N=`expr $N + 10 ` ## Margin for Tuxedoe startup and drivers creation.
echo `date` "Runtime: $TRANS_TIME -> $N samples"

I=1
while [ $I -le $N ]
do

echo
date

## Header
print -n "   #"
J=1
while [ $J -le $NR_CLIENT ]
do
printf " %7d" $J
J=`expr $J + 1`
done
printf "\n"

## Driver driver
print -n "drv:"
for drv in ${drivers}
do
 ND=`remsh $drv -n "ps -u tpcc | grep -c $DRVR_PROG"`
 printf " %7d" $ND
done
printf "\n"

if [ ${BATCH_TPCC} != "1" ]
then

## Client client + service
print -n "clt:"
for clt in ${clients}
do
 NC=`remsh $clt -n "ps -u tuxedo > /tmp/ps.tuxedo; grep -c client /tmp/ps.tuxedo"`
 NS=`remsh $clt -n "grep -c service /tmp/ps.tuxedo"`
 printf " %4d/%2d" $NC $NS
done
printf "\n"

fi

## Sut oracletpcc
print -n "sut:"
for sut in ${servers}
do
 NO=`remsh $sut -n "ps -ef | fgrep -v fgrep | fgrep -c oracletpcc"`
 printf " %7d" $NO
done
printf "\n"
date
echo

sleep $SLEEP
I=`expr $I + 1`

done
```

/source/bin/numproc.sh

```
#!/usr/bin/csh

source ~tpcc/TESTENV

~tpcc/bin/numproc.aux $*
```

/source/bin/set_rtprio

```
#!/bin/csh -f
#
set test = `ps -ef | head -1 | grep -c FSID`

if ($test == 1) then
     # FSID
     set BEG = 17
     set END = 22
else
     # NO FSID
     set BEG = 9
     set END = 15
endif

ps -ef > ps.$$

rtprio 10 -1      # init
rtprio 80 -2      # vhand


foreach i (`cat ps.$$ | grep "inetd" | cut -c${BEG}-${END}`)
rtprio 80 -${i}
end

foreach i (`cat ps.$$ | grep "syncer" | cut -c${BEG}-${END}`)
rtprio  86 -${i}
end

foreach i (`cat ps.$$ | grep "statdaemon" | cut -c${BEG}-${END}`)
rtprio  81 -${i}
end

foreach i (`cat ps.$$ | grep "unhashdaemon" | cut -c${BEG}-${END}`)
rtprio  80 -${i}
end

foreach i (`cat ps.$$ | grep "swapper" | cut -c${BEG}-${END}`)
rtprio  80 -${i}
end

foreach i (`cat ps.$$ | grep "ttisr" | cut -c${BEG}-${END}`)
rtprio  75 -${i}
end

foreach i (`cat ps.$$ | grep "netisr" | cut -c${BEG}-${END}`)
rtprio  75 -${i}
end

foreach i (`cat ps.$$ | grep "nvsisr" | cut -c${BEG}-${END}`)
rtprio  75 -${i}
end

foreach i (`cat ps.$$ | grep "supsched" | cut -c${BEG}-${END}`)
rtprio  75 -${i}
end
```

```
foreach i (`cat ps.$$ | grep "smpsched" | cut -c${BEG}-${END}`)
rtprio  75 -${i}
end

foreach i (`cat ps.$$ | grep "sblksched" | cut -c${BEG}-${END}`)
rtprio  75 -${i}
end

foreach i (`cat ps.$$ | grep "strmem" | cut -c${BEG}-${END}`)
rtprio  75 -${i}
end

foreach i (`cat ps.$$ | grep "strweld" | cut -c${BEG}-${END}`)
rtprio  75 -${i}
end

foreach i (`cat ps.$$ | grep "cm" | cut -c${BEG}-${END}`)
rtprio  20 -${i}
# /oracle/misc/bin/set_affinity 9 ${i}
end

foreach i (`cat ps.$$ | grep "_lck" | cut -c${BEG}-${END}`)
rtprio  80 -${i}
# /oracle/misc/bin/set_affinity 9 ${i}
end

foreach i (`cat ps.$$ | grep "lvmkd" | cut -c${BEG}-${END}`)
rtprio  80 -${i}
end

foreach i (`cat ps.$$ | grep "sh" | grep -v console | cut -c${BEG}-${END}`)
rtprio 80 -${i}
end

foreach i (`cat ps.$$ | grep "rlogin" | cut -c${BEG}-${END}`)
rtprio 80 -${i}
end

foreach i (`cat ps.$$ | grep "console" | cut -c${BEG}-${END}`)
rtprio 80 -${i}
end

# commented out for now.  Set manully before test starts

foreach i (`cat ps.$$ | grep "ora_" | cut -c${BEG}-${END}`)
rtprio 100 -${i}
set_affinity 1 $i
end

set CPU = 0
foreach i (`cat ps.$$ | egrep "ora_dbw" | cut -c${BEG}-${END}`)
rtprio 99 -${i}
set_affinity $CPU $i
set CPU = `expr $CPU + 1 `
end

foreach i (`cat ps.$$ | egrep "ora_lg" | cut -c${BEG}-${END}`)
rtprio 110 -${i}
set_affinity 3 $i
end

foreach i (`cat ps.$$ | grep "oracletpcc" | cut -c${BEG}-${END}`)
rtprio 110 -${i}
end

foreach i (`cat ps.$$ | grep "tnslsnr" | cut -c${BEG}-${END}`)
rtprio 105 -${i}
end


# For clients

foreach i (`cat ps.$$ | grep "service" | cut -c${BEG}-${END}`)
rtprio 80 -${i}
end

foreach i (`cat ps.$$ | grep "client" | cut -c${BEG}-${END}`)
rtprio 80 -${i}
end

foreach i (`cat ps.$$ | grep "BBL" | cut -c${BEG}-${END}`)
rtprio 80 -${i}
end

#rm ps.$$
mv ps.$$ /tmp/TPCC_STATS.PS
```

/source/bin/oracle/audit_table.sh

```
#!/usr/bin/sh
#*****************************************************************************
*
#@(#) Version: A.10.10 $Date: 2001/12/06 11:26:59 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************
*

echo "`date`: counting orders in dist table"

sqlplus -s tpcc/tpcc <<!
  select sum(d_next_o_id) from dist;
  exit;
!
```

/source/bin/oracle/countorders.sh

```
#!/usr/bin/sh
#*****************************************************************************
*
#@(#) Version: A.10.10 $Date: 2001/12/06 11:26:59 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************
*

echo "`date`: counting orders in dist table"

sqlplus -s tpcc/tpcc <<!
  select sum(d_next_o_id) from dist;
  exit;
!
```

/source/bin/oracle/logsize.sh

```
#!/usr/bin/sh
#*****************************************************************************
*
#@(#) Version: A.10.10 $Date: 2001/08/24 15:20:45 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************
*
echo "Log segment size at `date`"
```

/source/blocks/load_ordordl.sql

```
-- anonymous block for loading order/orderline

DECLARE
  order_idx      PLS_INTEGER;
  order_rows     PLS_INTEGER;
  ordl_rows      PLS_INTEGER;
  ordl_idx       PLS_INTEGER;
  ordl_idx_hi    PLS_INTEGER;
  local_idx      PLS_INTEGER;
BEGIN
  order_rows := :order_rows;
  ordl_rows  := :ordl_rows;
  order_idx  := 1;
  ordl_idx   := 1;

  WHILE (order_idx <= order_rows) LOOP

    INSERT INTO ordr (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(order_idx), :o_d_id(order_idx), :o_w_id(order_idx),
              :o_c_id(order_idx), SYSDATE, :o_carrier_id(order_idx),
              :o_ol_cnt(order_idx), 1);

    ordl_idx_hi := ordl_idx + :o_ol_cnt(order_idx) - 1;

    IF ( :o_id(order_idx) < 2101 ) THEN
      FORALL local_idx IN ordl_idx .. ordl_idx_hi
        INSERT INTO ordl (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(local_idx), :ol_d_id(local_idx),
                  :ol_w_id(local_idx), :ol_number(local_idx),
                  SYSDATE, :ol_i_id(local_idx),
                  :ol_supply_w_id(local_idx), 5, 0, :ol_dist_info(local_idx));
    ELSE
      FORALL local_idx IN ordl_idx .. ordl_idx_hi
        INSERT INTO ordl (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(local_idx), :ol_d_id(local_idx),
```

```
                    :ol_w_id(local_idx), :ol_number(local_idx),
                    to_date('01-Jan-1811'), :ol_i_id(local_idx),
                    :ol_supply_w_id(local_idx), 5,
                    :ol_amount(local_idx), :ol_dist_info(local_idx));
    END IF;
    ordl_idx := ordl_idx_hi + 1;
    order_idx := order_idx + 1;
  END LOOP;
END;
```

/source/blocks/tkvcinin.sql

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

CREATE OR REPLACE PACKAGE inittpcc
AS
 TYPE intarray IS TABLE OF INTEGER INDEX BY BINARY_INTEGER;
 TYPE distarray IS TABLE OF VARCHAR(24) INDEX BY BINARY_INTEGER;
 nulldate      DATE;
 TYPE rowidarray IS TABLE OF ROWID INDEX BY PLS_INTEGER;
 s_dist                                    distarray;
 idx1arr                    intarray;
 s_remote                   intarray;
 dist              intarray;
 row_id              rowidarray;
 cust_rowid            rowid;
 dist_name           VARCHAR2(11);
 ware_name            VARCHAR2(11);
 c_num              PLS_INTEGER;

 PROCEDURE init_no(idxarr intarray);
 PROCEDURE init_del;
 PROCEDURE init_pay;
END inittpcc;
/
show errors;

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

  PROCEDURE init_del
  IS
  BEGIN
    FOR i IN 1 .. 10 LOOP
      dist(i) := i;
    END LOOP;
  END init_del;

  PROCEDURE init_pay IS
  BEGIN
    NULL;
  END init_pay;

END inittpcc;
/
show errors
exit
```

/source/blocks/tkvcpdel.sql

```
declare
  TYPE numarray IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
  TYPE numlist is varray (10) of number;
  dist numarray;
  amt numarray ;
  cnt pls_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);

BEGIN
  LOOP BEGIN
    FORALL d IN 1..10
      DELETE FROM nord
        WHERE no_d_id = inittpcc.dist(d)
          AND no_w_id = :w_id
          AND ROWNUM <= 1
        RETURNING no_d_id, no_o_id BULK COLLECT INTO :d_id, :order_id;

  :ordcnt := SQL%ROWCOUNT;

  FORALL o in 1.. :ordcnt
    UPDATE ordr SET o_carrier_id = :carrier_id
    WHERE o_id = :order_id (o)
      AND o_d_id = :d_id(o)
      AND o_w_id = :w_id
    RETURNING o_c_id BULK COLLECT INTO :o_c_id;

  FORALL o in 1.. :ordcnt
    UPDATE ordl SET ol_delivery_d = :now
    WHERE ol_w_id = :w_id
      AND ol_d_id = :d_id(o)
      AND ol_o_id = :order_id(o)
    RETURNING sum(ol_amount) BULK COLLECT INTO  :sums;

  FORALL c IN 1.. :ordcnt
    UPDATE cust
      SET c_balance = c_balance + :sums(c),
                c_delivery_cnt = c_delivery_cnt + 1
     WHERE c_w_id = :w_id
       AND c_d_id = :d_id(c)
       AND c_id = :o_c_id(c);
  COMMIT;
  EXIT;
  EXCEPTION
    WHEN not_serializable OR deadlock OR snapshot_too_old
    THEN
      ROLLBACK;
      :retry := :retry + 1;
  END;

  END LOOP; -- for retry
END;
```

/source/blocks/tkvcpnew.sql

```
-- New Order Anonymous block

 DECLARE
    idx                 PLS_INTEGER;
    dummy_local           PLS_INTEGER;
    cache_ol_cnt          PLS_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 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                     THEN s_quantity +91
                     ELSE s_quantity
                   END) - :ol_quantity(idx)
       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,
              i_price*:ol_quantity(idx),
        CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
         END
     BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                 :ol_amount,:brand_generic;
  END u1;

  PROCEDURE u2 IS
  BEGIN
     FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                     THEN s_quantity +91
                     ELSE s_quantity
                   END) - :ol_quantity(idx)
       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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u2;

PROCEDURE u3 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u3;

PROCEDURE u4 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u4;

PROCEDURE u5 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u5;

PROCEDURE u6 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u6;

PROCEDURE u7 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u7;

PROCEDURE u8 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u8;

PROCEDURE u9 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
```

```
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u9;

PROCEDURE u10 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u10;

PROCEDURE fix_items IS
  rows_lost                 PLS_INTEGER;
  max_index                 PLS_INTEGER;
  temp_index                PLS_INTEGER;
BEGIN
  idx := 1;
  rows_lost := 0;
  max_index := dummy_local;

  WHILE (max_index != cache_ol_cnt) LOOP

  WHILE (idx <= sql%rowcount AND
              sql%bulk_rowcount(idx + rows_lost) = 1)
  LOOP
    idx := idx + 1;
  END LOOP;

  temp_index := max_index;
  WHILE (temp_index >= idx + rows_lost) LOOP
    :ol_amount(temp_index + 1)      :=  :ol_amount(temp_index);
    :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);
    inittpcc.s_dist(temp_index + 1) :=  inittpcc.s_dist(temp_index);
    :brand_generic(temp_index + 1)  :=  :brand_generic(temp_index);
    temp_index := temp_index - 1;
  END LOOP;

  IF (idx + rows_lost <= cache_ol_cnt) THEN
    :i_price(idx + rows_lost)       :=   0;
    :i_name(idx + rows_lost)        :=   'NO ITEM';
    :s_quantity(idx + rows_lost)    :=   0;
    inittpcc.s_dist(idx + rows_lost) := NULL;
    :brand_generic(idx + rows_lost)  := ' ';
    :ol_amount(idx + rows_lost)     := 0;
    rows_lost := rows_lost + 1;
    max_index := max_index + 1;
  END IF;

 END LOOP;
END fix_items;

BEGIN
  LOOP BEGIN
     cache_ol_cnt := :o_ol_cnt;

     UPDATE dist 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 cust , ware
      WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = w_id
      AND  w_id = :w_id;


     INSERT INTO nord (no_o_id, no_d_id, no_w_id)
      VALUES (:o_id, :d_id, :w_id);

     INSERT INTO ordr  (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 < 6) THEN
       IF (dummy_local < 3) THEN
         IF (dummy_local = 1) THEN
           u1;
         ELSE
           u2;
         END IF;
       ELSE
         IF (dummy_local = 3) THEN
           u3;
         ELSIF (dummy_local = 4) then
           u4;
         ELSE
           u5;
         END IF;
       END IF;
     ELSE
       IF (dummy_local < 8) THEN
         IF (dummy_local = 6) THEN
           u6;
         ELSE
           u7;
         END IF;
       ELSE
         IF (dummy_local = 8) THEN
           u8;
         ELSIF (dummy_local = 9) then
           u9;
         ELSE
           u10;
         END IF;
       END IF;
     END IF;

     dummy_local := sql%rowcount;

     IF (dummy_local != cache_ol_cnt )  THEN fix_items; END IF;

     FORALL idx IN 1..dummy_local
      INSERT INTO ordl
        (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, inittpcc.idx1arr(idx), inittpcc.nulldate,
            :ol_i_id(idx), :ol_supply_w_id(idx),
            :ol_quantity(idx), :ol_amount(idx), inittpcc.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;
```

/source/blocks/views.sql

```
create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
   from dist d, ware 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 stok s, item i
 where i.i_id = s.s_i_id
/
```

source/buildenv.mk

```
OPTIMIZE= +O2 +ESlit -z +Olibcalls +Ofastaccess +Oentrysched +Onolimit +Oprocelim

DATA_MODEL_FLAGS=+DA2.0
ARCHFLAGS= -Ae $(DATA_MODEL_FLAGS) +DS2.0 -DHPUX ## -D_REENTRANT

CINCLUDES= -I. -I../lib

BUILDFLAGS= $(ARCHFLAGS) $(OPTIMIZE) $(CINCLUDES)

MV=cp -r
```

source/make_no_pbo

```
#!/usr/bin/csh

setenv CCOPTS "-Ae +DA2.0 +DS2.0 +pgm client"
make -f Makefile tpcc_client

if (${DATABASE} == "sybase") then
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 +pgm service"
   make -f Makefile service_sybase
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 -DHPUX -D_REENTRANT"
   make -f Makefile others_sybase
else if (${DATABASE} == "oracle") then
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 +pgm service"
   make -f Makefile service_oracle
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 -DHPUX -D_REENTRANT"
   make -f Makefile others_oracle
else if (${DATABASE} == "sqlserver") then
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 +pgm service"
   make -f Makefile service_sqlserver
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 -DHPUX -D_REENTRANT"
   make -f Makefile others_sqlserver
endif
```

Appendix B Database Design

The source code for the process to define, create and populate the Oracle10i Database Standard Edition TPC-C database is included in this appendix.

B.1 Scripts addfile.sh

Addfile.sh

```
#!/usr/bin/sh
# $1 = tablespace name
# $2 = filename
# $3 = size
# $4 = temporary ts (1) or not (0)
# global variable $tpcc_listfiles, does not execute sql

if expr x$tpcc_listfiles = xt > /dev/null; then
  echo $2 $3 >> $tpcc_bench/files.dat
  exit 0
fi

if expr $4 = 1 > /dev/null; then
  altersql="alter tablespace $1 add tempfile '$2' size $3 reuse;"
else
  altersql="alter tablespace $1 add datafile '$2' size $3 reuse autoextend on;"
fi

$tpcc_sqlplus $tpcc_user_pass <<!
   spool addfile_$1.log
   set echo on
  $altersql
   set echo off
   spool off
   exit ;
!
```

Addts.sh

```
#!/usr/bin/sh
# $1 = tablespace name
# $2 = filename
# $3 = size
# $4 = uniform size
# $5 = block size
# $6 = temporary ts (1) or not (0)
# $7 = bitmapped manage (t) or not (f)
# global variable $tpcc_listfiles, does not execute sql

if expr x$tpcc_listfiles = xt > /dev/null; then
  echo $2 $3 >> $tpcc_bench/files.dat
  exit 0
fi


if expr $5 = auto > /dev/null; then
  bssql=
else
  bssql="blocksize $5"
fi

if expr $6 = 1 > /dev/null; then
  createsql="create temporary tablespace $1 tempfile '$2' size $3 reuse extent management local
uniform size $4;"
else
  if expr x$7 = xt > /dev/null; then
    autospace=auto
  else
    autospace=manual
  fi
    createsql="create tablespace $1 datafile '$2' size $3 reuse extent management local uniform size
$4 segment space management $autospace $bssql nologging ;"
fi

$tpcc_sqlplus $tpcc_user_pass <<!
   spool createts_$1.log
   set echo on
   drop tablespace $1 including contents;
 $createsql
   set echo off
   spool off
   exit ;
!
```

Analyze.sh

```
#!/usr/bin/sh
$tpcc_sqlplus $tpcc_user_pass @${tpcc_sql_dir}/analyze > $tpcc_log_dir/junk 2>&1

# this one tends to fail if indices aren't made, which is legal, so
# always exit without error.

exit 0
```

Analyse.sql

```
set echo on;
spool on;

ANALYZE TABLE stok ESTIMATE STATISTICS;
ANALYZE TABLE cust ESTIMATE STATISTICS;
ANALYZE TABLE ordr ESTIMATE STATISTICS;
ANALYZE TABLE ordl ESTIMATE STATISTICS;
ANALYZE TABLE hist ESTIMATE STATISTICS;
ANALYZE TABLE dist ESTIMATE STATISTICS;
ANALYZE TABLE item ESTIMATE STATISTICS;
ANALYZE TABLE ware ESTIMATE STATISTICS;
ANALYZE TABLE nord ESTIMATE STATISTICS;
ANALYZE index iware ESTIMATE STATISTICS;
ANALYZE index idist ESTIMATE STATISTICS;
ANALYZE index iitem ESTIMATE STATISTICS;
ANALYZE index icust1 ESTIMATE STATISTICS;
ANALYZE index icust2 ESTIMATE STATISTICS;
ANALYZE index istok ESTIMATE STATISTICS;
ANALYZE index iordr2 ESTIMATE STATISTICS;

set echo off;
spool off;

exit sql.sqlcode;
```

Assigntemp.sh

```
#!/usr/bin/sh

echo Assigning temporary tablespace to user tpcc...
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_sql_dir/assigntemp > junk 2>&1
if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

Assigntemp.sql

```
spool assigntemp.log;

set echo on;

alter user tpcc temporary tablespace temp_0;

set echo off;
spool off;

exit ;
```

Atom.sql

```
Rem
Rem $Header: atom.sql 01-jun-98.19:15:08 skareenh Exp $
Rem
Rem atom.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
```

```
Rem     atom.sql
Rem
Rem    DESCRIPTION
rem      Performs TPC-C atomicity test.
rem      Asks user to input values for ware_id, dist_id, cust_w_id,
rem      cust_d_id, cust_id and output file.
Rem
Rem    NOTES
rem                Usage:   sqlplus tpcc/tpcc @atom
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem

rem

column c_payment_cnt format 9999999
column c_balance     format 9999999999999
column c_ytd_payment format 9999999999999
column d_ytd         format 9999999999999
column w_ytd         format 9999999999999

Rem accept logfile     char     prompt  'Enter output file   =>  '

set verify off
spool &&1
set termout on
set echo on

Rem accept ware_id     number  prompt  'Enter ware_id       =>  '
Rem accept dist_id     number  prompt  'Enter dist_id       =>  '
Rem accept cust_w_id   number  prompt  'Enter cust_w_id     =>  '
Rem accept cust_d_id   number  prompt  'Enter cust_d_id     =>  '
Rem accept cust_id     number  prompt  'Enter cust_id       =>  '

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM
REM Retrieve initial values.
REM

select w_ytd from ware
   where w_id = &&2;
select d_ytd from dist
   where d_id = &&3 and d_w_id = &&2;
select c_balance, c_ytd_payment, c_payment_cnt from cust
   where c_id = &&6 and c_d_id = &&5 and c_w_id = &&4;
commit;

REM
REM Sleep for 30 seconds before re-reading values.
REM

execute dbms_lock.sleep (30);
commit;

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM
REM Retrieve final values.
REM

select w_ytd from ware
   where w_id = &&2;
select d_ytd from dist
   where d_id = &&3 and d_w_id = &&2;
select c_balance, c_ytd_payment, c_payment_cnt from cust
   where c_id = &&6 and c_d_id = &&5 and c_w_id = &&4;
commit;

spool off
exit
```

Atoma.sh

```
#!/bin/sh
#
# $Header: atoma.sh 01-jun-98.19:05:35 skareenh Exp $
#
# atoma.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      atoma.sh
#
#    DESCRIPTION
#    Perform abort part of TPC-C atomicity test.
#
#    NOTES
#    atoma.sh [output_file]
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

OFILE="atoma.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
      exit 0
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        TT=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

# Use abort versions of the anonymous blocks
cp ${TPCC_AUDIT}/blocks/payz.sql ${TPCC_AUDIT}/blocks/payz_std.sql
cp ${TPCC_AUDIT}/blocks/paynz.sql ${TPCC_AUDIT}/blocks/paynz_std.sql
cp ${TPCC_AUDIT}/blocks/payz_abort.sql  ${TPCC_AUDIT}/blocks/payz.sql
cp ${TPCC_AUDIT}/blocks/paynz_abort.sql ${TPCC_AUDIT}/blocks/paynz.sql

#
# w_id = 3, d_id = 4, c_w_id = 3, c_d_id = 4, c_id = 2000
#
cat > atoma.in1 <<!
```

```
atoma.out1
3
4
3
4
2000
!

#
# txn = 2 (payment), w_id = 3, d_id = 4, c_w_id = 3, c_d_id = 4,
# h_amount = 100.0, c_id = 2000, bylastname = 0, txn = 0 (quit)
#
cat > atoma.in2 <<!
2
3
4
3
4
100.0
2000
0
0
!

#
# check initial c_balance, sleep for 30 sec, check final c_balance
#
#sqlplus tpcc/tpcc @${SQLDIR}/atom < atoma.in1 &
sqlplus tpcc/tpcc @${SQLDIR}/atom atoma.out1 3 4 3 4 2000 &

#
# sleep for 15 sec and do a payment that rollbacks
#
sleep 15
${SRCDIR}/tpcc_acid.exe < atoma.in2 | tee atoma.out2

wait

if [ "$TT" != "" ]
then
   ${SRCDIR}/plsql_mon system/manager@${TT} 5 2 | tee atoma.out3 &
else
   ${SRCDIR}/plsql_mon system/manager 5 2 | tee atoma.out3 &
fi


#
# concatenate all output
#

wait

cat > ${ODIR}/${OFILE} <<!
Initial and Final C_balance
---------------------------

!

cat atoma.out1 >> ${ODIR}/${OFILE}


cat >> ${ODIR}/${OFILE} <<!

Txn info
--------

!

cat atoma.out3 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!

Input and Output of Payment Transaction
---------------------------------------

!

cat atoma.out2 >> ${ODIR}/${OFILE}

rm -f atoma.in1 atoma.in2 atoma.out1 atoma.out2 atoma.out3

# Restore standard versions
cp ${TPCC_AUDIT}/blocks/payz_std.sql  ${TPCC_AUDIT}/blocks/payz.sql
cp ${TPCC_AUDIT}/blocks/paynz_std.sql ${TPCC_AUDIT}/blocks/paynz.sql
```

Atomc.sh

```
#!/bin/sh
#
# $Header: atomc.sh 01-jun-98.19:05:53 skareenh Exp $
#
# atomc.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      atomc.sh
#
#    DESCRIPTION
#    Perform commit part of TPC-C atomicity test.
#
#    NOTES
#    atomc.sh [output_file]
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

OFILE="atomc.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
      exit 0
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        TT=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

# Use commit versions of the anonymous blocks
cp ${TPCC_AUDIT}/blocks/payz_commit.sql  ${TPCC_AUDIT}/blocks/payz.sql
cp ${TPCC_AUDIT}/blocks/paynz_commit.sql ${TPCC_AUDIT}/blocks/paynz.sql

#
# w_id = 2, d_id = 3, c_w_id = 2, c_d_id = 3, c_id = 1000
#
```

```
cat > atomc.in1 <<!
atomc.out1
2
3
2
3
1000
!

#
# txn = 2 (payment), w_id = 2, d_id = 3, c_w_id = 2, c_d_id = 3,
# h_amount = 100.0, c_id = 1000, bylastname = 0, txn = 0 (quit)
#
cat > atomc.in2 <<!
2
2
3
2
3
100.0
1000
0
0
!

#
# check initial c_balance, sleep for 30 sec, check final c_balance
#
#sqlplus tpcc/tpcc @${SQLDIR}/atom < atomc.in1 &
sqlplus tpcc/tpcc @${SQLDIR}/atom atomc.out1 2 3 2 3 1000 &

#
# sleep for 15 sec and do a payment that commits
#
sleep 15
${SRCDIR}/tpcc_acid.exe < atomc.in2 | tee atomc.out2

wait


if [ "$TT" != "" ]
then
   ${SRCDIR}/plsql_mon system/manager@${TT} 5 2 | tee atomc.out3 &
else
   ${SRCDIR}/plsql_mon system/manager 5 2 | tee atomc.out3 &
fi

wait

#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!
Initial and Final C_balance
---------------------------

!

cat atomc.out1 >> ${ODIR}/${OFILE}


cat >> ${ODIR}/${OFILE} <<!


Txn info
--------


!

cat atomc.out3 >> ${ODIR}/${OFILE}



cat >> ${ODIR}/${OFILE} <<!



Input and Output of Payment Transaction
---------------------------------------

!

cat atomc.out2 >> ${ODIR}/${OFILE}

rm -f atomc.in1 atomc.in2 atomc.out1 atomc.out2 atomc.out3

# Restore standard versions
cp ${TPCC_AUDIT}/blocks/payz_std.sql  ${TPCC_AUDIT}/blocks/payz.sql
cp ${TPCC_AUDIT}/blocks/paynz_std.sql ${TPCC_AUDIT}/blocks/paynz.sql
```

Bcexpr.sh

```
#!/bin/sh
# send command line to bc
echo "$*" | bc
```

Buildcreatedb.sh

```
#!/usr/bin/sh

# make sure this is not run accidentally
$tpcc_require tpcc_kit

# If we can do auto undo, add in the code here
undo_sql=
if expr $tpcc_auto_undo = t > /dev/null; then
  echo "Auto undo on."
  undo_sql=`cat <<!
create undo tablespace undo_ts datafile
   '\\$tpcc_disks_location/roll01' size $tpcc_undo_size reuse blocksize $tpcc_undo_bs;
!
`
fi

#save to here
createdbout=$tpcc_genscripts_dir/createdb.sql
rm $createdbout
echo '/* created automatically by ' $0 `date` '*/' > $createdbout

cat >> $createdbout <<!
spool createdb.log

set echo on

startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxinstances $tpcc_np
  datafile '\$tpcc_disks_location/system_001' size $tpcc_system_size reuse
  logfile '\$tpcc_disks_location/log_1' size $tpcc_logfile_size reuse,
          '\$tpcc_disks_location/log_2' size $tpcc_logfile_size reuse
  sysaux datafile '\$tpcc_disks_location/aux.df' size $tpcc_sysaux_size reuse;

$undo_sql

set echo off
exit sql.sqlcode
!
```

Buildcreateindex.sh

```
#!/bin/sh

cd $tpcc_bench/log

. $tpcc_tabledata

if $tpcc_require tpcc_kit; then
  exit 1
fi

createindexout=$tpcc_genscripts_dir/createindex_${1}.sql
rm $createindexout
echo '/*' created automatically by $0 `date` '*/> $createindexout

for index in $1; do
  if $tpcc_require tpcc_kit tpcc_${index}_imp tpcc_${index}_used tpcc_${index}_free
tpcc_${index}_trans tpcc_${index}_flg tpcc_${index}_fl tpcc_${index}_warecol
tpcc_${index}_cols tpcc_${index}_indices tpcc_${index}_indexon; then
    exit 0
  fi
  indeximp=`tp $index imp`

  indexfree=`tp $index free`
  indexfree=`$tpcc_notneg pctfree $indexfree`
  indexused=`tp $index used`
  indexused=`$tpcc_notneg pctused $indexused`
  indextrans=`tp $index trans`
  indextrans=`$tpcc_notneg initrans $indextrans`
  indexflg=`tp $index flg`
  indexfl=`tp $index fl`

  if test x`tp $index autospace` = xt; then
    indexsto=
```

```
  else
    if test -n "`$tpcc_notneg flg $indexflg``$tpcc_notneg fl $indexfl`"; then
      indexsto="freelists $indexfl freelist groups $indexflg "
    fi
  fi

  indexindexon=`tp $index indexon`

  physicalattr=`cat <<!
$indexfree $indexused $indextrans
!
`

  echo Creating index $index as $indeximp.
  indexindices=`tp $index indices`
  indexindices=`indexnames $index $indexindices`

  case $indeximp in
    index)
beginsql=`cat <<!
create unique index $index on $indexindexon ( $indexindices )
  $physicalattr
  storage ( buffer_pool default ${indexsto})
  tablespace ${index}_0 ;
!
`
        ;;
################## index end ############
    partindex)
      indexpara=`$tpcc_bcexpr $tpcc_cpu \* 2`
      partsql=`createindexpart $index`
      if expr $? = 1 > /dev/null; then
        echo index $index may not be partitioned.
        exit 1
      fi
beginsql=`cat <<!
create unique index $index on $indexindexon ( $indexindices )
  $partsql
  parallel $indexpara
  $physicalattr
  storage ( buffer_pool default $indexsto )
  tablespace ${index}_0 ;
!
`

############### partindex end ######################
        ;;
    none)
        beginsql=
        ;;
    *)   echo Invalid index implementation for $index : $indeximp
        exit 1
        ;;
  esac

 if test $indeximp != none; then
cat >> $createindexout <<!
    set sqlblanklines on
    spool createindex_${index}.log ;
    set echo on ;
    drop index $index ;
      $beginsql
    set echo off
    spool off
    exit sql.sqlcode;
!
 else
cat >> $createindexout <<!
  exit 0;
!
 fi

done
exit $?
```

Buildcreatetable.sh

```
#!/bin/sh

. $tpcc_tabledata

initcluster(){
  #common cluster init for clusters and queue.
  if $tpcc_require tpcc_${table}_clustcols tpcc_${table}_hkey tpcc_${table}_hash
tpcc_${table}_size; then
    exit 1
  fi
  clusterkeys=`tp $table hkey`
  clusterhash=`tp $table hash`
  clustersize=`tp $table rsize`
}

if $tpcc_require tpcc_kit; then
  exit 1
fi

createtableout=${tpcc_genscripts_dir}/createtable_${1}.sql
rm $createtableout
echo '/*' created automatically by $0 `date` '*/'> $createtableout

for table in $1; do
  if $tpcc_require tpcc_kit tpcc_${table}_imp tpcc_${table}_used tpcc_${table}_free
tpcc_${table}_trans tpcc_${table}_flg tpcc_${table}_fl tpcc_${table}_warecol
tpcc_${table}_cols tpcc_${table}_indices; then
    exit 1
  fi
  tableimp=`tp $table imp`

  # this is used in several places, so we create it here.
  tablefree=`tp $table free`
  tablefree=`$tpcc_notneg pctfree $tablefree`
  tableused=`tp $table used`
  tableused=`$tpcc_notneg pctused $tableused`
  tabletrans=`tp $table trans`
  tabletrans=`$tpcc_notneg initrans $tabletrans`
  tableflg=`tp $table flg`
  tablefl=`tp $table fl`

  if test x`tp $table autospace` = xt; then
    tablesto=
  else
    if test -n "`$tpcc_notneg flg $tableflg``$tpcc_notneg fl $tablefl`"; then
      tablesto="freelists $tablefl freelist groups $tableflg "
    fi
  fi

  physicalattr="$tablefree $tableused $tabletrans"
  if test -z "`tp $table bpool`"; then
    physicalsto="storage ( buffer_pool default ${tablesto})"
  else
    physicalsto="storage ( buffer_pool `tp $table bpool` ${tablesto})"
  fi

  echo Creating table $table as $tableimp.
  tablecomp=`tp $table compress`
  tablecols=`colize $table all f f $tablecomp`
  case $tableimp in
    table)
beginsql=`cat <<!
drop table $table ;

create table $table (
  $tablecols
)
  $physicalattr
  $physicalsto
  tablespace ${table}_0 ;
!
 `
        ;;
################## table end ############
    cluster)
  initcluster
  clusterusecols=`tp $table clustcols`
  clustercols=`colize $table $clusterusecols f`
  clusternames=`colize $table $clusterusecols t`

beginsql=`cat <<!
drop cluster ${table}cluster including tables ;

create cluster ${table}cluster (
  $clustercols
  )
  single table
  hashkeys $clusterkeys
  hash is ( $clusterhash )
  size $clustersize
  $physicalattr
  $physicalsto
  tablespace ${table}_0;

create table $table (
  $tablecols
)
cluster ${table}cluster (
  $clusternames
);
!
 `
################ cluster end ######################
        ;;
    queue)
```

```
usecluster=`tp $table usecluster`
if test -n "$usecluster"; then
  othercluster=`tp $table othercluster`
  if expr x`tp $othercluster imp` != xqueue > /dev/null; then
    echo table $table shares queue with $othercluster - $othercluster must be queue.
    exit 1
  fi
  beginsql=`cat <<!
  create table $table (
    $tablecols
  $usecluster ;
!
`

else   #not using another cluster
  #note- do we need physical attr in here?
  queuesortcols=`tp $table queuesort`
  if expr x$queuesortcols = x > /dev/null; then
    echo table $table may not be a queue- missing queuesortcols.
    exit 1
  fi
  queuenames=`tp $table queuenames`
  if expr x$queuenames = x > /dev/null; then
    echo table $table may not be a queue- missing queuenames.
    exit 1
  fi

 initcluster
if test -n "`tp $table qcols`"; then
  eval "tpcc_${table}_cols=\$tpcc_${table}_qcols"
fi

 queuecols=`indexnames $table $queuesortcols`
 queuenamecol=`indexnames $table $queuenames`
  beginsql=`cat <<!
  drop cluster ${table}cluster_queue including tables ;

  create cluster ${table}cluster_queue (
    $queuecols
    )

    hashkeys $clusterkeys
    hash is ( $clusterhash )
    size $clustersize
    tablespace ${table}_0;

  create table $table (
    $tablecols
    , constraint ${table}_uk primary key ( ${queuenamecol} )
  )
  cluster ${table}cluster_queue (
    $queuenamecol
  );
!
`
fi

############### queue end ######################
          ;;
    parttable)
      partsql=`createpart $table`
      if expr $? = 1 > /dev/null; then
        echo table $table may not be partitioned.
        exit 1
      fi
beginsql=`cat <<!
drop table $table ;

create table $table (
  $tablecols
)
  $partsql
  $physicalattr
  $physicalsto
  storage ( buffer_pool default $tablesto ) ;
!
`
##############  parttable end ##################
          ;;
    iot)
iotindices=`tp $table indices`
tableind=`indexnames $table $iotindices`
      if test -z "$tableind" > /dev/null; then
        echo table $table does not have an index, so it may not be an iot.
        exit 1
      fi
beginsql=`cat <<!
drop table $table ;

create table $table (
  $tablecols
  , constraint i$table primary key ( $tableind )
)  organization index
  $physicalattr
  $physicalsto
  tablespace ${table}_0 ;
!
`
############## iot end ####################
          ;;
    partiot)
      partsql=`createpart $table`
      if expr $? = 1 > /dev/null; then
        echo table $table may not be partitioned.
        exit 1
      fi
iotindices=`tp $table indices`
tableind=`indexnames $table $iotindices`
      if expr x$tableind = x > /dev/null; then
        echo table $table does not have an index, so it may not be an partitioned iot.
        exit 1
      fi
beginsql=`cat <<!
drop table $table ;

create table $table (
  $tablecols
  , constraint i$table primary key ( $tableind )
)  organization index
  $partsql
  $physicalattr
  $physicalsto
  storage ( buffer_pool default $tablesto )
!
 `
############## partiot end ####################
          ;;
    none)
          ;;
    *)    echo Invalid table implementation for $table : $tableimp
          exit 1
          ;;
  esac

#log to output file
  if test $tableimp != none; then
cat >> $createtableout <<!
    set sqlblanklines on
    spool createtable_$table.log
    set echo on
      $beginsql
    set echo off
    spool off
    exit sql.sqlcode;
!
  else
#create empty shell
cat >> $createtableout <<!
   exit 0;
!
  fi

  if expr $? != 0 > /dev/null; then
    echo table $table failed creation.
    exit 1
  fi
done
```

Buildcreatets.sh

```
#!/usr/bin/sh

#make sure to spool to log directory so we don't leave a mess

if expr x$tpcc_listfiles != xt > /dev/null; then
  cd $tpcc_log_dir
fi

#save to here
createtsout=${tpcc_genscripts_dir}/createts.sh
rm $createtsout
echo \#created automatically by $0 `date` > $createtsout

#tp- get table param.  (that is, $tpcc_tablename_tableparam)
tp(){
  eval echo \$tpcc_$1_$2
}

#numpart- given imp, parts, return 1 if not partitioned, parts otherwise
numpart(){
  case $1 in
  parttable) echo $2 ;;
```

```
  partindex) echo $2 ;;
  partiot) echo $2 ;;
  *) echo 1 ;;
  esac
}

if expr $tpcc_ordr_imp = queue > /dev/null; then
  tpcc_ordl_imp=none
  tpcc_ordr_size=`$tpcc_bcexpr $tpcc_ordr_size + $tpcc_ordl_size`
  tpcc_ordr_growth=`$tpcc_bcexpr $tpcc_ordr_growth + $tpcc_ordl_growth`
fi

#count total number of files, needed for createts
totalfiles=0

#figure out max for temp- if no indices (or really small) use min index size
curmaxindexsize=`$tpcc_bcexpr $tpcc_tempts_min / 2`
#note temp must come last since its size is based on the indices
for table in $tpcc_table_list $tpcc_index_list temp; do
# make sure we have all params
  if $tpcc_require tpcc_kit tpcc_${table}_imp tpcc_${table}_size tpcc_${table}_nf
tpcc_${table}_ext tpcc_${table}_growth tpcc_${table}_bs; then
    exit 0
  fi

  tableimp=`tp $table imp`

  if test $tableimp = none > /dev/null; then
    echo No tablespace created for $table.
  else
    echo Creating tablespace for: $table
# 1 - figure out the size of the tablespace, and the extent growth type

# first, get base size from params
    tssize=`tp $table size`
    tssize=`$tpcc_tokilobytes $tssize`
# scale up base size (for all sizes that grow)
# cases based on runlength, hist ordr, nord, ordl iordr1, iordr2, iordl, inord,
# and cases that do not scale at all (item, iitem):
    growthtype=`tp $table growth`
    if expr $growthtype != regular > /dev/null; then
      echo Growable table
      #we want a growable extent size (that is, a reasonable size) except for item/iitem/temp
      if expr $growthtype != 0 > /dev/null; then
        tssize=`$tpcc_bcexpr \( $tssize \* $tpcc_scale \) + \( \( $tpcc_scale \* 13 \* $tpcc_runlen \*
$growthtype \) / 1000 \)`
        extentgrowth=t
      else
        extentgrowth=f
      fi
    else
      echo Fixed, scaled table
      extentgrowth=f
      tssize=`$tpcc_bcexpr $tssize \* $tpcc_scale`
    fi
    # 5% miscellany overhead
    tssize=`$tpcc_bcexpr \( $tssize \* 105 \) / 100`

    # save maximum if type is index
    if test $tableimp = index -o $tableimp = partindex > /dev/null; then
      if $tpcc_isneg `$tpcc_bcexpr $curmaxindexsize - $tssize`; then
        curmaxindexsize=$tssize
      fi
    fi
    # set if temp
echo cur max is $curmaxindexsize
    if test $table = temp > /dev/null; then
      tssize=`$tpcc_bcexpr $curmaxindexsize \* 2`
    fi

    echo Base size after scale: $tssize k

# 2 - Calculate the number of files, and the size of each file.
    numfiles=`tp $table nf`
# typically for linux, but this may be useful for other things
    if test -z "`tp $table tsfileinc`"; then
      #default
      fileincrement=`$tpcc_lcm $tpcc_np $tpcc_ldrive`
    else
      #regardless of number of drives and partitions, just try to
      #keep the number of files down, or a fixed amount.
      fileincrement=`tp $table tsfileinc`
    fi


    if expr x$numfiles = xcalc \| x$numfiles = x > /dev/null; then
      # start with the lowest possible number of files, and go up
      numfiles=$fileincrement
    fi

    # calculate our max table filesize and max extent size
    tablebs=`tp $table bs`
    if expr x$tablebs = xauto > /dev/null; then
      max_fsize=$tpcc_fsize_limit_k
    else
      bskilo=`$tpcc_tokilobytes $tablebs`
      # tpcc_fsize_limit_k is for 2k, so divide bskilo/2 and multiply to
      # find the limit for us
      max_fsize=`$tpcc_bcexpr \( $bskilo \* $tpcc_fsize_limit_k \) / 2`
    fi

    # keep under maximum filesize
    while $tpcc_isneg $max_fsize - \( $tssize / $numfiles \); do
      numfiles=`$tpcc_bcexpr $numfiles + $fileincrement`
    done
    filesize=`$tpcc_bcexpr $tssize / $numfiles`

  echo got filesize as $filesize

    # make files at least 10m each, so the extent ends up 1m.
    if $tpcc_isneg `$tpcc_bcexpr $filesize - 1024`; then
      filesize=1024;
    fi

# 3 - Calculate the sextent size of each file
    if expr $extentgrowth = t > /dev/null; then
      if $tpcc_isneg `$tpcc_bcexpr 1024 \* 1024 \* 1024 - $filesize`; then
        #100 meg extent for large files
        extentsize=`$tpcc_bcexpr 1024 \* 1024 \* 100`
      else
        #small case not as important, just do 1/10 the expected size
        extentsize=`$tpcc_bcexpr \( $filesize + 9 \) / 10`
      fi
    else
      # fixed size table, make the extent size as large as possible.
      extentsub=`$tpcc_bcexpr $filesize / 10`
      # if subtracting more than a meg, just subtract a meg, since we don't need more than that
overhead.
      if $tpcc_isneg `$tpcc_bcexpr $extentsub - 1024`; then
        extentsize=`$tpcc_bcexpr $filesize - \( $filesize / 10 \)`
      else
        extentsize=`$tpcc_bcexpr $filesize - 1024`
      fi

      if $tpcc_isneg `$tpcc_bcexpr $tpcc_extent_limit_k - $extentsize`; then
        extentsize=$tpcc_extent_limit_k
      fi
    fi
    echo Table $table tssize $tssize k, files $numfiles, file size $filesize k, ext size $extentsize k
    roundtssize=`$tpcc_fromkilobytes $tssize`
    roundfisize=`$tpcc_fromkilobytes $filesize`
    roundexsize=`$tpcc_fromkilobytes $extentsize`

# 4- Add call createts to log file
    tablebs=`tp $table bs`
    # don't run tokilo on tablebs, since it can be 'auto'
    autospace=`tp $table autospace`

    # only give partitions if we are using a partitioned table,
    # otherwise tablespaces will be created that will be unused.
    tableparts=`numpart $tableimp $tpcc_np`
   cat >> $createtsout <<!
# Tablespace $table, ts size $roundtssize (${tssize}K)
# each file aprrox $roundfisize (${filesize}K)
# extents approx $roundexsize (${extentsize}K)
# $numfiles files
\$tpcc_createts $table $numfiles $tableparts \
      ${filesize}K ${extentsize}K $tpcc_os `expr $table = temp` \
      $totalfiles $tpcc_cpu $tablebs $autospace
    if expr \$? != 0 > /dev/null; then
      echo Creating tablespace for $table failed.  Exiting.
      exit 0
    fi
!

    totalfiles=`expr $totalfiles + $numfiles`

  fi  #imp != null
done
exit 0
```

Buildfixoo.sh

```
#!/bin/sh
if test $tpcc_ordl_imp = queue; then
${tpcc_scripts}/evenload.sh fixordrordl $tpcc_scale b e o f '$tpcc_updateordrordl'
else
echo ' ' > $tpcc_fixordrordl
fi
```

Buildloadcust.sh

```
#!/usr/bin/sh
```

```
${tpcc_scripts}/evenload.sh cust $tpcc_scale b e c
exit $?
```

Buildloaddist.sh

```
#!/bin/sh
echo 'cd $tpcc_bench' > $tpcc_genscripts_dir/loaddist.sh
echo "\$tpcc_load -M \$tpcc_scale -d > loaddist.log 2>&1" >> $tpcc_genscripts_dir/loaddist.sh
exit $?
```

Buildloaditem.sh

```
#!/bin/sh
echo 'cd $tpcc_bench' > $tpcc_genscripts_dir/loaditem.sh
echo "\$tpcc_load -M \$tpcc_scale -i > loaditem.log 2>&1" >> $tpcc_genscripts_dir/loaditem.sh
exit $?
```

Buildloadnord.sh

```
#!/bin/sh
${tpcc_scripts}/evenload.sh nord $tpcc_scale b e n
exit $?
```

Buildloadordrordl.sh

```
#!/bin/sh
${tpcc_scripts}/evenload.sh ordrordl $tpcc_scale b e o t
exit $?
```

Buildloadstok.sh

```
#!/bin/sh
${tpcc_scripts}/evenload.sh stok 100000 j k S
exit $?
```

Buildloadware.sh

```
#!/bin/sh
echo 'cd $tpcc_bench' > $tpcc_genscripts_dir/loadware.sh
echo "\$tpcc_load -M \$tpcc_scale -w > loadware.log 2>&1" >>
$tpcc_genscripts_dir/loadware.sh
exit $?
```

Consist.sh

```
#!/bin/sh
#
# $Header: consist.sh 01-jun-98.19:06:11 skareenh Exp $
#
# consist.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      consist.sh
#
#    DESCRIPTION
#                  Perform TPC-C consistency tests.
#
#    NOTES
#                  consist.sh [options]
#
#                  options:
#         -o <output file>        : name of output file
#         -h or -help             : print list of arguments
#         -m <max w_id>           : maximum warehouse id in database
#         -net "list of aliases"  : list of SQL*Net V2 connect string aliases
#         -wids "list of w_id's"  : list of warehouse ids for sampling
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#


#
# Defaults
#
OFILE="const.v1"
MULT=
SNET=
WIDS=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -m <max w_id>          : maximum warehouse id in database"
  echo "  -net \"list of aliases\" : list of SQL*Net V2 connect string aliases"
  echo "  -wids \"list of w_id's\" : list of warehouse ids for sampling"
      exit 0
      ;;
    -m|-mult)
      shift
      if [ "$1" != "" ]
      then
        MULT=$1
        shift
      fi
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        SNET=$1
        shift
      fi
      ;;
    -wids)
      shift
      if [ "$1" != "" ]
      then
        WIDS=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
  echo "Error: output file is not specified"
  echo "Use -help option to see correct usage."
  exit 1
fi

NHOSTS=0
for HOST in $SNET
do
  NHOSTS=`expr $NHOSTS + 1`
done

NWIDS=0
for WID in $WIDS
do
  NWIDS=`expr $NWIDS + 1`
done

if [ $NHOSTS != 0 -a $NWIDS != 0 -a $NHOSTS != $NWIDS ]
then
  echo "Error: # of SQL*Net V2 aliases != # of samples"
  echo "Use -help option to see correct usage."
  exit 1
fi

#
# By default, if no SQL*Net V2 string is specified, all checks are
# run on the local node.  If no sampling w_id's are specified, then
```

```
# 4 samples will be generated automatically.
#
if [ $NHOSTS = 0 -a $NWIDS = 0 ]
then
  if [ "$MULT" = "" ]
  then
    echo "Error: max warehouse id in this database is not specified"
    echo "Use -help option to see correct usage."
    exit 1
  elif [ $MULT = 1 ]
  then
    NWIDS=1
    WIDS="1"
  elif [ $MULT = 2 ]
  then
    NWIDS=2
    WIDS="1 2"
  elif [ $MULT = 3 ]
  then
    NWIDS=3
    WIDS="1 2 3"
  else
    NWIDS=4
    WIDS="1"
    WPERN=`expr $MULT / 4`
    SW=`expr $WPERN + 1`
    EW=`expr 2 \* $WPERN`
    TID=`expr $SW + $EW`
    TID=`expr $TID / 2`
    WIDS="$WIDS $TID"
    SW=`expr $EW + 1`
    EW=`expr 3 \* $WPERN`
    TID=`expr $SW + $EW`
    TID=`expr $TID / 2`
    WIDS="$WIDS $TID $MULT"
  fi
fi

#
# If SQL*Net V2 strings are specified, but no sampling w_id's are given,
# 1 sample per SQL*Net V2 string will be generated automatically.
#

if [ $NHOSTS != 0 -a $NWIDS = 0 ]
then
  EVENW=`expr $MULT % $NHOSTS`
  if [ $EVENW != 0 ]
  then
    echo "Error: # of warehouses cannot be evenly divided by # of nodes"
    echo "       so cannot generate sampling w_id's automatically"
    echo "Use -help option to see correct usage."
    exit 1
  fi
  WPERN=`expr $MULT / $NHOSTS`
  WIDS="1"
  I=2
  SW=`expr $WPERN + 1`
  EW=`expr 2 \* $WPERN`
  while [ $I -lt $NHOSTS ]
  do
    TID=`expr $SW + $EW`
    TID=`expr $TID / 2`
    WIDS="$WIDS $TID"
    I=`expr $I + 1`
    SW=`expr $SW + $WPERN`
    EW=`expr $EW + $WPERN`
  done
  WIDS="$WIDS $MULT"
  NWIDS=$NHOSTS
fi

#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# run consistency checks
#
date | tee ${ODIR}/${OFILE}
#${SDIR}/swt_temp.sh temp

cat >> ${ODIR}/${OFILE} <<!



Consistency Check 1, 3 and Order Count
--------------------------------------

!

sqlplus tpcc/tpcc @${SQLDIR}/consist1 | tee -a ${ODIR}/${OFILE}

I=1
while [ $I -le $NWIDS ]
do
  NSWID=1
  for DUMSWID in $WIDS
  do
    if [ $NSWID = $I ]
    then
      SWID=$DUMSWID
      break;
    fi
    NSWID=`expr $NSWID + 1`
  done

  if [ $NHOSTS = 0 ]
  then
    sqlplus tpcc/tpcc @${SQLDIR}/consist $I $SWID | tee ${ODIR}/${OFILE}.$I &
  else
    NCSTR=1
    for DUMCSTR in $SNET
    do
      if [ $NCSTR = $I ]
      then
        CSTR=$DUMCSTR
        break;
      fi
      NCSTR=`expr $NCSTR + 1`
    done
    sqlplus tpcc/tpcc@$CSTR @${SQLDIR}/consist $I $SWID | \
          tee ${ODIR}/${OFILE}.$I &
  fi
  I=`expr $I + 1`
done

wait

I=1
while [ $I -le $NWIDS ]
do
  cat >> ${ODIR}/${OFILE} <<!



Sample $I of Consistency Check 2 and 4
--------------------------------------

!

  cat ${ODIR}/${OFILE}.$I >> ${ODIR}/${OFILE}
  rm -f ${ODIR}/${OFILE}.$I
  I=`expr $I + 1`
done

#${SDIR}/swt_temp.sh system
date | tee -a ${ODIR}/${OFILE}
```

Consist.sql

```
Rem
Rem $Header: consist.sql 01-jun-98.19:15:43 skareenh Exp $
```

```
Rem
Rem consist.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      consist.sql
Rem
Rem    DESCRIPTION
Rem       Does spot checking for consistency condidtions 2 and 4
Rem       for TPC-C database.
Rem
Rem
Rem    NOTES
Rem               Usage:   sqlplus tpcc/tpcc @consist <sample #> <warehouse id>
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem


set pagesize 100
set termout on
set echo on
set timing on


rem consistency condition 2

drop table temp_o1_&&1;
drop table temp_no_&&1;
create table temp_o1_&&1 (
   o_w_id  integer,
   o_d_id  integer,
   o_o_id  integer);
create table temp_no_&&1 (
   no_w_id integer,
   no_d_id integer,
   no_o_id integer);

insert into temp_o1_&&1 select o_w_id, o_d_id, max(o_id) from ordr
   where o_w_id = &&2 group by o_w_id, o_d_id;
insert into temp_no_&&1 select no_w_id, no_d_id, max(no_o_id) from nord
   where no_w_id = &&2 group by no_w_id, no_d_id;

select d_w_id, d_id, (d_next_o_id - o_o_id - 1)
   from dist, temp_o1_&&1, temp_no_&&1
   where d_w_id = o_w_id and d_w_id = no_w_id
   and d_id = o_d_id and d_id = no_d_id;
select d_w_id, d_id, (d_next_o_id - no_o_id - 1)
   from dist, temp_o1_&&1, temp_no_&&1
   where d_w_id = o_w_id and d_w_id = no_w_id
   and d_id = o_d_id and d_id = no_d_id;

commit;


rem consistency condition 4

drop table temp_o2_&&1;
drop table temp_ol_&&1;
create table temp_o2_&&1 (
   o_w_id  integer,
   o_d_id  integer,
   o_count integer);
create table temp_ol_&&1 (
   ol_w_id  integer,
   ol_d_id  integer,
   ol_count integer);

insert into temp_o2_&&1 select o_w_id, o_d_id, sum(o_ol_cnt) from ordr
   where o_w_id = &&2 group by o_w_id, o_d_id;
insert into temp_ol_&&1 select /*+ parallel(ordl, 30) */ ol_w_id, ol_d_id, count(*) from ordl
   where ol_w_id = &&2 group by ol_w_id, ol_d_id;

select o_w_id, o_d_id, (o_count - ol_count) from temp_o2_&&1, temp_ol_&&1
   where o_w_id = ol_w_id and o_d_id = ol_d_id;

commit;

quit;
```

Consist1.sql

```
Rem
Rem $Header: consist1.sql 01-jun-98.19:16:00 skareenh Exp $
Rem
Rem consist1.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      consist1.sql
Rem
Rem    DESCRIPTION
rem       Does consistency check 1 and 3 for TPC-C database.
Rem
Rem    NOTES
rem               Usage:   sqlplus tpcc/tpcc @consist1
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem


set pagesize 100
set termout on
set echo on
set timing on


rem consistency condition 1

select w_id, w_ytd, (w_ytd - sum(d_ytd)) from ware, dist
   where d_w_id = w_id group by w_id, w_ytd having w_ytd - sum(d_ytd) <> 0;

rem consistency condition 3

select /*+ parallel(nord, 32) */ no_w_id, no_d_id, ((max(no_o_id) - min(no_o_id) + 1) - count(*))
   from nord
   group by no_w_id, no_d_id
   having ((max(no_o_id) - min(no_o_id) + 1) - count(*) <> 0);


rem compared to previous value to verify the number of orders processed

select sum(d_next_o_id) from dist;

quit;
```

Consist3.sql

```
Rem
Rem $Header: consist1.sql 01-jun-98.19:16:00 skareenh Exp $
Rem
Rem consist1.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      consist1.sql
Rem
Rem    DESCRIPTION
rem       Does consistency 3 for TPC-C database.
Rem
Rem    NOTES
rem               Usage:   sqlplus tpcc/tpcc @consist3
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem


set pagesize 100
set termout on
set echo on
set timing on


rem consistency condition 3

select /*+ parallel(nord, 32) */ no_w_id, no_d_id, ((max(no_o_id) - min(no_o_id) + 1) - count(*))
   from nord
   group by no_w_id, no_d_id
   having ((max(no_o_id) - min(no_o_id) + 1) - count(*) <> 0);


rem compared to previous value to verify the number of orders processed

select sum(d_next_o_id) from dist;

quit;
```

Cre_tab.sql

```
rem
rem
================================================================+
rem        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
rem                OPEN SYSTEMS PERFORMANCE GROUP                  |
```

```
rem                    All Rights Reserved                    |
rem
=================================================================+
rem  FILENAME
rem       cre_tab.sql
rem  DESCRIPTION
rem       Create temporary tables for consistency tests.
rem ================================================================
rem
rem Usage:   sqlplus tpcc/tpcc @cre_tab
rem

connect tpcc/tpcc;
set echo on;

drop table temp_o1;
drop table temp_no;
drop table temp_o2;
drop table temp_ol;
drop table tpcc_audit_tab;

create table temp_o1 (
  o_w_id  integer,
  o_d_id  integer,
  o_o_id  integer);

create table temp_no (
  no_w_id integer,
  no_d_id integer,
  no_o_id integer);

create table temp_o2 (
  o_w_id  integer,
  o_d_id  integer,
  o_count integer);

create table temp_ol (
  ol_w_id  integer,
  ol_d_id  integer,
  ol_count integer);

create table tpcc_audit_tab (starttime date);

delete from tpcc_audit_tab;

set echo off;
```

Create_cache.views.sql

```
rem This script creates four views that when queried will return
rem the total number of buffers in the buffer cache and the total
rem number of cloned buffers from each of the database's tablespaces.
rem
rem This assumes that each table and index is in its own tablespace.
rem If this is not the case, another query can be used which uses the
rem database's object tables to decipher the different objects.  However,
rem this query is slower.
rem
rem This script assumes 7.3.x.  If you are using V7.2.x or below, please
rem replace svrmgrl with sqldba lmode=y.
rem
rem Modification History:
rem
rem wbattist      16-Jun-1996     Create two additional views to keep
rem                               track of the number of clones in each
rem                               tablespace.
rem
rem wbattist      24-May-1995     Add the state check for the cbf view
rem                               to ensure that cloned blocks are not
rem                               counted.
rem

connect $oracle_dba/$oracle_dba_password;
set echo on;
drop view cbf;
create view cbf as
 select distinct(dbarfil) file#, count(1) blocks
 from x$bh
  where dbarfil > 0 and state <> 3
 group by dbarfil;
drop view cbt;
create view cbt as
  select ts$.name name,sum(cbf.blocks) buffers
  from cbf, file$, ts$
   where cbf.file#=file$.file# and file$.ts#=ts$.ts#
  group by file$.ts#, ts$.name;
drop view cbfcln;
create view cbfcln as
 select distinct(dbarfil) file#, count(1) blocks
 from x$bh
  where dbarfil > 0
 group by dbarfil;
drop view cbtcln;
create view cbtcln as
  select ts$.name name,sum(cbfcln.blocks) buffers
  from cbfcln, file$, ts$
   where cbfcln.file#=file$.file# and file$.ts#=ts$.ts#
  group by file$.ts#, ts$.name;

set echo off;
```

Createdb.sql

```
/* created automatically by  /ORACLE/tpcc6700.100402.16K/scripts/buildcreatedb.sh Fri Oct 4
16:40:56 PDT 2002 */
spool createdb.log

set echo on

startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxinstances 1
  datafile '$tpcc_disks_location/system_001' size 200M reuse
  logfile '$tpcc_disks_location/log_1' size 6720M reuse,
          '$tpcc_disks_location/log_2' size 6720M reuse
  sysaux datafile '$tpcc_disks_location/aux.df' size 120M reuse;

create undo tablespace undo_ts datafile
   '$tpcc_disks_location/roll01' size 13400M reuse blocksize 8K;

set echo off
exit sql.sqlcode
```

Createindex_icust1.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:37 PDT 2002 */
    set sqlblanklines on
    spool createindex_icust1.log ;
    set echo on ;
    drop index icust1 ;
      create unique index icust1 on cust ( c_w_id
, c_d_id
, c_id )
  pctfree 1  initrans 3
  parallel 12
  storage ( buffer_pool default )
  tablespace icust1_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

Createindex_icust2.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:39 PDT 2002 */
    set sqlblanklines on
    spool createindex_icust2.log ;
    set echo on ;
    drop index icust2 ;
      create unique index icust2 on cust ( c_last
, c_w_id
, c_d_id
, c_first
, c_id )
  pctfree 1  initrans 3
  parallel 12
  storage ( buffer_pool default )
  tablespace icust2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

Createindex_idist.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:41 PDT 2002 */
    set sqlblanklines on
    spool createindex_idist.log ;
    set echo on ;
    drop index idist ;
      create unique index idist on dist ( d_w_id
```

```
, d_id )
  pctfree 5  initrans 3
  storage ( buffer_pool default )
  tablespace idist_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

Createindex_item.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:45 PDT 2002 */
    set sqlblanklines on
    spool createindex_iitem.log ;
    set echo on ;
    drop index iitem ;
      create unique index iitem on item ( i_id )
  pctfree 5  initrans 4
  storage ( buffer_pool default )
  tablespace iitem_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

Createindex_inord.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:51 PDT 2002 */
  exit 0;
```

Createindex_iordl.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:50 PDT 2002 */
  exit 0;
```

Createindex_inord.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:51 PDT 2002 */
  exit 0;
```

Createindex_iordl.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:50 PDT 2002 */
  exit 0;
```

Createindex_iordr2.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:48 PDT 2002 */
    set sqlblanklines on
    spool createindex_iordr2.log ;
    set echo on ;
    drop index iordr2 ;
      create unique index iordr2 on ordr ( o_c_id
, o_d_id
, o_w_id
, o_id )
  pctfree 25  initrans 4
  parallel 12
  storage ( buffer_pool default )
  tablespace iordr2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

Createindex_istok.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:43 PDT 2002 */
    set sqlblanklines on
    spool createindex_istok.log ;
    set echo on ;
    drop index istok ;
      create unique index istok on stok ( s_i_id
, s_w_id )
  pctfree 1  initrans 3
  parallel 12
  storage ( buffer_pool default )
  tablespace istok_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

Createindex_iware.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreateindex.sh Fri Oct 4
16:41:35 PDT 2002 */
    set sqlblanklines on
    spool createindex_iware.log ;
    set echo on ;
    drop index iware ;
      create unique index iware on ware ( w_id )
  pctfree 1  initrans 3
  storage ( buffer_pool default )
  tablespace iware_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

Createmisc.sh

```
#!/usr/bin/sh

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool createmisc.log
set echo on;
alter user tpcc temporary tablespace system;
grant execute on dbms_lock to public;
grant execute on dbms_pipe to public;
grant select on v_\$parameter to public;

REM
REM begin plsql_mon.sql
REM

connect tpcc/tpcc;
set echo on;
CREATE OR REPLACE PACKAGE plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  );
END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  )
  IS
     s          NUMBER;
  BEGIN
    dbms_pipe.pack_message (info);
    s := dbms_pipe.send_message ('plsql_mon');
    IF (s <> 0) THEN
      raise_application_error (-20000, 'Error:' || to_char(s) ||
                               ' sending on pipe');
    END IF;
  END;
END;
/
show errors;

set echo off;

REM
REM end plsql_mon.sql
REM

REM
REM  begin cre_tab.sql
REM
```

```
connect tpcc/tpcc;
set echo on;

drop table temp_o1;
drop table temp_no;
drop table temp_o2;
drop table temp_ol;
drop table tpcc_audit_tab;

create table temp_o1 (
  o_w_id  integer,
  o_d_id  integer,
  o_o_id  integer);

create table temp_no (
  no_w_id integer,
  no_d_id integer,
  no_o_id integer);

create table temp_o2 (
  o_w_id  integer,
  o_d_id  integer,
  o_count integer);

create table temp_ol (
  ol_w_id  integer,
  ol_d_id  integer,
  ol_count integer);

create table tpcc_audit_tab (starttime date);

delete from tpcc_audit_tab;

set echo off;

REM
REM  end cre_tab.sql
REM

REM
REM  begin views.sql
REM

connect tpcc/tpcc;
set echo on;

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 cust c, ware 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 dist d, ware 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 stok s, item i
 where i.i_id = s.s_i_id;

set echo off;

REM
REM  end views.sql
REM


REM
REM  begin dml.sql
REM
connect tpcc/tpcc;
set echo on;

  alter table ware disable table lock;
  alter table dist disable table lock;
  alter table cust disable table lock;
  alter table hist disable table lock;
  alter table item disable table lock;
  alter table stok disable table lock;
  alter table ordr disable table lock;
  alter table nord disable table lock;
  alter table ordl disable table lock;

set echo off;

REM
REM  end dml.sql
REM

REM
REM  begin extent.sql
REM

$SYS_CONNECTION_STRING

@$tpcc_sql_dir/extent

@$tpcc_sql_dir/freeext

exit sql.sqlcode;

!
```

Createspacestats.sh

```
#!/usr/bin/sh
$tpcc_sqlplus $tpcc_dba_user_pass
@$tpcc_sql_dir/createspacestats > junk 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

Createspacestats.sql

```
spool createspacestats.log
@$tpcc_sql_dir/space_init
@$tpcc_sql_dir/space_get 12 10
@$tpcc_sql_dir/space_rpt
spool off
exit sql.sqlcode;
```

Createstats.sh

```
#!/usr/bin/sh

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

REM
REM create tablespace for statspack user sp begin
REM

spool createstats.log

set echo on
   drop tablespace sp including contents;
   create tablespace sp_0 datafile '$tpcc_disks_location/sp_0' size $tpcc_statspack_size reuse
autoextend on extent management local uniform size 1M segment space management auto
nologging ;
   spool off

REM
```

```
REM create tablespace for statspack user sp end
REM

REM
REM  begin now call spcreate to create statspack sp package
REM

$tpcc_internal_connect

@$ORACLE_HOME/rdbms/admin/spdrop
@$ORACLE_HOME/rdbms/admin/spcreate
perfstat
sp_0
temp_0


REM note that the last thing (after spcreate) is the perfstat password.
REM since we're not worried about security, perfstat will do.

REM
REM tpcc stat table for NT, it is not working so I comment it out
REM shui.lau@oracle.com it is better to use perfmon
REM

@$tpcc_sql_dir/cs_tpcc
@$tpcc_sql_dir/cs_cpu
@$tpcc_sql_dir/cs_os
@$tpcc_sql_dir/cs_proc
@$tpcc_sql_dir/cs_thread

REM
REM tpcc result table for unix and NT
REM

@$tpcc_sql_dir/c_stat
@$tpcc_sql_dir/pst_c

!
```

Createstoredproc.sh

```
#!/usr/bin/sh
$tpcc_sqlplus $tpcc_user_pass @${tpcc_sql_dir}/createstoredprocs > junk 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

Createsotredproc.sql

```
spool createstoredprocs.log
@$tpcc_sql_dir/tkvcinin.sql
spool off
exit sql.sqlcode;
```

Createtable_cust.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatetable.sh Fri Oct 4
16:41:00 PDT 2002 */
    set sqlblanklines on
    spool createtable_cust.log
    set echo on
      drop cluster custcluster including tables ;

create cluster custcluster (
  c_id number
, c_d_id number
, c_w_id number
  )
  single table
  hashkeys 201000000
  hash is ( (c_id * ( 6700 * 10 ) + c_w_id * 10 + c_d_id) )
  size 350
  pctfree 0  initrans 3
  storage ( buffer_pool keep )
  tablespace cust_0;

create table cust (
  c_id number
, c_d_id number
, c_w_id number
, c_discount number
, c_credit char(2)
, c_last varchar2(16)
, c_first varchar2(16)
, c_credit_lim number
, c_balance number
, c_ytd_payment number
, c_payment_cnt number
, c_delivery_cnt number
, c_street_1 varchar2(20)
, c_street_2 varchar2(20)
, c_city varchar2(20)
, c_state char(2)
, c_zip char(9)
, c_phone char(16)
, c_since date
, c_middle char(2)
, c_data varchar2(500)
)
cluster custcluster (
  c_id
, c_d_id
, c_w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

Createtable_dist.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatetable.sh Fri Oct 4
16:41:07 PDT 2002 */
    set sqlblanklines on
    spool createtable_dist.log
    set echo on
      drop cluster distcluster including tables ;

create cluster distcluster (
  d_id number
, d_w_id number
  )
  single table
  hashkeys 67000
  hash is ( ((d_w_id * 10) + d_id) )
  size 1536
    initrans 4
  storage ( buffer_pool default )
  tablespace dist_0;

create table dist (
  d_id number
, d_w_id number
, d_ytd number
, d_next_o_id number
, d_tax number
, d_name varchar2(10)
, d_street_1 varchar2(20)
, d_street_2 varchar2(20)
, d_city varchar2(20)
, d_state char(2)
, d_zip char(9)
)
cluster distcluster (
  d_id
, d_w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

Createtable_hist.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatetable.sh Fri Oct 4
16:41:11 PDT 2002 */
    set sqlblanklines on
    spool createtable_hist.log
    set echo on
      drop table hist ;

create table hist (
  h_c_id number
, h_c_d_id number
, h_c_w_id number
, h_d_id number
, h_w_id number
, h_date date
, h_amount number
, h_data varchar2(24)
)
```

```
  pctfree 5  initrans 4
  storage ( buffer_pool default )
  tablespace hist_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

Createtable_item.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatetable.sh Fri Oct 4
16:41:19 PDT 2002 */
    set sqlblanklines on
    spool createtable_item.log
    set echo on
      drop cluster itemcluster including tables ;

create cluster itemcluster (
  i_id number(6,0)
  )
  single table
  hashkeys 100000
  hash is ( i_id )
  size 120
  pctfree 0  initrans 3
  storage ( buffer_pool default )
  tablespace item_0;

create table item (
  i_id number(6,0)
, i_name varchar2(24)
, i_price number
, i_data varchar2(50)
, i_im_id number
)
cluster itemcluster (
  i_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

Createtable_nord.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatetable.sh Fri Oct 4
16:41:28 PDT 2002 */
    set sqlblanklines on
    spool createtable_nord.log
    set echo on
        drop cluster nordcluster_queue including tables ;

  create cluster nordcluster_queue (
    no_w_id number
, no_d_id number
, no_o_id number SORT
    )

    hashkeys 67000
    hash is ( (no_w_id - 1) * 10 + no_d_id - 1 )
    size 190
    tablespace nord_0;

  create table nord (
    no_w_id number
, no_d_id number
, no_o_id number SORT
    , constraint nord_uk primary key ( no_w_id
, no_d_id
, no_o_id )
  )
  cluster nordcluster_queue (
    no_w_id
, no_d_id
, no_o_id
  );
    set echo off
    spool off
    exit sql.sqlcode;
```

Createtable_ordl.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatetable.sh Fri Oct 4
16:41:28 PDT 2002 */
    set sqlblanklines on
    spool createtable_nord.log
    set echo on
        drop cluster nordcluster_queue including tables ;

  create cluster nordcluster_queue (
    no_w_id number
, no_d_id number
, no_o_id number SORT
    )

    hashkeys 67000
    hash is ( (no_w_id - 1) * 10 + no_d_id - 1 )
    size 190
    tablespace nord_0;

  create table nord (
    no_w_id number
, no_d_id number
, no_o_id number SORT
    , constraint nord_uk primary key ( no_w_id
, no_d_id
, no_o_id )
  )
  cluster nordcluster_queue (
    no_w_id
, no_d_id
, no_o_id
  );
    set echo off
    spool off
    exit sql.sqlcode;
```

Createtable_ordr.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatetable.sh Fri Oct 4
16:41:22 PDT 2002 */
    set sqlblanklines on
    spool createtable_ordr.log
    set echo on
        drop cluster ordrcluster_queue including tables ;

  create cluster ordrcluster_queue (
    o_w_id number
, o_d_id number
, o_id number SORT
, o_number number SORT
    )

    hashkeys 67000
    hash is ( (o_w_id - 1) * 10 + o_d_id - 1 )
    size 1490
    tablespace ordr_0;

  create table ordr (
    o_id number sort
, 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
    , constraint ordr_uk primary key ( o_w_id
, o_d_id
, o_id )
  )
  cluster ordrcluster_queue (
    o_w_id
, o_d_id
, o_id
  );
    set echo off
    spool off
    exit sql.sqlcode;
```

Createtable_stok.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatetable.sh Fri Oct 4
16:41:13 PDT 2002 */
    set sqlblanklines on
    spool createtable_stok.log
    set echo on
      drop cluster stokcluster including tables ;

create cluster stokcluster (
  s_i_id number
, s_w_id number
  )
  single table
  hashkeys 670000000
```

```
  hash is ( (s_i_id * 6700 + s_w_id) )
  size 350
  pctfree 0  initrans 3
  storage ( buffer_pool keep )
  tablespace stok_0;

create table stok (
  s_i_id number
, s_w_id number
, s_quantity number
, s_ytd number
, s_order_cnt number
, s_remote_cnt number
, s_data varchar2(50)
, s_dist_01 char(24)
, s_dist_02 char(24)
, s_dist_03 char(24)
, s_dist_04 char(24)
, s_dist_05 char(24)
, s_dist_06 char(24)
, s_dist_07 char(24)
, s_dist_08 char(24)
, s_dist_09 char(24)
, s_dist_10 char(24)
)
cluster stokcluster (
  s_i_id
, s_w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

Createtable_ware.sql

```
/* created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatetable.sh Fri Oct 4
16:40:57 PDT 2002 */
    set sqlblanklines on
    spool createtable_ware.log
    set echo on
      drop cluster warecluster including tables ;

create cluster warecluster (
  w_id number(5,0)
  )
  single table
  hashkeys 6700
  hash is ( (w_id) )
  size 1536
    initrans 2
  storage ( buffer_pool default )
  tablespace ware_0;

create table ware (
  w_id number(5,0)
, w_ytd number
, w_tax number
, w_name varchar2(10)
, w_street_1 varchar2(20)
, w_street_2 varchar2(20)
, w_city varchar2(20)
, w_state char(2)
, w_zip char(9)
)
cluster warecluster (
  w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

Createts.sh

```
#created automatically by /ORACLE/tpcc6700.100402.16K/scripts/buildcreatets.sh Fri Oct 4
16:40:35 PDT 2002
# Tablespace ware, ts size 13M (14070K)
# each file aprrox 13M (14070K)
# extents approx 12M (13046K)
# 1 files
$tpcc_createts ware 1 1       14070K 13046K unix 0       0 5 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for ware failed.  Exiting.
      exit 0
    fi
# Tablespace cust, ts size 201G (211500240K)
# each file aprrox 7G (8134624K)
# extents approx 1G (2097151K)
# 26 files
$tpcc_createts cust 26 1       8134624K 307200K unix 0       1 5 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for cust failed.  Exiting.
      exit 0
    fi
# Tablespace dist, ts size 137M (140700K)
# each file aprrox 137M (140700K)
# extents approx 136M (139676K)
# 1 files
$tpcc_createts dist 1 1       140700K 139676K unix 0       27 5 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for dist failed.  Exiting.
      exit 0
    fi
# Tablespace hist, ts size 33G (34987869K)
# each file aprrox 6G (6997573K)
# extents approx 683M (699758K)
# 5 files
$tpcc_createts hist 5 1       6997573K 699758K unix 0       28 5 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for hist failed.  Exiting.
      exit 0
    fi
# Tablespace stok, ts size 274G (288273195K)
# each file aprrox 7G (8236377K)
# extents approx 1G (2097151K)
# 41 files
$tpcc_createts stok 41 1       8236377K 2048000K unix 0       33 5 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for stok failed.  Exiting.
      exit 0
    fi
# Tablespace item, ts size 15M (15380K)
# each file aprrox 15M (15380K)
# extents approx 14M (14356K)
# 1 files
$tpcc_createts item 1 1       15380K 14356K unix 0       68 5 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for item failed.  Exiting.
      exit 0
    fi
# Tablespace ordr, ts size 545G (572121375K)
# each file aprrox 60G (63569041K)
# extents approx 6G (6356905K)
# 9 files
$tpcc_createts ordr 9 1       63569041K 6356905K unix 0       69 5 16K t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for ordr failed.  Exiting.
      exit 0
    fi
# Tablespace nord, ts size 6G (6959022K)
# each file aprrox 6G (6959022K)
# extents approx 679M (695903K)
# 1 files
$tpcc_createts nord 1 1       6959022K 695903K unix 0       78 5 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for nord failed.  Exiting.
      exit 0
    fi
# Tablespace iware, ts size 6M (7035K)
# each file aprrox 6M (7035K)
# extents approx 6M (6332K)
# 1 files
$tpcc_createts iware 1 1       7035K 6332K unix 0       79 5 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for iware failed.  Exiting.
      exit 0
    fi
# Tablespace icust1, ts size 4G (5177760K)
# each file aprrox 4G (5177760K)
# extents approx 1G (2097151K)
# 1 files
$tpcc_createts icust1 1 1       5177760K 204800K unix 0       80 5 16K t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for icust1 failed.  Exiting.
      exit 0
    fi
# Tablespace icust2, ts size 10G (11270070K)
# each file aprrox 10G (11270070K)
# extents approx 1G (2097151K)
# 1 files
$tpcc_createts icust2 1 1       11270070K 204800K unix 0       81 5 16K t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for icust2 failed.  Exiting.
      exit 0
    fi
# Tablespace idist, ts size 27M (28140K)
# each file aprrox 27M (28140K)
# extents approx 26M (27116K)
# 1 files
$tpcc_createts idist 1 1       28140K 27116K unix 0       82 5 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for idist failed.  Exiting.
      exit 0
    fi
```

```
# Tablespace istok, ts size 14G (14703150K)
# each file aprrox 14G (14703150K)
# extents approx 1G (2097151K)
# 1 files
$tpcc_createts istok 1 1       14703150K 204800K unix 0       83 5 16K t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for istok failed.  Exiting.
     exit 0
   fi
# Tablespace iitem, ts size 2M (2150K)
# each file aprrox 2M (2150K)
# extents approx 1M (1935K)
# 1 files
$tpcc_createts iitem 1 1       2150K 1935K unix 0       84 5 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for iitem failed.  Exiting.
     exit 0
   fi
# Tablespace iordr2, ts size 15G (16764405K)
# each file aprrox 15G (16764405K)
# extents approx 1G (1676441K)
# 1 files
$tpcc_createts iordr2 1 1       16764405K 204800K unix 0       85 5 16K t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for iordr2 failed.  Exiting.
     exit 0
   fi
# Tablespace temp, ts size 31G (33528810K)
# each file aprrox 7G (8382202K)
# extents approx 1G (2097151K)
# 14 files
$tpcc_createts temp 4 1       8382202K 2097151K unix 1       86 5 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for temp failed.  Exiting.
     exit 0
   fi
```

Createuser.sh

```
#!/usr/bin/sh

echo Creating user tpcc...
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_sql_dir/createuser > junk 2>&1
if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

Createuser.sql

```
spool createusertpcc.log;

set echo on;

create user tpcc identified by tpcc;

grant dba to tpcc;

set echo off;
spool off;

exit ;
```

Cust_uniq.sql

```
Rem
Rem $Header: dbcheck.sql 01-jun-98.19:16:35 skareenh Exp $
Rem
Rem dbcheck.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      dbcheck.sql
Rem
Rem    DESCRIPTION
rem       Check TPC-C database.
Rem
Rem    NOTES
rem               Usage:   sqlplus tpcc/tpcc @dbcheck <max_warehouse_id>
rem               Note:    May need to modify degree of parallelism for table scans
rem          in order to speed up the run time of this script.
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem

set pagesize 100
set termout on
set echo on
set timing on

column w_tax         format          9.9999
column w_ytd         format 9999999999.99
column d_tax         format          9.9999
column d_ytd         format 9999999999.99
column c_credit_lim  format 9999999999.99
column c_discount    format          9.9999
column c_balance     format 9999999999.99
column c_ytd_payment format 9999999999.99
column h_amount      format       9999.99
column ol_amount     format       9999.99
column i_price       format       9999.99


REM get timestamp

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

insert into cust (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 (2345,9,9,'x','x','x','x','x','x','x','x','x',
        sysdate,'x',9,.9999,9,9,9,9,'x');

exit;
```

Dbcheck.sh

```
#!/bin/sh
#
# $Header: dbcheck.sh 01-jun-98.19:06:29 skareenh Exp $
#
# dbcheck.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      dbcheck.sh
#
#    DESCRIPTION
#                  Check integrity of database.
#
#    NOTES
#                  dbcheck.sh [max_#_warehouse [output_file]]
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#    rkambo      07/29/99 - Modification
#


if [ $1 ]
then
   MAXWARE=$1
else
   MAXWARE=1200
fi

if [ $2 ]
then
   OFILE="$2"
else
   OFILE="dbcheck.v1"
fi

if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
    SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
```

```
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

date | tee ${ODIR}/${OFILE}

#${SDIR}/swt_temp.sh temp

#sqlplus system/manager <<!
#   alter tablespace temp permanent;
#   quit;
#!

sqlplus tpcc/tpcc @${SQLDIR}/dbcheck $MAXWARE > ${ODIR}/${OFILE}

#sqlplus system/manager <<!
#   alter tablespace temp temporary;
#   quit;
#!

#${SDIR}/swt_temp.sh system
date | tee -a ${ODIR}/${OFILE}
```

Dbcheck.sql

```
Rem
Rem $Header: dbcheck.sql 01-jun-98.19:16:35 skareenh Exp $
Rem
Rem dbcheck.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      dbcheck.sql
Rem
Rem    DESCRIPTION
rem      Check TPC-C database.
Rem
Rem    NOTES
rem              Usage:   sqlplus tpcc/tpcc @dbcheck <max_warehouse_id>
rem              Note:    May need to modify degree of parallelism for table scans
rem        in order to speed up the run time of this script.
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem
Rem              Lorna Livingtree 11/8/02 - Modified
Rem                                 added select out of sample table.
Rem                                 made inserts maximum size
Rem

set pagesize 100
set termout on
set echo on
set timing on

column w_tax        format         9.9999
column w_ytd        format 9999999999.99
column d_tax        format         9.9999
column d_ytd        format 9999999999.99
column c_credit_lim  format 9999999999.99
column c_discount    format         9.9999
column c_balance     format 9999999999.99
column c_ytd_payment format 9999999999.99
column h_amount      format       9999.99
column ol_amount     format       9999.99
column i_price       format       9999.99


REM get timestamp

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM C_LAST distribution

drop table temp_c_last;
create table temp_c_last (cnt) unrecoverable
   storage (initial 50M next 50M pctincrease 0) as
   select /*+ full(cust) parallel(cust,20) */
        count(*) from cust group by c_w_id, c_d_id, c_last;
select /*+ full(temp_c_last) parallel(temp_c_last,10) */
       cnt, count(*) from temp_c_last
   group by cnt order by cnt;

drop table temp_c_last;
commit;

REM C_LAST sampling

drop table temp_sample_w;
create table temp_sample_w (w_id number);
insert into temp_sample_w values (1);
insert into temp_sample_w values (round(0.33 * &&1));
insert into temp_sample_w values (round(0.66 * &&1));
insert into temp_sample_w values (&&1);
commit;

REM LLL 11/8/02
select * from temp_sample_w;

select c_last from cust
   where c_w_id in (select * from temp_sample_w);

commit;

REM S_DATA distribution

select /*+ full(stok) parallel(stok,50) */
   count(*) from stok where s_data like '%ORIGINAL%';


REM I_DATA distribution

select count(*) from item where i_data like '%ORIGINAL%';


REM C_CREDIT distribution

select /*+ full(cust) parallel(cust,50) */
   count(*) from cust where c_credit = 'BC';


REM O_OL_CNT distribution

select /*+ full(ordr) parallel(ordr,16) */
   o_ol_cnt, count(*) from ordr group by o_ol_cnt order by o_ol_cnt;
select /*+ full(ordr) parallel(ordr,16) */ avg(o_ol_cnt) from ordr;


REM INSERT tests with duplicate values

insert into item (i_id, i_im_id, i_name, i_price, i_data)
   values (50000, 1, 'x', 9, 'x');
insert into ware (w_id, w_name, w_street_1, w_street_2,
                  w_city, w_state, w_zip, w_tax, w_ytd)
   values (9,'x','x','x','x','x','x',.9999,9);
insert into stok (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 (1234,9,9,'x','x','x','x','x','x','x','x','x','x',9,9,9,'x');

insert into dist (d_id, d_w_id, d_name, d_street_1, d_street_2, d_city,
                  d_state, d_zip, d_tax, d_ytd, d_next_o_id)
                   values (10,9,'x','x','x','x','x','x',.9999,9,9);

insert into cust (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 (2345,9,9,'x','x','x','x','x','x','x','x','x',
         sysdate,'x',9,.9999,9,9,9,'x');

insert into ordl (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                  ol_supply_w_id, ol_delivery_d, ol_quantity,
                  ol_amount, ol_dist_info)
   values (1456,9,9,1,9,9,NULL,9,9,'x');
insert into ordr (o_id, o_d_id, o_w_id, o_c_id, o_entry_d, o_carrier_id,
                  o_ol_cnt, o_all_local)
   values (345,9,9,9,sysdate,9,9,9);
insert into nord (no_o_id, no_d_id, no_w_id)
   values (2468,9,9);


REM INSERT tests with new values
REM LLL 11/8/02 maximized column content

insert into item (i_id, i_im_id, i_name, i_price, i_data)
   values (200000, 200000, 'abcdefghijklmnopqrstuvwx', 12345,
'x23456789x123456789x123456789x123456789x123456789x');
select /*+ index(item, iitem) */ * from item where i_id = 200000;
delete /*+ index(item, iitem) */ from item where i_id = 200000;


REM w_id must be double of max(w_id)

insert into ware (w_id, w_name, w_street_1, w_street_2,
                  w_city, w_state, w_zip, w_tax, w_ytd )
   values (2 * &&1,'x23456789x','x23456789x123456789x','x23456789x123456789x',
```

```
'x23456789x123456789x','LL','916496151',.1234, 123456789012);
select /*+ index(ware, iware) */ * from ware where w_id = 2 * &&1;
delete /*+ index(ware, iware) */ from ware where w_id = 2 * &&1;

REM s_w_id must be double of max(w_id)

insert into stok (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 (200000,2 * &&1,1234,
'xxxxxxxxxxxxxxxxxxxxxxxx','xxxxxxxxxxxxxxxxxxxxxxxx','xxxxxxxxxxxxxxxxxxxxxxxx',
'xxxxxxxxxxxxxxxxxxxxxxxx','xxxxxxxxxxxxxxxxxxxxxxxx','xxxxxxxxxxxxxxxxxxxxxxxx',
'xxxxxxxxxxxxxxxxxxxxxxxx','xxxxxxxxxxxxxxxxxxxxxxxx','xxxxxxxxxxxxxxxxxxxxxxxx',
'xxxxxxxxxxxxxxxxxxxxxxxx',12345678,1234,1234,
'01234567890123456789012345678901234567890123456789');
select /*+ index(stok, istok) */ * from stok where s_i_id=200000 and s_w_id = 2 * &&1;
delete /*+ index(stok, istok) */ from stok where s_i_id=200000 and s_w_id = 2 * &&1;


REM d_w_id must be double of max(w_id)

insert into dist (d_id, d_w_id, d_name, d_street_1, d_street_2, d_city,
                  d_state, d_zip, d_tax, d_ytd, d_next_o_id)
   values (20,2 *
&&1,'xxxxxxxxxx','xxxxxxxxxxxxxxxxxxxx','xxxxxxxxxxxxxxxxxxxx',
'xxxxxxxxxxxxxxxxxx','LL','x23456789',.1234,999999999999,10000000);
select /*+ index(dist, idist) */ * from dist where d_id=20 and d_w_id = 2 * &&1;
delete /*+ index(dist, idist) */ from dist where d_id=20 and d_w_id = 2 * &&1;


REM c_w_id must be double of max(w_id)

insert into cust (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 (96000,20,2 * &&1,'x23456789x123456','OE','x23456789x123456',
'01234567890123456789',
'x23456789x123456789x','x23456789x123456789x','CA','x23456789',
'x23456789x123456',
sysdate,'GC',123456789012,0.1234,123456789012,123456789012,1234,1234,'x');
select /*+ index(cust, icust) */ * from cust where c_id=96000 and c_d_id=20 and c_w_id = 2 *
&&1;
delete /*+ index(cust, icust) */ from cust where c_id=96000 and c_d_id=20 and c_w_id = 2 *
&&1;

REM no_w_id must be double of max(w_id)

insert into nord (no_o_id, no_d_id, no_w_id)
   values (10000000, 20, 2 * &&1);
select * from nord
   where no_o_id=10000000 and no_d_id = 20 and no_w_id = 2 * &&1;
delete from nord
   where no_o_id=10000000 and no_d_id = 20 and no_w_id = 2 * &&1;

REM o_w_id must be double of max(w_id)

insert into ordr (o_id, o_d_id, o_w_id, o_c_id, o_entry_d, o_carrier_id,
                  o_ol_cnt, o_all_local)
   values (10000000, 20, 2 * &&1, 96000, sysdate, 10, 15, 9);
select * from ordr where o_id=10000000 and o_d_id=20 and o_w_id = 2 * &&1;
delete from ordr where o_id=10000000 and o_d_id=20 and o_w_id = 2 * &&1;

REM ol_w_id must be double of max(w_id)

insert into ordl (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                  ol_supply_w_id, ol_delivery_d, ol_quantity,
                  ol_amount, ol_dist_info)
   values (10000000, 20, 2 * &&1, 15, 200000, 2 * &&1, sysdate, 99,
         1234.56, 'TEXT size = 24 Confirmed');
select * from ordl
   where ol_o_id=10000000 and ol_d_id=20 and ol_w_id = 2 * &&1 and ol_number=15;
delete from ordl
   where ol_o_id=10000000 and ol_d_id=20 and ol_w_id = 2 * &&1 and ol_number=15;

REM get timestamp

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

quit;
```

Dbms_sats.analsyse.sql

```
spool analyze.log;
set echo on;

connect system/manager

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                       TABNAME=>'STOK', -
                                       PARTNAME=>NULL, -
                                       ESTIMATE_PERCENT=>1, -
                                       BLOCK_SAMPLE=>TRUE, -
                                       METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                       DEGREE=>10, -
                                       GRANULARITY=>'DEFAULT', -
                                       CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                       TABNAME=>'CUST', -
                                       PARTNAME=>NULL, -
                                       ESTIMATE_PERCENT=>1, -
                                       BLOCK_SAMPLE=>TRUE, -
                                       METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                       DEGREE=>10, -
                                       GRANULARITY=>'DEFAULT', -
                                       CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                       TABNAME=>'ORDR', -
                                       PARTNAME=>NULL, -
                                       ESTIMATE_PERCENT=>1, -
                                       BLOCK_SAMPLE=>TRUE, -
                                       METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                       DEGREE=>10, -
                                       GRANULARITY=>'DEFAULT', -
                                       CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                       TABNAME=>'ORDL', -
                                       PARTNAME=>NULL, -
                                       ESTIMATE_PERCENT=>1, -
                                       BLOCK_SAMPLE=>TRUE, -
                                       METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                       DEGREE=>10, -
                                       GRANULARITY=>'DEFAULT', -
                                       CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                       TABNAME=>'NORD', -
                                       PARTNAME=>NULL, -
                                       ESTIMATE_PERCENT=>1, -
                                       BLOCK_SAMPLE=>TRUE, -
                                       METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                       DEGREE=>10, -
                                       GRANULARITY=>'DEFAULT', -
                                       CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
                                       TABNAME=>'HIST', -
                                       PARTNAME=>NULL, -
                                       ESTIMATE_PERCENT=>1, -
                                       BLOCK_SAMPLE=>TRUE, -
                                       METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                       DEGREE=>10, -
                                       GRANULARITY=>'DEFAULT', -
                                       CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
                                       TABNAME=>'DIST', -
                                       PARTNAME=>NULL, -
                                       ESTIMATE_PERCENT=>1, -
                                       BLOCK_SAMPLE=>TRUE, -
                                       METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                       DEGREE=>10, -
                                       GRANULARITY=>'DEFAULT', -
                                       CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
                                       TABNAME=>'ITEM', -
                                       PARTNAME=>NULL, -
                                       ESTIMATE_PERCENT=>10, -
                                       BLOCK_SAMPLE=>TRUE, -
                                       METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                       DEGREE=>1, -
                                       GRANULARITY=>'DEFAULT', -
                                       CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
                                       TABNAME=>'WARE', -
                                       PARTNAME=>NULL, -
                                       ESTIMATE_PERCENT=>10, -
                                       BLOCK_SAMPLE=>TRUE, -
                                       METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                       DEGREE=>10, -
                                       GRANULARITY=>'DEFAULT', -
                                       CASCADE=>TRUE);


set echo off;
spool off;

exit sql.sqlcode;
```

Dbtables.sh

```
#!/bin/sh
#
# $Header: dbtables.sh 01-jun-98.19:06:47 skareenh Exp $
#
# dbtables.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      dbtables.sh
#
#    DESCRIPTION
#                  Check min, max, count etc. of TPC-C tables.
#
#    NOTES
#                  dbtables.sh [output_file]
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#    rkambo      07/29/99 - Modification
#

if [ $1 ]
then
  OFILE="$1"
else
  OFILE="dbtables.v1"
fi

if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

date | tee ${ODIR}/${OFILE}
#${SDIR}/swt_temp.sh temp

#sqlplus system/manager <<!
#   alter tablespace temp permanent;
#   quit;
#!

sqlplus tpcc/tpcc @${SQLDIR}/dbtables | tee -a ${ODIR}/${OFILE}

#sqlplus system/manager <<!
#   alter tablespace temp temporary;
#   quit;
#!


#${SDIR}/swt_temp.sh system
date | tee -a ${ODIR}/${OFILE}
```

Dbtables.sql

```
Rem
Rem $Header: dbtables.sql 01-jun-98.19:16:53 skareenh Exp $
Rem
Rem dbtables.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem
Rem    DESCRIPTION
rem       Verify TPC-C tables.
Rem
Rem    NOTES
rem               Usage:   sqlplus tpcc/tpcc @table
rem               Note:    May need to modify degree of parallelism for table scans
rem             in order to speed up the run time of this script.
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem

set pagesize 100
set termout on
set echo on
set timing on

column w_tax         format          99999
column w_ytd         format 9999999999.99
column d_tax         format          99999
column d_ytd         format 9999999999.99
column c_credit_lim  format 9999999999.99
column c_discount    format          99999
column c_balance     format 9999999999.99
column c_ytd_payment format 9999999999.99
column h_amount      format        9999.99
column ol_amount     format        9999.99
column i_price/100   format        9999.99
column cnt           format 999999999999


REM get timestamp

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM get all min and max

drop table minmax;
create table minmax (name varchar2(30), minid number, maxid number, cnt number);
create unique index iminmax on minmax (name);
insert into minmax (name, minid, maxid, cnt)
   select 'WAREHOUSE',  min(w_id), max(w_id), count(*) from ware;
insert into minmax (name, minid, maxid, cnt)
   select 'DISTRICT',  min(d_id), max(d_id), count(*) from dist;
insert into minmax (name, minid, maxid, cnt)
   select /*+ parallel(cust,50) */
         'CUSTOMER', min(c_id), max(c_id), count(*) from cust;
insert into minmax (name, minid, maxid, cnt)
   select /*+ parallel(hist,20) */
         'HISTORY', min(h_c_id), max(h_c_id), count(*) from hist;
insert into minmax (name, minid, maxid, cnt)
   select /*+ parallel(ordr,50) */
         'ORDERS', min(o_id), max(o_id), count(*) from ordr;
insert into minmax (name, minid, maxid, cnt)
   select /*+ parallel(nord,50) */
         'NEW_ORDER', min(no_o_id), max(no_o_id), count(*) from nord;
insert into minmax (name, minid, maxid, cnt)
   select /*+ parallel(ordl,50) */
         'ORDER_LINE', min(ol_o_id), max(ol_o_id), count(*) from ordl;
insert into minmax (name, minid, maxid, cnt)
   select 'ITEM', min(i_id), max(i_id), count(*) from item;
insert into minmax (name, minid, maxid, cnt)
   select /*+ parallel(stok,50) */
         'STOCK', 1, 100000, count(*) from stok;

commit;

select * from minmax;


REM warehouse table

select * from ware
   where w_id = (select min(minid) from minmax where name = 'WAREHOUSE');
select w_id, w_ytd from ware
   where w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');
select cnt from minmax where name = 'WAREHOUSE';


REM district table

select * from dist
   where d_id = (select min(minid) from minmax where name = 'DISTRICT')
   and d_w_id = (select min(minid) from minmax where name = 'WAREHOUSE');
select d_w_id, d_id, d_ytd from dist
   where d_id = (select max(maxid) from minmax where name = 'DISTRICT')
   and d_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');
select cnt from minmax where name = 'DISTRICT';
select count(*) from dist
   where d_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');
```

```
REM customer table

select * from cust
   where c_id = (select min(minid) from minmax where name = 'CUSTOMER')
   and c_d_id = (select min(minid) from minmax where name = 'DISTRICT')
   and c_w_id = (select min(minid) from minmax where name = 'WAREHOUSE');
select c_w_id, c_d_id, c_id, c_balance, c_last from cust
   where c_id = (select max(maxid) from minmax where name = 'CUSTOMER')
   and c_d_id = (select max(maxid) from minmax where name = 'DISTRICT')
   and c_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');
select cnt from minmax where name = 'CUSTOMER';
select count(*) from cust
   where c_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');


REM history table

select /*+ full(hist) parallel(hist,20) */ * from hist
   where h_c_id = (select min(minid) from minmax where name = 'HISTORY')
   and h_c_d_id = (select min(minid) from minmax where name = 'DISTRICT')
   and h_c_w_id = (select min(minid) from minmax where name = 'WAREHOUSE');
select /*+ full(hist) parallel(hist,20) */
   h_c_w_id, h_c_d_id, h_c_id, h_amount from hist
   where h_c_id = (select max(maxid) from minmax where name = 'HISTORY')
   and h_c_d_id = (select max(maxid) from minmax where name = 'DISTRICT')
   and h_c_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');
select cnt from minmax where name = 'HISTORY';
select /*+ full(hist) parallel(hist,20) */ count(*) from hist
   where h_c_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');


REM orders table

select * from ordr
   where o_id = (select min(minid) from minmax where name = 'ORDERS')
   and o_d_id = (select min(minid) from minmax where name = 'DISTRICT')
   and o_w_id = (select min(minid) from minmax where name = 'WAREHOUSE');
select o_w_id, o_d_id, o_id, o_c_id from ordr
   where o_id = (select max(maxid) from minmax where name = 'ORDERS')
   and o_d_id = (select max(maxid) from minmax where name = 'DISTRICT')
   and o_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');
select cnt from minmax where name = 'ORDERS';
select count(*) from ordr
   where o_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');


REM new_order table

select * from nord
   where no_o_id = (select min(minid) from minmax where name = 'NEW_ORDER')
   and no_d_id = (select min(minid) from minmax where name = 'DISTRICT')
   and no_w_id = (select min(minid) from minmax where name = 'WAREHOUSE');
select no_w_id, no_d_id, no_o_id from nord
   where no_o_id = (select max(maxid) from minmax where name = 'NEW_ORDER')
   and no_d_id = (select max(maxid) from minmax where name = 'DISTRICT')
   and no_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');
select cnt from minmax where name = 'NEW_ORDER';
select count(*) from nord
   where no_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');


REM order_line table

select * from ordl
   where ol_o_id = (select min(minid) from minmax where name = 'ORDER_LINE')
   and ol_d_id = (select min(minid) from minmax where name = 'DISTRICT')
   and ol_w_id = (select min(minid) from minmax where name = 'WAREHOUSE')
   and ol_number = 1;
select ol_w_id, ol_d_id, ol_o_id, ol_number, ol_i_id from ordl
   where ol_o_id = (select max(maxid) from minmax where name = 'ORDER_LINE')
   and ol_d_id = (select max(maxid) from minmax where name = 'DISTRICT')
   and ol_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE')
   and ol_number = 1;
select cnt from minmax where name = 'ORDER_LINE';
select /*+ parallel(ordl,50) */ count(*) from ordl
   where ol_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');


REM item table

select * from item
   where i_id = (select min(minid) from minmax where name = 'ITEM');
select i_id, i_price/100 from item
   where i_id = (select max(maxid) from minmax where name = 'ITEM');
select cnt from minmax where name = 'ITEM';


REM stock table

select * from stok
   where s_w_id = (select min(minid) from minmax where name = 'WAREHOUSE')
   and s_i_id = (select min(minid) from minmax where name = 'ITEM');
select s_i_id, s_w_id, s_quantity from stok
   where s_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE')
   and s_i_id = (select max(maxid) from minmax where name = 'ITEM');
select cnt from minmax where name = 'STOCK';
select /*+ full(stok) parallel(stok,50) */ count(*) from stok
   where s_w_id = (select max(maxid) from minmax where name = 'WAREHOUSE');

REM get timestamp

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

quit;
```

Dbtest.sql

```
set pagesize 100
set termout on
set echo on
set timing on

column w_tax         format          9.9999
column w_ytd         format 9999999999.99
column d_tax         format          9.9999
column d_ytd         format 9999999999.99
column c_credit_lim  format 9999999999.99
column c_discount    format          9.9999
column c_balance     format 9999999999.99
column c_ytd_payment format 9999999999.99
column h_amount      format       9999.99
column ol_amount     format       9999.99
column i_price       format       9999.99

insert into dist (d_id, d_w_id, d_name, d_street_1, d_street_2, d_city,
                  d_state, d_zip, d_tax, d_ytd, d_next_o_id)
                    values (10,9,'x','x','x','x','x','x',.9999,9,9);

exit
```

Dbview.sh

```
#!/usr/bin/sh

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool ddview.log


REM
REM In an ade/nde view we might need to run standard.sql and dbmsstdx manually
REM catalog and catproc suppose to take care of it
REM

@$ORACLE_HOME/plsql/admin/standard
@$ORACLE_HOME/rdbms/admin/dbmsstdx

@$ORACLE_HOME/rdbms/admin/catalog
@$ORACLE_HOME/rdbms/admin/catproc

REM
REM In an ade/nde view we might need to run pupbld manually
REM catalog and catproc suppose to take care of it
REM

connect system/manager
REM @$ORACLE_HOME/sqlplus/admin/pupbld


REM
REM Oracle
REM

REM if test $NUMBER_ORACLE_NODE -qt 1
REM then

@$ORACLE_HOME/rdbms/admin/catparr

REM fi

spool off
!

sh $tpcc_scripts/queue.sh
```

Defaultopts.sh

```
# configurable vars
tpcc_os=unix
tpcc_version=ttt
tpcc_ldrive=1
tpcc_scale=10
tpcc_np=1
tpcc_cpu=1
#megabytes
tpcc_memsize=512
#minutes
tpcc_runlen=24

tpcc_temp_imp=temp
tpcc_temp_size=10M
tpcc_temp_ext=calc
tpcc_temp_nf=calc
tpcc_temp_bs=2K

tpcc_ware_imp=iot
tpcc_ware_size=10M
tpcc_ware_ext=calc
tpcc_ware_nf=calc
tpcc_ware_bs=auto

tpcc_ware_used=-1
tpcc_ware_free=-1
tpcc_ware_trans=-1

tpcc_ware_indices=1-

tpcc_ware_autospace=t
tpcc_ware_flg=30
tpcc_ware_fl=22

tpcc_iware_imp=none
tpcc_iware_size=1M
tpcc_iware_ext=calc
tpcc_iware_nf=calc
tpcc_iware_bs=2K

tpcc_iware_used=-1
tpcc_iware_free=-1
tpcc_iware_trans=-1

tpcc_iware_autospace=t
tpcc_iware_flg=30
tpcc_iware_fl=22

tpcc_iware_indices=1-



tpcc_dist_imp=cluster
tpcc_dist_size=10M
tpcc_dist_ext=calc
tpcc_dist_nf=calc
tpcc_dist_bs=auto

tpcc_dist_used=-1
tpcc_dist_free=-1
tpcc_dist_trans=-1

tpcc_dist_indices=2-1-

tpcc_dist_autospace=t
tpcc_dist_flg=30
tpcc_dist_fl=22

tpcc_idist_imp=index
tpcc_idist_size=1M
tpcc_idist_ext=calc
tpcc_idist_nf=calc
tpcc_idist_bs=2K

tpcc_idist_used=-1
tpcc_idist_free=-1
tpcc_idist_trans=-1

tpcc_idist_autospace=t
tpcc_idist_flg=30
tpcc_idist_fl=22

tpcc_idist_indices=2-1-


tpcc_item_imp=cluster
tpcc_item_size=15M
tpcc_item_ext=calc
tpcc_item_nf=calc
tpcc_item_bs=auto

tpcc_item_used=-1
tpcc_item_free=-1
tpcc_item_trans=-1

tpcc_item_indices=1-

tpcc_item_autospace=t
tpcc_item_flg=30
tpcc_item_fl=22

tpcc_iitem_imp=index
tpcc_iitem_size=1M
tpcc_iitem_ext=calc
tpcc_iitem_nf=calc
tpcc_iitem_bs=2K

tpcc_iitem_used=-1
tpcc_iitem_free=-1
tpcc_iitem_trans=-1

tpcc_iitem_autospace=t
tpcc_iitem_flg=30
tpcc_iitem_fl=22

tpcc_iitem_indices=1-


tpcc_nord_imp=iot
tpcc_nord_size=10M
tpcc_nord_ext=calc
tpcc_nord_nf=calc
tpcc_nord_bs=auto

tpcc_nord_used=-1
tpcc_nord_free=-1
tpcc_nord_trans=-1

tpcc_nord_indices=1-2-3-

tpcc_nord_autospace=t
tpcc_nord_flg=30
tpcc_nord_fl=22

tpcc_inord_imp=none
tpcc_inord_size=1M
tpcc_inord_ext=calc
tpcc_inord_nf=calc
tpcc_inord_bs=2K

tpcc_inord_used=-1
tpcc_inord_free=-1
tpcc_inord_trans=-1

tpcc_inord_autospace=t
tpcc_inord_flg=30
tpcc_inord_fl=22

tpcc_inord_indices=1-2-3-


tpcc_ordl_imp=iot
tpcc_ordl_size=10M
tpcc_ordl_ext=calc
tpcc_ordl_nf=calc
tpcc_ordl_bs=auto

tpcc_ordl_used=-1
tpcc_ordl_free=-1
tpcc_ordl_trans=-1

tpcc_ordl_indices=1-2-3-4-

tpcc_ordl_autospace=t
tpcc_ordl_flg=30
tpcc_ordl_fl=22

tpcc_iordl_imp=none
tpcc_iordl_size=1M
tpcc_iordl_ext=calc
tpcc_iordl_nf=calc
tpcc_iordl_bs=2K

tpcc_iordl_used=-1
tpcc_iordl_free=-1
tpcc_iordl_trans=-1

tpcc_iordl_autospace=t
tpcc_iordl_flg=30
tpcc_iordl_fl=22

tpcc_iordl_indices=1-2-3-4-


tpcc_ordr_imp=table
tpcc_ordr_size=10M
tpcc_ordr_ext=calc
```

```
tpcc_ordr_nf=calc
tpcc_ordr_bs=auto

tpcc_ordr_used=-1
tpcc_ordr_free=-1
tpcc_ordr_trans=-1

tpcc_ordr_indices=2-3-1

tpcc_ordr_autospace=t
tpcc_ordr_flg=30
tpcc_ordr_fl=22

tpcc_iordr1_imp=index
tpcc_iordr1_size=1M
tpcc_iordr1_ext=calc
tpcc_iordr1_nf=calc
tpcc_iordr1_bs=2K

tpcc_iordr1_used=-1
tpcc_iordr1_free=-1
tpcc_iordr1_trans=-1

tpcc_iordr1_autospace=t
tpcc_iordr1_flg=30
tpcc_iordr1_fl=22

tpcc_iordr1_indices=2-3-1

tpcc_iordr2_imp=index
tpcc_iordr2_size=1M
tpcc_iordr2_ext=calc
tpcc_iordr2_nf=calc
tpcc_iordr2_bs=2K

tpcc_iordr2_used=-1
tpcc_iordr2_free=-1
tpcc_iordr2_trans=-1

tpcc_iordr2_autospace=t
tpcc_iordr2_flg=30
tpcc_iordr2_fl=22

tpcc_iordr2_indices=2-3-4-1



tpcc_stok_imp=cluster
tpcc_stok_size=35M
tpcc_stok_ext=calc
tpcc_stok_nf=calc
tpcc_stok_bs=auto

tpcc_stok_used=-1
tpcc_stok_free=-1
tpcc_stok_trans=-1

tpcc_stok_indices=1-2-

tpcc_stok_autospace=t
tpcc_stok_flg=30
tpcc_stok_fl=22

tpcc_istok_imp=index
tpcc_istok_size=1M
tpcc_istok_ext=calc
tpcc_istok_nf=calc
tpcc_istok_bs=2K

tpcc_istok_used=-1
tpcc_istok_free=-1
tpcc_istok_trans=-1

tpcc_istok_autospace=t
tpcc_istok_flg=30
tpcc_istok_fl=22

tpcc_istok_indices=1-2-


tpcc_cust_imp=cluster
tpcc_cust_size=25M
tpcc_cust_ext=calc
tpcc_cust_nf=calc
#bs
tpcc_cust_bs=2K

tpcc_cust_used=-1
tpcc_cust_free=-1
tpcc_cust_trans=-1

tpcc_cust_indices=1-2-3-

tpcc_cust_autospace=t
tpcc_cust_flg=30
tpcc_cust_fl=22


tpcc_icust1_imp=index
tpcc_icust1_size=1M
tpcc_icust1_ext=calc
tpcc_icust1_nf=calc
tpcc_icust1_bs=2K

tpcc_icust1_used=-1
tpcc_icust1_free=-1
tpcc_icust1_trans=-1

tpcc_icust1_autospace=t
tpcc_icust1_flg=30
tpcc_icust1_fl=22

tpcc_icust1_indices=1-2-3-

tpcc_icust2_imp=index
tpcc_icust2_size=1M
tpcc_icust2_ext=calc
tpcc_icust2_nf=calc
tpcc_icust2_bs=2K

tpcc_icust2_used=-1
tpcc_icust2_free=-1
tpcc_icust2_trans=-1

tpcc_icust2_autospace=t
tpcc_icust2_flg=30
tpcc_icust2_fl=22

tpcc_icust2_indices=6-7-1-2-3-

tpcc_hist_imp=table
tpcc_hist_size=10M
tpcc_hist_ext=calc
tpcc_hist_nf=calc
tpcc_hist_bs=auto

tpcc_hist_used=-1
tpcc_hist_free=-1
tpcc_hist_trans=-1

tpcc_hist_indices=no

tpcc_hist_autospace=t
tpcc_hist_flg=30
tpcc_hist_fl=22
```

Del_iso5.sql

```
Rem
Rem $Header: del_iso5.sql 01-jun-98.19:17:10 skareenh Exp $
Rem
Rem del_iso5.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      del_iso5.sql - <one-line expansion of the name>
Rem
Rem    DESCRIPTION
rem       SQL script to create a stored procedure for delivery
rem       transactions for isolation tests 5 and 9.
Rem
Rem    NOTES
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem


CREATE OR REPLACE PACKAGE adelivery
IS
  TYPE intarray IS TABLE OF INTEGER INDEX BY BINARY_INTEGER;
  PROCEDURE adeliver
  (
    ware_id          INTEGER,
    carrier_id       INTEGER,
    order_id   IN OUT intarray,
    retry      IN OUT INTEGER
  );
END;
/


CREATE OR REPLACE PACKAGE BODY adelivery
IS
  PROCEDURE adeliver
  (
    ware_id          INTEGER,
```

```
  carrier_id        INTEGER,
  order_id   IN OUT intarray,
  retry      IN OUT INTEGER
)
IS
  dist_id       INTEGER;
  cust_id       INTEGER;
  amount_sum    NUMBER;
  no_rowid      UROWID;
  node_num      VARCHAR2(10);
  not_serializable EXCEPTION;
  PRAGMA EXCEPTION_INIT(not_serializable, -8177);
  deadlock         EXCEPTION;
  PRAGMA EXCEPTION_INIT(deadlock, -60);
  snapshot_too_old EXCEPTION;
  PRAGMA EXCEPTION_INIT(snapshot_too_old, -1555);
  CURSOR n_cur IS
    SELECT no_o_id, rowid
    FROM nord
    WHERE no_w_id = ware_id AND no_d_id = dist_id
    ORDER BY no_w_id, no_d_id, no_o_id;
BEGIN

  SELECT substr(value,1,5)
    INTO node_num
    FROM v$parameter
    WHERE name = 'instance_number';

  plsql_mon_pack.print ('Delivery started at ' ||
                  to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                  node_num);

  FOR i IN 1 .. 10 LOOP
    dist_id := i;

    LOOP BEGIN
      OPEN n_cur;
      FETCH n_cur INTO order_id(i), no_rowid;

      IF (n_cur%NOTFOUND) THEN             -- no new order
        CLOSE n_cur;
        plsql_mon_pack.print ('Delivery [dist ' || to_char (dist_id) ||
                        '] found no new order at ' ||
                        to_char (sysdate, 'HH24:MI:SS'));
        dbms_lock.sleep (60);
        plsql_mon_pack.print ('Delivery [dist ' || to_char (dist_id) ||
                        '] wake up at ' ||
                        to_char (sysdate, 'HH24:MI:SS'));
        OPEN n_cur;
        FETCH n_cur INTO order_id(i), no_rowid;
        IF (n_cur%NOTFOUND) THEN
          plsql_mon_pack.print ('Delivery [dist ' || to_char(dist_id) ||
                        '] found no new order at ' ||
                        to_char (sysdate, 'HH24:MI:SS'));
        ELSE
          plsql_mon_pack.print ('Delivery [dist ' || to_char(dist_id) ||
                        '] found new order ' ||
                        to_char (order_id(i)) || ' at ' ||
                      to_char (sysdate, 'HH24:MI:SS'));
        END IF;
        CLOSE n_cur;
        COMMIT;
        order_id(i) := 0;
        EXIT;
      END IF;

      CLOSE n_cur;

      DELETE FROM nord
        WHERE rowid = no_rowid;

      UPDATE ordr
        SET o_carrier_id = carrier_id
        WHERE o_d_id = dist_id AND o_w_id = ware_id AND
            o_id = order_id(i);

      SELECT o_c_id
        INTO cust_id
        FROM ordr
        WHERE o_d_id = dist_id AND o_w_id = ware_id AND
            o_id = order_id(i);

      UPDATE ordl
        SET ol_delivery_d = SYSDATE
        WHERE ol_d_id = dist_id AND ol_w_id = ware_id AND
            ol_o_id = order_id(i);

      SELECT sum(ol_amount)
        INTO amount_sum
        FROM ordl
        WHERE ol_d_id = dist_id AND ol_w_id = ware_id AND
            ol_o_id = order_id(i);

      UPDATE cust
        SET c_balance = c_balance + amount_sum,
            c_delivery_cnt = c_delivery_cnt + 1
        WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id = ware_id;

      IF (dist_id = 5) THEN
        plsql_mon_pack.print ('Delivery [dist 5] sleep before commit at '
                        || to_char (sysdate, 'HH24:MI:SS'));
        plsql_mon_pack.print ('   cust_id = ' || to_char (cust_id) ||
                        '   amount_sum = ' ||
                        to_char (amount_sum, '9999999999.99'));
        dbms_lock.sleep (60);
        plsql_mon_pack.print ('Delivery [dist 5] wake up at ' ||
                        to_char (sysdate, 'HH24:MI:SS'));
      END IF;
      COMMIT;
      EXIT;

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

    END LOOP;
  END LOOP;
END;
/

quit;
```

Del_iso6.sql

```
Rem
Rem $Header: del_iso6.sql 01-jun-98.19:17:29 skareenh Exp $
Rem
Rem del_iso6.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      del_iso6.sql
Rem
Rem    DESCRIPTION
rem       SQL script to create a stored procedure for delivery
rem       transactions for isolation test 6.
Rem
Rem    NOTES
Rem      <other useful comments, qualifications, etc.>
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem


CREATE OR REPLACE PACKAGE adelivery
IS
  TYPE intarray IS TABLE OF INTEGER INDEX BY BINARY_INTEGER;
  PROCEDURE adeliver
  (
    ware_id          INTEGER,
    carrier_id       INTEGER,
    order_id   IN OUT intarray,
    retry      IN OUT INTEGER
  );
END;
/


CREATE OR REPLACE PACKAGE BODY adelivery
IS
  PROCEDURE adeliver
  (
    ware_id          INTEGER,
    carrier_id       INTEGER,
    order_id   IN OUT intarray,
    retry      IN OUT INTEGER
  )
  IS
    dist_id       INTEGER;
    cust_id       INTEGER;
    amount_sum    NUMBER;
    no_rowid      ROWID;
    node_num      VARCHAR2(10);
    not_serializable EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable, -8177);
    deadlock         EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock, -60);
    snapshot_too_old EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old, -1555);
    CURSOR n_cur IS
      SELECT no_o_id, rowid
      FROM nord
```

```
    WHERE no_w_id = ware_id AND no_d_id = dist_id
    ORDER BY no_w_id, no_d_id, no_o_id;
BEGIN

  SELECT substr(value,1,5)
    INTO node_num
    FROM v$parameter
    WHERE name = 'instance_number';

  plsql_mon_pack.print ('Delivery started at ' ||
                  to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                  node_num);

  FOR i IN 1 .. 10 LOOP
    dist_id := i;

    LOOP BEGIN
      OPEN n_cur;
      FETCH n_cur INTO order_id(i), no_rowid;

      IF (n_cur%NOTFOUND) THEN              -- no new order
        CLOSE n_cur;
        plsql_mon_pack.print ('Delivery [dist ' || to_char (dist_id) ||
                        '] found no new order at ' ||
                        to_char (sysdate, 'HH24:MI:SS'));
        dbms_lock.sleep (60);
        plsql_mon_pack.print ('Delivery [dist ' || to_char (dist_id) ||
                        '] wake up at ' ||
                        to_char (sysdate, 'HH24:MI:SS'));
        OPEN n_cur;
        FETCH n_cur INTO order_id(i), no_rowid;
        IF (n_cur%NOTFOUND) THEN
          plsql_mon_pack.print ('Delivery [dist ' || to_char(dist_id) ||
                        '] found no new order at ' ||
                        to_char (sysdate, 'HH24:MI:SS'));
        ELSE
          plsql_mon_pack.print ('Delivery [dist ' || to_char(dist_id) ||
                        '] found new order ' ||
                        to_char (order_id(i)) || ' at ' ||
                      to_char (sysdate, 'HH24:MI:SS'));
        END IF;
        CLOSE n_cur;
        COMMIT;
        order_id(i) := 0;
        EXIT;
      END IF;

      CLOSE n_cur;

      DELETE FROM nord
        WHERE rowid = no_rowid;

      UPDATE ordr
        SET o_carrier_id = carrier_id
        WHERE o_d_id = dist_id AND o_w_id = ware_id AND
            o_id = order_id(i);

      SELECT o_c_id
        INTO cust_id
        FROM ordr
        WHERE o_d_id = dist_id AND o_w_id = ware_id AND
            o_id = order_id(i);

      UPDATE ordl
        SET ol_delivery_d = SYSDATE
        WHERE ol_d_id = dist_id AND ol_w_id = ware_id AND
            ol_o_id = order_id(i);

      SELECT sum(ol_amount)
        INTO amount_sum
        FROM ordl
        WHERE ol_d_id = dist_id AND ol_w_id = ware_id AND
            ol_o_id = order_id(i);

      UPDATE cust
        SET c_balance = c_balance + amount_sum,
          c_delivery_cnt = c_delivery_cnt + 1
        WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id = ware_id;

      IF (dist_id = 5) THEN
        plsql_mon_pack.print ('Delivery [dist 5] sleep before abort at '
                        || to_char (sysdate, 'HH24:MI:SS'));
        plsql_mon_pack.print ('   cust_id = ' || to_char (cust_id) ||
                        '   amount_sum = ' ||
                        to_char (amount_sum, '9999999999.99'));
        dbms_lock.sleep (60);
        plsql_mon_pack.print ('Delivery [dist 5] wake up at ' ||
                        to_char (sysdate, 'HH24:MI:SS'));
      END IF;
      ROLLBACK;
      EXIT;

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

        END LOOP;
      END LOOP;
    END;
  END;
  /

  quit;
```

Dml.sql

```
REM============================================================
+
REM       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
REM                  OPEN SYSTEMS PERFORMANCE GROUP                 |
REM                       All Rights Reserved                       |
REM============================================================
+
REM FILENAME
REM     dml.sql
REM DESCRIPTION
REM     Disable table locks for TPC-C tables.
REM USAGE
REM     sqlplus tpcc/tpcc dml.sql
REM============================================================
=

connect tpcc/tpcc;
set echo on;

   alter table ware disable table lock;
   alter table dist disable table lock;
   alter table cust disable table lock;
   alter table hist disable table lock;
   alter table item disable table lock;
   alter table stok disable table lock;
   alter table ordr disable table lock;
   alter table nord disable table lock;
   alter table ordl disable table lock;

set echo off;

connect $oracle_dba/$oracle_dba_password;
```

Dnoid.sql

```
set pagesize 1000
set termout on
set echo on
set timing on

select d_w_id, avg(d_next_o_id) avg_next_oid
   from dist
   where d_w_id > 3300
   group by d_w_id
   order by d_w_id;
```

Double.sql

```
Rem
Rem $Header: dbcheck.sql 01-jun-98.19:16:35 skareenh Exp $
Rem
Rem dbcheck.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      dbcheck.sql
Rem
Rem    DESCRIPTION
rem       Check TPC-C database.
Rem
Rem    NOTES
rem               Usage:   sqlplus tpcc/tpcc @dbcheck <max_warehouse_id>
rem               Note:    May need to modify degree of parallelism for table scans
rem         in order to speed up the run time of this script.
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem
```

```
set pagesize 100
set termout on
set echo on
set timing on

column w_tax         format          9.9999
column w_ytd         format 9999999999.99
column d_tax         format          9.9999
column d_ytd         format 9999999999.99
column c_credit_lim  format 9999999999.99
column c_discount    format          9.9999
column c_balance     format 9999999999.99
column c_ytd_payment format 9999999999.99
column h_amount      format       9999.99
column ol_amount     format       9999.99
column i_price       format       9999.99


REM get timestamp

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM w_id must be double of max(w_id)

insert into ware (w_id, w_name, w_street_1, w_street_2,
                  w_city, w_state, w_zip, w_tax, w_ytd )
  values (2 * &&1,'x','x','x','x','x','x',.9999,999999999999);
select /*+ index(ware, iware) */ * from ware where w_id = 2 * &&1;
delete /*+ index(ware, iware) */ from ware where w_id = 2 * &&1;

REM s_w_id must be double of max(w_id)

insert into stok (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 (200000,2 *
&&1,9999,'xxxxxxxxxxxxxxxxxxxxxxxxxx','x','x','x','x','x','x','x','x',99999999,9999,9999,'012345
678901234567890123456789012345678901234567890123456789');
select /*+ index(stok, istok) */ * from stok where s_i_id=200000 and s_w_id = 2 * &&1;
delete /*+ index(stok, istok) */ from stok where s_i_id=200000 and s_w_id = 2 * &&1;


REM d_w_id must be double of max(w_id)

insert into dist (d_id, d_w_id, d_name, d_street_1, d_street_2, d_city,
                  d_state, d_zip, d_tax, d_ytd, d_next_o_id)
  values (20,2 * &&1,'x','xxxxxxxxxxxxxxxxxxxx','x','x','x','x',.9999,9999999999999,99999999);
select /*+ index(dist, idist) */ * from dist where d_id=20 and d_w_id = 2 * &&1;
delete /*+ index(dist, idist) */ from dist where d_id=20 and d_w_id = 2 * &&1;


REM c_w_id must be double of max(w_id)

insert into cust (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 (96000,20,2 * &&1,'x','x','x','0123456789012345678 9','x','x','x','x','x',
        sysdate,'x',999999999999,0.9999,999999999999,9999,9999,9,'x');
select /*+ index(cust, icust) */ * from cust where c_id=96000 and c_d_id=20 and c_w_id = 2 *
&&1;
delete /*+ index(cust, icust) */ from cust where c_id=96000 and c_d_id=20 and c_w_id = 2 *
&&1;

REM no_w_id must be double of max(w_id)

insert into nord (no_o_id, no_d_id, no_w_id)
  values (10000000, 20, 2 * &&1);
select * from nord
  where no_o_id=10000000 and no_d_id = 20 and no_w_id = 2 * &&1;
delete from nord
  where no_o_id=10000000 and no_d_id = 20 and no_w_id = 2 * &&1;

REM o_w_id must be double of max(w_id)

insert into ordr (o_id, o_d_id, o_w_id, o_c_id, o_entry_d, o_carrier_id,
                  o_ol_cnt, o_all_local)
  values (10000000, 20, 2 * &&1, 96000, sysdate, 10, 15, 9);
select * from ordr where o_id=10000000 and o_d_id=20 and o_w_id = 2 * &&1;
delete from ordr where o_id=10000000 and o_d_id=20 and o_w_id = 2 * &&1;

REM ol_w_id must be double of max(w_id)

insert into ordl (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                  ol_supply_w_id, ol_delivery_d, ol_quantity,
                  ol_amount, ol_dist_info)
  values (10000000, 20, 2 * &&1, 15, 200000, 2 * &&1, sysdate, 99,
        1234.56, 'TEXT size = 24 Confirmed');
select * from ordl
  where ol_o_id=10000000 and ol_d_id=20 and ol_w_id = 2 * &&1 and ol_number=15;
delete from ordl
  where ol_o_id=10000000 and ol_d_id=20 and ol_w_id = 2 * &&1 and ol_number=15;

REM get timestamp

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

quit;
```

Driver.sh

```
#!/bin/sh

. ./stepenv.sh

if expr $# \< 1 > /dev/null; then
  echo "$0 <starting stepname> <optional: only>"
  echo OR use:
  echo "$0 buildcreate  - to build the database creation scripts"
  echo "$0 create       - to create the database (after buildcreate)"
  echo "$0 steps        - to list individual steps"
  exit 1
fi

if expr x$1 = xsteps > /dev/null; then
  echo stepnames are from creation scripts: $tpcc_create_steps
  echo or running steps: $tpcc_steps
  echo "use the 'only' option to only do that step (otherwise all steps after will also be executed.)"
  exit 1
fi

startstep=$1
onlystep=$2

# Aliases for special steps
if expr $startstep = buildcreate > /dev/null; then
  startstep=`echo $tpcc_create_steps | cut -d' ' -f1`
fi

if expr $startstep = create > /dev/null; then
  startstep=`echo $tpcc_steps | cut -d' ' -f1`
fi

echo Starting from step: $startstep

dostep=f
for step in $tpcc_create_steps $tpcc_steps ; do
  if expr $step = $startstep > /dev/null; then
    dostep=t
  fi

  if expr $dostep = t > /dev/null; then
    echo $step
    cd $tpcc_bench
    $tpcc_scripts/`echo $step | cut -d- -f1`.sh `echo $step | sed -e's/-*$/-/' | cut -d- -f2- | sed -e's/-/
/g'`
    lasterror=$?
    cd $tpcc_bench
#    if test `find $tpcc_bench -follow -name '*.log'|wc -l` -ne 0; then
#                  mv `find $tpcc_bench -follow -name '*.log'` $tpcc_bench/log/
#    fi
    if test `ls $tpcc_bench/*.log |wc -l` -ne 0; then
                  mv $tpcc_bench/*.log $tpcc_bench/log/
    fi

    if expr $lasterror != 0 > /dev/null; then
      if expr $lasterror != 99 > /dev/null; then
        echo Step $step failed.  Stopping driver.
        exit 1
      else
        echo Step $step completed and requested stop.  Stopping driver.
        exit 0
      fi
    fi
    if expr x$onlystep = xonly > /dev/null; then
      exit 0
    fi
  fi
done

if expr $dostep = f > /dev/null; then
  echo No such step: $1
fi
```

Dura.sh

```
#!/bin/sh
#
# $Header: dura.sh 01-jun-98.19:07:04 skareenh Exp $
#
# dura.sh
```

```
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#   NAME
#     dura.sh
#
#   DESCRIPTION
#                 Check for the existence of a row in the orders table with
#                 the specified o_w_id, o_d_id, o_c_id, o_id.  This is used
#                 in durability tests to spot-check the database after a crash.
#
#   NOTES
#                 dura.sh <o_w_id> <o_d_id> <o_c_id> <o_id>
#
#   MODIFIED   (MM/DD/YY)
#   skareenh    06/01/98 - Creation
#


if [ "$#" -ne "4" ]
then
    echo "dura.sh: Wrong number of parameters!"
    echo "Usage: dura.sh <o_w_id> <o_d_id> <o_c_id> <o_id>"
    exit 0;
fi

sqlplus tpcc/tpcc <<!
   set termout on
   set echo on
   select o_id, o_w_id, o_d_id, o_c_id, o_ol_cnt
   from   ordr
   where  o_w_id = $1 and o_d_id = $2 and o_c_id = $3 and o_id = $4;
   exit;
!
```

Ec.sql

```
set pagesize 100
set termout on
set echo on
set timing on

column w_tax         format          9.9999
column w_ytd         format 9999999999.99
column d_tax         format          9.9999
column d_ytd         format 9999999999.99
column c_credit_lim  format 9999999999.99
column c_discount    format          9.9999
column c_balance     format 9999999999.99
column c_ytd_payment format 9999999999.99
column h_amount      format       9999.99
column ol_amount     format       9999.99
column i_price       format       9999.99


REM get timestamp

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;


REM INSERT tests with new values

insert into item (i_id, i_im_id, i_name, i_price, i_data)
   values (200000, 200000, 'x', 99999, 'x');
select * from item where i_id > 100000;
delete from item where i_id >  100000;


REM w_id must be double of max(w_id)

insert into ware (w_id, w_name, w_street_1, w_street_2,
                  w_city, w_state, w_zip, w_tax, w_ytd )
   values (2 * &&1,'x','x','x','x','x','x',.9999,999999999999);
select * from ware where w_id >  &&1;
delete from ware where w_id >  &&1;


REM s_w_id must be double of max(w_id)

insert into stok (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 (200000,2 *
&&1,9999,'xxxxxxxxxxxxxxxxxxxxxxxxxx','x','x','x','x','x','x','x','x',99999999,9999,9999,'012345
6789012345678901234567890123456789012345678901234567890123456789');
select * from stok where s_i_id > 100000 or s_w_id > &&1;
delete from stok where s_i_id > 100000 or s_w_id > &&1;


REM d_w_id must be double of max(w_id)

insert into dist (d_id, d_w_id, d_name, d_street_1, d_street_2, d_city,
                  d_state, d_zip, d_tax, d_ytd, d_next_o_id)
   values (20,2 * &&1,'x','xxxxxxxxxxxxxxxxxxxx','x','x','x','x',.9999,999999999999,99999999);
select * from dist where d_id>10 or d_w_id > &&1;
delete from dist where d_id>10 or d_w_id > &&1;


REM c_w_id must be double of max(w_id)

insert into cust (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 (96000,20,2 * &&1,'x','x','x','0123456789012345678 9','x','x','x','x','x',
          sysdate,'x',999999999999,9999,999999999999,9999,9999,9,'x');
select * from cust where c_id>3000 or c_d_id>10 or c_w_id > &&1;
delete from cust where c_id>3000 or c_d_id>10 or c_w_id > &&1;

REM no_w_id must be double of max(w_id)

insert into nord (no_o_id, no_d_id, no_w_id)
   values (10000000, 20, 2 * &&1);
select * from nord
   where no_o_id=10000000 and no_d_id = 20 and no_w_id = 2 * &&1;
delete from nord
   where no_o_id=10000000 and no_d_id = 20 and no_w_id = 2 * &&1;

REM o_w_id must be double of max(w_id)

insert into ordr (o_id, o_d_id, o_w_id, o_c_id, o_entry_d, o_carrier_id,
                  o_ol_cnt, o_all_local)
   values (10000000, 20, 2 * &&1, 96000, sysdate, 10, 15, 9);
select * from ordr where o_id=10000000 and o_d_id=20 and o_w_id = 2 * &&1;
delete from ordr where o_id=10000000 and o_d_id=20 and o_w_id = 2 * &&1;

REM ol_w_id must be double of max(w_id)

insert into ordl (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                    ol_supply_w_id, ol_delivery_d, ol_quantity,
                    ol_amount, ol_dist_info)
   values (10000000, 20, 2 * &&1, 15, 200000, 2 * &&1, sysdate, 99,
          1234.56, 'TEXT size = 24 Confirmed');
select * from ordl
   where ol_o_id=10000000 and ol_d_id=20 and ol_w_id = 2 * &&1 and ol_number=15;
delete from ordl
   where ol_o_id=10000000 and ol_d_id=20 and ol_w_id = 2 * &&1 and ol_number=15;

REM get timestamp

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

quit;
```

Evenload.sh

```
#!/bin/sh
#evenly load using tpcc load, following parameters:
#$1 name of the table to load- this is used to choose where to log.
#$2 the number of things to load
#$3 the starting flag (usually b or j)
#$4 the ending flag (usually e or k)
#$5 the flag to load (h for history , c for cust, S for stock, etc.
#$6 if true, add dummy (only used for -o so far.)
#$7 the command to be used, if not $tpcc_load

command=$7
if test -z "$command"; then
command='$tpcc_load'
fi

tablename=$1
# write out to file to load later
loadout=$tpcc_genscripts_dir/load${tablename}.sh
rm ${loadout}
echo \#created automatically by $0 `date` > $loadout
echo "rm load${tablename}*.log" >> $loadout
echo 'cd $tpcc_bench' >> $loadout

numloaders=`$tpcc_bcexpr 2 \* $tpcc_cpu`
if expr $numloaders \> $2 > /dev/null; then
  numloaders=$2
fi

echo "allprocs=" >> $loadout
curstuff=1
stuffextra=`expr $2 \% $numloaders`
stuffinc=`expr $2 / $numloaders`
curloader=0
while expr $curloader \< $numloaders > /dev/null; do
  newstuff=`expr $curstuff + $stuffinc + \( $stuffextra / $numloaders \)`
  if expr x$6 = xt > /dev/null; then
```

```
    if expr $tpcc_os = unix > /dev/null; then
      adddummy=\$tpcc_disks_location/dummy${curloader}.dat
    else
      # is this what we actually want to do?  check nt stuff
      adddummy=\\\\\\\\\\\\\\\\.\\\\\\\dummy${curloader}.dat
    fi
  else
    adddummy=
  fi
  echo "$command -M $tpcc_scale -$5 $adddummy -$3 $curstuff -$4 `expr $newstuff - 1` >>
load${tablename}${curloader}.log 2>&1 &" >> $loadout
  echo 'allprocs="$allprocs ${!}"' >> $loadout
  curstuff=$newstuff

  stuffextra=`expr $stuffextra + 1`
  curloader=`expr 1 + $curloader`
done

cat >> $loadout <<!
error=0
for curproc in \$allprocs; do
  wait \$curproc
  error=\`expr \$? + \$error\`
done
exit \`expr \$error != 0\`
!

exit 0
```

Extractcols.sh

```
#!/bin/sh

extractcols(){
  table=$1
  tablecols=`tp $table cols`

  lines=`echo "$tablecols" | sed -e's/^ *//' | cut -d'-' -f1 |\
    sed -e's/\(.*\)/"\1", /' | tr -s '\n' ' '`
  echo "tablecols['${table}'] = [$lines" | sed -e's/, $/];/'
}

defaultcols(){
  table=$1
  tableinds=`tp $table indices`

  indarr=`echo "$tableinds" | sed -e's/\([0-9][0-9]*\)/\1,/g'`
  echo "tableinds['${table}'] = [$indarr];" | sed -e's/-//g' | sed -e's/,]/]/' | sed -e's/\[no\]/[]/g'
}

. ../stepenv.sh
. $tpcc_scripts/tabledata.sh
for table in $tpcc_table_list; do
  extractcols $table
done

for table in $tpcc_table_list $tpcc_index_list; do
  defaultcols $table
done
```

Freeext.sql

```
REM============================================================================
+
REM           Copyright (c) 1994  Oracle Corp, Belmont, CA         |
REM                OPEN SYSTEMS PERFORMANCE GROUP                  |
REM                    All Rights Reserved                         |
REM============================================================================
+
REM FILENAME
REM       freeext.sql
REM DESCRIPTION
REM       List all free extents in all the TPCC tablespace
REM
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @freeext
REM============================================================================
*/
      set space 2
      set pagesize 2000
      set echo off
      set termout off
      set verify off
      set feedback off
      spool freeextent.rpt
      select substr(e.tablespace_name,1,8) tspace, file_id, block_id, blocks,
            blocks * t.block_size / 1048576 size_MB
      from dba_free_space e, dba_tablespaces t
      where e.tablespace_name = t.tablespace_name
      order by e.tablespace_name, file_id, block_id;
      select substr(e.tablespace_name,1,8) tspace, sum(blocks) tot_blk,
            sum(blocks) * t.block_size / 1048576 size_MB
      from dba_free_space e, dba_tablespaces t
      where e.tablespace_name = t.tablespace_name
      group by e.tablespace_name, t.block_size
      order by e.tablespace_name;
```

Fromkilobytes.sh

```
#!/bin/sh
# convert into closest k, m, g, t from number of kilobytes.

amount=$1
if $tpcc_isneg `$tpcc_bcexpr $amount - 1024`; then
  echo ${amount}K
  exit 0;
fi;
amount=`$tpcc_bcexpr $amount / 1024`
if $tpcc_isneg `$tpcc_bcexpr $amount - 1024`; then
  echo ${amount}M
  exit 0;
fi;
amount=`$tpcc_bcexpr $amount / 1024`
if $tpcc_isneg `$tpcc_bcexpr $amount - 1024`; then
  echo ${amount}G
  exit 0;
fi;
amount=`$tpcc_bcexpr $amount / 1024`
echo ${amount}T
```

Get_cust.sql

```
Rem
Rem $Header: get_cust.sql 01-jun-98.19:17:47 skareenh Exp $
Rem
Rem get_cust.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      get_cust.sql
Rem
Rem    DESCRIPTION
rem       Get customer id that will be hit by the current delivery
rem       transaction.  This is used for isolation test 5 and 6.
Rem
Rem    NOTES
rem                 Usage:   sqlplus tpcc/tpcc @get_cust <ware_id> <dist_id>
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem

select 'THE_VALUE ' || to_char (o_c_id) o_c_id
   from ordr
   where o_d_id = &&2 and o_w_id = &&1 and o_id =
     (select min(no_o_id)
        from nord
        where no_w_id = &&1 and no_d_id = &&2);
commit;
exit
```

Grant.sh

```
#!/bin/sh
#
# $Header: grant.sh 01-jun-98.19:07:21 skareenh Exp $
#
# grant.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      grant.sh
#
#    DESCRIPTION
#                  Set up for ACID tests.
#
#    NOTES
#                  grant.sh
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
```

```
#

if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
    SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit
SQLDIR=${TPCC_AUDIT}/sql

sqlplus 'sys/change_on_install as sysdba' <<!
   grant execute on dbms_lock to public;
   grant execute on dbms_pipe to public;
   grant select on v_\$parameter to public;
   quit;
!

sqlplus tpcc/tpcc @${SQLDIR}/plsql_mon
sqlplus tpcc/tpcc @${SQLDIR}/cre_tab
```

Hardanalyse.sql

```
spool analyze.log;
set echo on;

connect tpcc/tpcc;

ANALYZE TABLE stok COMPUTE STATISTICS;
ANALYZE TABLE cust COMPUTE STATISTICS;
ANALYZE TABLE ordr COMPUTE STATISTICS;
ANALYZE TABLE ordl COMPUTE STATISTICS;
ANALYZE TABLE hist COMPUTE STATISTICS;
ANALYZE TABLE dist COMPUTE STATISTICS;
ANALYZE TABLE item COMPUTE STATISTICS;
ANALYZE TABLE ware COMPUTE STATISTICS;
ANALYZE TABLE nord COMPUTE STATISTICS;
ANALYZE index iware COMPUTE STATISTICS;
ANALYZE index idist COMPUTE STATISTICS;
ANALYZE index iitem COMPUTE STATISTICS;
ANALYZE index icust1 COMPUTE STATISTICS;
ANALYZE index icust2 COMPUTE STATISTICS;
ANALYZE index istok COMPUTE STATISTICS;
ANALYZE index iordr1 COMPUTE STATISTICS;
ANALYZE index iordr2 COMPUTE STATISTICS;

set echo off;
spool off;

exit sql.sqlcode;
```

Initipay.sql

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

exit;
```

Isneg.sh

```
#!/bin/sh
# exit true if negative, else false

if test `$tpcc_bcexpr "$*" | cut -b1` = -; then
  exit 0
else
  exit 1
fi
```

Iso_all.sh

```
#!/bin/sh
#
# $Header: iso_all.sh 01-jun-98.19:10:48 skareenh Exp $
#
# iso_all.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      iso_all.sh
#
#    DESCRIPTION
#      <short description of component this file declares/defines>
#
#    NOTES
#      <other useful comments, qualifications, etc.>
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

> isov.out

for I in 1 2 3 4 5 6 7 8 9
do
echo "++++++++++++++++++++++++++++++++" >> isov.out
echo iso${I}.sh >> isov.out
iso${I}.sh >> isov.out 2>&1
done

echo "++++++++++++++++++++++++++++++++" >> isov.out
echo $I >> isov.out
atoma.sh >> isov.out 2>&1
atomc.sh >> isov.out 2>&1
```

Iso1.sh

```
#!/bin/sh
#
# $Header: iso1.sh 01-jun-98.19:07:39 skareenh Exp $
#
# iso1.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      iso1.sh
#
#    DESCRIPTION
#                Perform TPC-C isolation test 1.
#
#    NOTES
#                iso1.sh [options]
#
#                options:
#        -o <output file>      : name of output file
#        -h or -help           : print list of arguments
#        -net <TWO_TASK alias> : SQL*Net V2 connect string
#                                                                alias for remote node
#
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

OFILE="iso1.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>      : name of output file"
```

```
    echo "  -h or -help             : print list of arguments"
    echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
        exit 0
        ;;
      -net)
        shift
        if [ "$1" != "" ]
        then
          TT=$1
          shift
        fi
        ;;
      *)
        echo "Bad argument: $1"
        echo "Use -help option to see correct usage."
        exit 1
        ;;
    esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
    echo "Error: output file is not specified"
    echo "Use -help option to see correct usage."
    exit 1
fi

#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
     SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
     mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# stored procedures directory
#
SPRDIR=${BENCH_HOME}/tpc/tpcc/stored_proc

#
# If this is OPS isolation test, load stored procedures of new order and
# order status that dump what node they are executed on.
#
if [ "$TT" != "" ]
then
    echo "Loading dump versions of new and ord stored procedures."
    sqlplus tpcc/tpcc @${SQLDIR}/new.sql
    sqlplus tpcc/tpcc @${SQLDIR}/ord.sql
fi


#
# txn = 3 (order status), w_id = 5, d_id = 6, c_id = 12, bylastname = 0,
# txn = 0 (quit)
#
cat > iso1.in0 <<!
3
5
6
12
0
0
!

#
# txn = 1 (new order), w_id = 5, d_id = 6, c_id = 12, o_ol_cnt = 5
# order_lines:      i_id        supply_w_id      quantity
#                   1           5           5
#                   2           5           5
#                   3           5           5
#                   4           5           5
#                   5           5           5
# txn = 0 (quit)
cat > iso1.in1 <<!
1
5
6
12
5
1
5
5
2
5
5
3
5
5
4
5
5
5
5
5
0
!

#
# If this is OPS isolation test, start plsql_mon to catch trace output
# from all transactions.
#
if [ "$TT" != "" ]
then
    ${SRCDIR}/plsql_mon system/manager 5 20 | tee iso1.out4 &
    ${SRCDIR}/plsql_mon system/manager@${TT} 5 20 | tee iso1.out5 &
else
    touch iso1.out5
    ${SRCDIR}/plsql_mon system/manager 5 20 | tee iso1.out4 &
fi

#
# Do initial order status transaction (on node 2 if OPS).
#
if [ "$TT" != "" ]
then
    TWO_TASK=$TT
    export TWO_TASK
    set local=$TT
    ${SRCDIR}/tpcc_acid.exe < iso1.in0 | tee iso1.out0
    unset TWO_TASK
    unset local
else
    ${SRCDIR}/tpcc_acid.exe < iso1.in0 | tee iso1.out0
fi

#
# Start New Order T1, sleep for 30 sec prior to commit, then commit.
#
sleep 5
${SRCDIR}/tpccacid_newc.exe < iso1.in1 | tee iso1.out1 &

#
# Sleep for 15 sec and do another order status (on node 2 if OPS).
#
sleep 15
if [ "$TT" != "" ]
then
    TWO_TASK=$TT
    export TWO_TASK
    set local=$TT
    ${SRCDIR}/tpcc_acid.exe   < iso1.in0 | tee iso1.out2
    unset TWO_TASK
    unset local
else
    ${SRCDIR}/tpcc_acid.exe   < iso1.in0 | tee iso1.out2
fi

#
# Do order status transaction one more time (on node 2 if OPS)
# after another 30 sec.
#
sleep 30
if [ "$TT" != "" ]
then
    TWO_TASK=$TT
    export TWO_TASK
    set local=$TT
    ${SRCDIR}/tpcc_acid.exe < iso1.in0 | tee iso1.out3
    unset TWO_TASK
```

```
   unset local
else
   ${SRCDIR}/tpcc_acid.exe < iso1.in0 | tee iso1.out3
fi

#
# Wait for all transactions to complete.
#
wait


#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!
Input and Output of Initial Order Status Transaction (T0) (on node 2 if OPS)
--------------------------------------------------------------------------------

!

cat iso1.out0 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of New Order Transaction (T1) (on node 1 if OPS)
-----------------------------------------------------------------

!

cat iso1.out1 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of Second Order Status Transaction (T2) (on node 2 if OPS)
--------------------------------------------------------------------------------

!

cat iso1.out2 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of Last Order Status Transaction (T3) (on node 2 if OPS)
--------------------------------------------------------------------------------

!

cat iso1.out3 >> ${ODIR}/${OFILE}


if [ "$TT" != "" ]
then

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 1
------------------------------------------------

!

cat iso1.out4 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 2
------------------------------------------------

!

cat iso1.out5 >> ${ODIR}/${OFILE}

else

cat >> ${ODIR}/${OFILE} <<!


Txn info
--------

!

cat iso1.out4  >> ${ODIR}/${OFILE}
fi


rm -f iso1.in0 iso1.in1 iso1.out0 iso1.out1 iso1.out2 iso1.out3 \
     iso1.out4 iso1.out5
```

Iso2.sh

```
#!/bin/sh
#
# $Header: iso2.sh 01-jun-98.19:08:18 skareenh Exp $
#
# iso2.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      iso2.sh
#
#    DESCRIPTION
#    Perform TPC-C isolation test 2.
#
#    NOTES
#                  iso2.sh [options]
#
#                  options:
#          -o <output file>      : name of output file
#          -h or -help           : print list of arguments
#          -net <TWO_TASK alias> : SQL*Net V2 connect string
#                                                                  alias for remote node
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#


#
# Defaults
#

OFILE="iso2.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
      exit 0
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        TT=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
  echo "Error: output file is not specified"
  echo "Use -help option to see correct usage."
  exit 1
fi
```

```
#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# stored procedures directory
#
SPRDIR=${BENCH_HOME}/tpc/tpcc/stored_proc

#
# If this is OPS isolation test, load stored procedures of new order and
# order status that dump what node they are executed on.
#
if [ "$TT" != "" ]
then
   sqlplus tpcc/tpcc @${SQLDIR}/new.sql
   sqlplus tpcc/tpcc @${SQLDIR}/ord.sql
fi


#
# txn = 3 (order status), w_id = 5, d_id = 6, c_id = 12, bylastname = 0,
# txn = 0 (quit)
#
cat > iso2.in0 <<!
3
5
6
12
0
0
!

#
# txn = 1 (new order), w_id = 5, d_id = 6, c_id = 12, o_ol_cnt = 5
# order_lines:      i_id        supply_w_id      quantity
#                    1             5              5
#                    2             5              5
#                    3             5              5
#                    4             5              5
#                   -1             5              5
# txn = 0 (quit)
cat > iso2.in1 <<!
1
5
6
12
5
1
5
5
2
5
5
3
5
5
4
5
5
-1
5
5
0
!

#
# If this is OPS isolation test, start plsql_mon to catch trace output
# from all transactions.
#
if [ "$TT" != "" ]
then
   ${SRCDIR}/plsql_mon system/manager 5 20 | tee iso2.out4 &
   ${SRCDIR}/plsql_mon system/manager@${TT} 5 20 | tee iso2.out5 &
fi

#
# Do initial order status transaction (on node 2 if OPS).
#
if [ "$TT" != "" ]
then
   TWO_TASK=$TT
   export TWO_TASK
   set local=$TT
   ${SRCDIR}/tpcc_acid.exe  < iso2.in0 | tee iso2.out0
   unset TWO_TASK
   unset local
else
   ${SRCDIR}/tpcc_acid.exe  < iso2.in0 | tee iso2.out0
fi

#
# Start New Order T1, sleep for 30 sec prior to rollback, then rollback.
#
sleep 5
${SRCDIR}/tpccacid_newa.exe < iso2.in1 | tee iso2.out1 &

#
# Sleep for 15 sec and do another order status (on node 2 if OPS).
#
sleep 15
if [ "$TT" != "" ]
then
   TWO_TASK=$TT
   export TWO_TASK
   set local=$TT
   ${SRCDIR}/tpcc_acid.exe   < iso2.in0 | tee iso2.out2
   unset TWO_TASK
   unset local
else
   ${SRCDIR}/tpcc_acid.exe   < iso2.in0 | tee iso2.out2
fi

#
# Do order status transaction one more time (on node 2 if OPS)
# after another 30 sec.
#
sleep 30
if [ "$TT" != "" ]
then
   TWO_TASK=$TT
   export TWO_TASK
   set local=$TT
   ${SRCDIR}/tpcc_acid.exe   < iso2.in0 | tee iso2.out3
   unset TWO_TASK
   unset local
else
   ${SRCDIR}/tpcc_acid.exe   < iso2.in0 | tee iso2.out3
fi

#
# Wait for all transactions to complete.
#
wait


#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!
Input and Output of Initial Order Status Transaction (T0) (on node 2 if OPS)
--------------------------------------------------------------------------------

!

cat iso2.out0 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of New Order Transaction (T1) (on node 1 if OPS)
-----------------------------------------------------------------

!

cat iso2.out1 >> ${ODIR}/${OFILE}
```

```
cat >> ${ODIR}/${OFILE} <<!


Input and Output of Second Order Status Transaction (T2) (on node 2 if OPS)
--------------------------------------------------------------------------------

!

cat iso2.out2 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of Last Order Status Transaction (T3) (on node 2 if OPS)
--------------------------------------------------------------------------------

!

cat iso2.out3 >> ${ODIR}/${OFILE}


if [ "$TT" != "" ]
then

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 1
------------------------------------------------

!

cat iso2.out4 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 2
------------------------------------------------

!

cat iso2.out5 >> ${ODIR}/${OFILE}

fi

rm -f iso2.in0 iso2.in1 iso2.out0 iso2.out1 iso2.out2 iso2.out3 \
    iso2.out4 iso2.out5
```

Iso3.sh

```
#!/bin/sh
#
# $Header: iso3.sh 01-jun-98.19:08:36 skareenh Exp $
#
# iso3.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      iso3.sh
#
#    DESCRIPTION
#    Perform TPC-C isolation test 3.
#
#    NOTES
#                  iso3.sh [options]
#
#                  options:
#          -o <output file>      : name of output file
#          -h or -help           : print list of arguments
#          -net <TWO_TASK alias> : SQL*Net V2 connect string
#                                                          alias for remote node
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

#
# Defaults
#

OFILE="iso3.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
      exit 0
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        TT=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
  echo "Error: output file is not specified"
  echo "Use -help option to see correct usage."
  exit 1
fi

#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# stored procedures directory
#
SPRDIR=${BENCH_HOME}/tpc/tpcc/stored_proc

#
# out_file = iso3.out0, w_id = 5, d_id = 8
#
cat > iso3.in0 <<!
iso3.out0
5
8
!

#
```

```
# txn = 1 (new order), w_id = 5, d_id = 8, c_id = 12, o_ol_cnt = 5
# order_lines:      i_id        supply_w_id      quantity
#                   11          7                5
#                   12          7                5
#                   13          7                5
#                   14          7                5
#                   15          7                5
# txn = 0 (quit)
cat > iso3.in1 <<!
1
5
8
12
5
11
7
5
12
7
5
13
7
5
14
7
5
15
7
5
0
!

#
# If this is OPS isolation test, start plsql_mon to catch trace output
# from all transactions.
#
if [ "$TT" != "" ]
then
  ${SRCDIR}/plsql_mon system/manager 5 24 | tee iso3.out3 &
  ${SRCDIR}/plsql_mon system/manager@${TT} 5 24 | tee iso3.out4 &
fi

#
# Check initial d_next_o_id, sleep for 120 sec, check final d_next_o_id
#
#sqlplus tpcc/tpcc @${SQLDIR}/iso3 < iso3.in0 &
sqlplus tpcc/tpcc @${SQLDIR}/iso3 iso3.out0 5 8 &

#
# Sleep for 15 sec, then start New Order T1, sleep for 30 sec prior to commit,
# then commit (on node 2 if OPS).
#
sleep 15
if [ "$TT" != "" ]
then
  TWO_TASK=$TT
  export TWO_TASK
  set local=$TT
  ${SRCDIR}/tpccacid_newc.exe < iso3.in1 | tee iso3.out1 &
  unset TWO_TASK
  unset local
else
  ${SRCDIR}/tpccacid_newc.exe < iso3.in1 | tee iso3.out1 &
fi

#
# Sleep for another 15 sec and do another new order.
#
sleep 15
${SRCDIR}/tpcc_acid.exe < iso3.in1 | tee iso3.out2

#
# Wait for all transactions to complete.
#
wait


#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!
Initial and Final d_next_o_id (on node 1 if OPS)
-------------------------------------------------

!

cat iso3.out0 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of First New Order Transaction (T1) (on node 2 if OPS)
------------------------------------------------------------------------

!

cat iso3.out1 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of Second New Order Transaction (T2) (on node 1 if OPS)
------------------------------------------------------------------------

!

cat iso3.out2 >> ${ODIR}/${OFILE}

if [ "$TT" != "" ]
then

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 1
------------------------------------------------

!

cat iso3.out3 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 2
------------------------------------------------

!

cat iso3.out4 >> ${ODIR}/${OFILE}

fi

rm -f iso3.in0 iso3.in1 iso3.out0 iso3.out1 iso3.out2 iso3.out3 iso3.out4
```

Iso3.sql

```
Rem
Rem $Header: iso3.sql 01-jun-98.19:18:41 skareenh Exp $
Rem
Rem iso3.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      iso3.sql
Rem
Rem    DESCRIPTION
rem       Performs TPC-C isolation tests 3 and 4.
rem       Asks user to input values for ware_id, dist_id and output
rem       file.
Rem
Rem    NOTES
rem                 Usage:   sqlplus tpcc/tpcc @iso3
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem

rem accept logfile     char     prompt  'Enter output file   =>  '

set verify off
spool &&1
set termout on
set echo on

rem accept ware_id     number  prompt  'Enter ware_id       =>  '
rem accept dist_id     number  prompt  'Enter dist_id       =>  '

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM
REM Retrieve initial values.
REM

select d_next_o_id from dist where d_w_id = &&2 and d_id = &&3;
commit;
```

```
REM
REM Sleep for 120 seconds before re-reading values.
REM

execute dbms_lock.sleep (120);
commit;

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM
REM Retrieve final values.
REM

select d_next_o_id from dist where d_w_id = &&2 and d_id = &&3;
commit;

spool off
exit
```

Iso4.sh

```
#!/bin/sh
#
# $Header: iso4.sh 01-jun-98.19:08:55 skareenh Exp $
#
# iso4.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      iso4.sh
#
#    DESCRIPTION
#                  Perform TPC-C isolation test 4.
#
#    NOTES
#                  iso4.sh [options]
#
#                  options:
#          -o <output file>      : name of output file
#          -h or -help           : print list of arguments
#          -net <TWO_TASK alias> : SQL*Net V2 connect string
#                                                          alias for remote node
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

#
# Defaults
#

OFILE="iso4.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
      exit 0
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        TT=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
  echo "Error: output file is not specified"
  echo "Use -help option to see correct usage."
  exit 1
fi

#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
  mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# stored procedures directory
#
SPRDIR=${BENCH_HOME}/tpc/tpcc/stored_proc

#
# If this is OPS isolation test, load stored procedure of
# new order that dumps what node it is executed on.
#
if [ "$TT" != "" ]
then
  sqlplus tpcc/tpcc @${SQLDIR}/new.sql
fi


#
# out_file = iso4.out0, w_id = 5, d_id = 8
#
cat > iso4.in0 <<!
iso4.out0
5
8
!

#
# txn = 1 (new order), w_id = 5, d_id = 8, c_id = 12, o_ol_cnt = 5
# order_lines:      i_id        supply_w_id      quantity
#                   11          7                5
#                   12          7                5
#                   13          7                5
#                   14          7                5
#                   -1          7                5
# txn = 0 (quit)
cat > iso4.in1 <<!
1
5
8
12
5
11
7
5
12
7
```

```
5
13
7
5
14
7
5
-1
7
5
0
!

#
# txn = 1 (new order), w_id = 5, d_id = 8, c_id = 12, o_ol_cnt = 5
# order_lines:      i_id        supply_w_id      quantity
#                   11          7                5
#                   12          7                5
#                   13          7                5
#                   14          7                5
#                   15          7                5
# txn = 0 (quit)
cat > iso4.in2 <<!
1
5
8
12
5
11
7
5
12
7
5
13
7
5
14
7
5
15
7
5
0
!

#
# If this is OPS isolation test, start plsql_mon to catch trace output
# from all transactions.
#
if [ "$TT" != "" ]
then
  ${SRCDIR}/plsql_mon system/manager 5 24 | tee iso4.out3 &
  ${SRCDIR}/plsql_mon system/manager@${TT} 5 24 | tee iso4.out4 &
fi

#
# Check initial d_next_o_id, sleep for 120 sec, check final d_next_o_id
#
#sqlplus tpcc/tpcc @${SQLDIR}/iso3 < iso4.in0 &
sqlplus tpcc/tpcc @${SQLDIR}/iso3 iso4.out0 5 8 &

#
# Sleep for 15 sec, then start New Order T1, sleep for 30 sec prior to commit,
# then commit (on node 2 if OPS).
#
sleep 15
if [ "$TT" != "" ]
then
  TWO_TASK=$TT
  export TWO_TASK
  set local=$TT
  ${SRCDIR}/tpccacid_newc.exe < iso4.in1 | tee iso4.out1 &
  unset TWO_TASK
  unset local
else
  ${SRCDIR}/tpccacid_newc.exe < iso4.in1 | tee iso4.out1 &
fi

#
# Sleep for another 15 sec and do another new order.
#
sleep 15
${SRCDIR}/tpcc_acid.exe < iso4.in2 | tee iso4.out2

#
# Wait for all transactions to complete.
#
wait

#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!

Initial and Final d_next_o_id (on node 1 if OPS)
------------------------------------------------

!

cat iso4.out0 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of First New Order Transaction (T1) (on node 2 if OPS)
-----------------------------------------------------------------------

!

cat iso4.out1 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of Second New Order Transaction (T2) (on node 1 if OPS)
------------------------------------------------------------------------

!

cat iso4.out2 >> ${ODIR}/${OFILE}

if [ "$TT" != "" ]
then

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 1
------------------------------------------------

!

cat iso4.out3 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 2
------------------------------------------------

!

cat iso4.out4 >> ${ODIR}/${OFILE}

fi

rm -f iso4.in0 iso4.in1 iso4.in2 iso4.out0 iso4.out1 iso4.out2 \
    iso4.out3 iso4.out4
```

Iso5.sh

```
#!/bin/sh
#
# $Header: iso5.sh 01-jun-98.19:09:13 skareenh Exp $
#
# iso5.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#   NAME
#     iso5.sh
#
#   DESCRIPTION
#                 Perform TPC-C isolation test 5.
#
#   NOTES
#                 iso5.sh [options]
#
#      options:
#         -o <output file>      : name of output file
#         -h or -help           : print list of arguments
#         -net <TWO_TASK alias> : SQL*Net V2 connect string
#                                                          alias for remote node
#
#   MODIFIED   (MM/DD/YY)
#   skareenh    06/01/98 - Creation
#

#
# Defaults
#
```

```
OFILE="iso5.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
      exit 0
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        TT=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
  echo "Error: output file is not specified"
  echo "Use -help option to see correct usage."
  exit 1
fi

#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# stored procedures directory
#
SPRDIR=${BENCH_HOME}/tpc/tpcc/stored_proc

#
# Load stored procedure of delivery that dumps messages.
#
# sqlplus tpcc/tpcc @${SQLDIR}/del_iso5.sql

#
# If this is OPS isolation test, load stored procedure of
# payment that dumps what node it is executed on.
#
if [ "$TT" != "" ]
then
  sqlplus tpcc/tpcc @${SQLDIR}/pay.sql
fi

#
# Get customer id that the delivery and payment transaction will collide on.
# w_id = 5, d_id = 5
# Note: The stored procedure of the ACID delivery transaction has been
#       hardcoded to dump out info only when d_id = 5, so DON't use
#       a different value of d_id here.
#
CUST_ID=`sqlplus tpcc/tpcc @${SQLDIR}/get_cust 5 5 | grep THE_VALUE | \
        awk '{print $2}'`
echo "Warehouse ID = 5, District ID = 5, Customer ID = $CUST_ID"

#
# out_file = iso5.out0, w_id = 5, d_id = 5, c_id = $CUST_ID
#
cat > iso5.in0 <<!
iso5.out0
5
5
$CUST_ID
!

#
# txn = 4 (delivery), w_id = 5, o_carrier_id = 9, txn = 0 (quit)
#
cat > iso5.in1 <<!
4
5
9
0
!

#
# txn = 2 (payment), w_id = 5, d_id = 5, c_w_id = 5, c_d_id = 5,
# h_amount = 300.0, c_id = $CUST_ID, bylastname = 0, txn = 0 (quit)
#
cat > iso5.in2 <<!
2
5
5
5
5
300.0
$CUST_ID
0
0
!

#
# Start plsql_mon to catch trace output from delivery transaction.
#
${SRCDIR}/plsql_mon system/manager 5 24 | tee iso5.out3 &

#
# If this is OPS isolation test, start plsql_mon to catch trace output
# from all transactions.
#
if [ "$TT" != "" ]
then
  ${SRCDIR}/plsql_mon system/manager@${TT} 5 24 | tee iso5.out4 &
fi

#
# Check initial c_balance, sleep for 120 sec, check final c_balance.
#
#sqlplus tpcc/tpcc @${SQLDIR}/iso5 < iso5.in0 &
sqlplus tpcc/tpcc @${SQLDIR}/iso5 iso5.out0 5 5 $CUST_ID &

#
# Sleep for 15 sec, then start Delivery T1, sleep for 60 sec prior to commit,
# then commit.
#
sleep 15
${SRCDIR}/tpccacid_iso5.exe < iso5.in1 | tee iso5.out1 &

#
# Sleep for another 30 sec and do a Payment (on node 2 if OPS).
#
sleep 30
if [ "$TT" != "" ]
then
  TWO_TASK=$TT
  export TWO_TASK
  set local=$TT
  ${SRCDIR}/tpcc_acid.exe < iso5.in2 | tee iso5.out2
  unset TWO_TASK
```

```
   unset local
else
   ${SRCDIR}/tpcc_acid.exe < iso5.in2 | tee iso5.out2
fi

#
# Wait for all transactions to complete.
#
wait


#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!
Warehouse ID = 11, District ID = 5, Customer ID = $CUST_ID



!

cat >> ${ODIR}/${OFILE} <<!
Initial and Final c_balance (on node 1 if OPS)
-----------------------------------------------

!

cat iso5.out0 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!



Input and Output of Delivery Transaction (T1) (on node 1 if OPS)
-----------------------------------------------------------------

!

cat iso5.out1 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!



Trace output from Delivery Transaction (T1) (on node 1 if OPS)
---------------------------------------------------------------

!

cat iso5.out3 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!



Input and Output of Payment Transaction (T2) (on node 2 if OPS)
---------------------------------------------------------------

!

cat iso5.out2 >> ${ODIR}/${OFILE}

if [ "$TT" != "" ]
then

cat >> ${ODIR}/${OFILE} <<!



Trace output for Transactions Executed on Node 2
-------------------------------------------------

!

cat iso5.out4 >> ${ODIR}/${OFILE}

fi

rm -f iso5.in0 iso5.in1 iso5.in2 iso5.out0 iso5.out1 iso5.out2 iso5.out3 \
     iso5.out4
```

Iso6.sh

```
#!/bin/sh
#
# $Header: iso6.sh 01-jun-98.19:09:31 skareenh Exp $
#
# iso6.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#   NAME
#     iso6.sh
#
#   DESCRIPTION
#                 Perform TPC-C isolation test 6.
#
#   NOTES
#                 iso6.sh [options]
#
#                 options:
#         -o <output file>      : name of output file
#         -h or -help           : print list of arguments
#         -net <TWO_TASK alias> : SQL*Net V2 connect string
#                                                                  alias for remote node
#
#   MODIFIED   (MM/DD/YY)
#   skareenh    06/01/98 - Creation
#

#
# Defaults
#

OFILE="iso6.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
   case $1 in
     -o|-ofile)
        shift
        if [ "$1" != "" ]
        then
          OFILE=$1
          shift
        fi
        ;;
     -h|-help)
   echo "Options:"
   echo "  -o <output file>       : name of output file"
   echo "  -h or -help            : print list of arguments"
   echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
        exit 0
        ;;
     -net)
        shift
        if [ "$1" != "" ]
        then
          TT=$1
          shift
        fi
        ;;
     *)
        echo "Bad argument: $1"
        echo "Use -help option to see correct usage."
        exit 1
        ;;
   esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
   echo "Error: output file is not specified"
   echo "Use -help option to see correct usage."
   exit 1
fi

#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
    SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
```

```
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# stored procedures directory
#
SPRDIR=${BENCH_HOME}/tpc/tpcc/stored_proc

#
# Load stored procedure of delivery that dumps messages and rollbacks.
#
# sqlplus tpcc/tpcc @${SQLDIR}/del_iso6.sql

#
# If this is OPS isolation test, load stored procedure of
# payment that dumps what node it is executed on.
#
if [ "$TT" != "" ]
then
  sqlplus tpcc/tpcc @${SQLDIR}/pay.sql
fi

#
# Get customer id that the delivery and payment transaction will collide on.
# w_id = 5, d_id = 5
# Note: The stored procedure of the ACID delivery transaction has been
#       hardcoded to dump out info only when d_id = 5, so DON't use
#       a different value of d_id here.
#
CUST_ID=`sqlplus tpcc/tpcc @${SQLDIR}/get_cust 5 5 | grep THE_VALUE | \
       awk '{print $2}'`
echo "Warehouse ID = 5, District ID = 5, Customer ID = $CUST_ID"

#
# out_file = iso6.out0, w_id = 5, d_id = 5, c_id = $CUST_ID
#
cat > iso6.in0 <<!
iso6.out0
5
5
$CUST_ID
!

#
# txn = 4 (delivery), w_id = 5, o_carrier_id = 9, txn = 0 (quit)
#
cat > iso6.in1 <<!
4
5
9
0
!

#
# txn = 2 (payment), w_id = 5, d_id = 5, c_w_id = 5, c_d_id = 5,
# h_amount = 300.0, c_id = $CUST_ID, bylastname = 0, txn = 0 (quit)
#
cat > iso6.in2 <<!
2
5
5
5
5
300.0
$CUST_ID
0
0
!

#
# Start plsql_mon to catch trace output from delivery transaction.
#
${SRCDIR}/plsql_mon system/manager 5 24 | tee iso6.out3 &

#
# If this is OPS isolation test, start plsql_mon to catch trace output
# from all transactions.
#
if [ "$TT" != "" ]
then
  ${SRCDIR}/plsql_mon system/manager@${TT} 5 24 | tee iso6.out4 &
fi

#
# Check initial c_balance, sleep for 120 sec, check final c_balance.
#
#sqlplus tpcc/tpcc @${SQLDIR}/iso5 < iso6.in0 &
sqlplus tpcc/tpcc @${SQLDIR}/iso5 iso6.out0 5 5 $CUST_ID &

#
# Sleep for 15 sec, then start Delivery T1, sleep for 60 sec prior to rollback,
# then rollback.
#
sleep 15
${SRCDIR}/tpccacid_iso6.exe < iso6.in1 | tee iso6.out1 &

#
# Sleep for another 30 sec and do a Payment (on node 2 if OPS).
#
sleep 30
if [ "$TT" != "" ]
then
  TWO_TASK=$TT
  export TWO_TASK
  set local=$TT
  ${SRCDIR}/tpcc_acid.exe < iso6.in2 | tee iso6.out2
  unset TWO_TASK
  unset local
else
  ${SRCDIR}/tpcc_acid.exe < iso6.in2 | tee iso6.out2
fi

#
# Wait for all transactions to complete.
#
wait

#
# If this is OPS isolation test, restore stored procedure of payment.
#
if [ "$TT" != "" ]
then
  sqlplus tpcc/tpcc @${SPRDIR}/pay.sql
fi

#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!
Warehouse ID = 11, District ID = 5, Customer ID = $CUST_ID



!

cat >> ${ODIR}/${OFILE} <<!
Initial and Final c_balance (on node 1 if OPS)
-----------------------------------------------

!

cat iso6.out0 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of Delivery Transaction (T1) (on node 1 if OPS)
-----------------------------------------------------------------

!

cat iso6.out1 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Trace output from Delivery Transaction (T1) (on node 1 if OPS)
---------------------------------------------------------------

!

cat iso6.out3 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of Payment Transaction (T2) (on node 2 if OPS)
---------------------------------------------------------------

!

cat iso6.out2 >> ${ODIR}/${OFILE}

if [ "$TT" != "" ]
then
```

```
cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 2
------------------------------------------------

!

cat iso6.out4 >> ${ODIR}/${OFILE}

fi

rm -f iso6.in0 iso6.in1 iso6.in2 iso6.out0 iso6.out1 iso6.out2 iso6.out3 \
    iso6.out4
```

Iso7.sh

```
#!/bin/sh
#
# $Header: iso7.sh 01-jun-98.19:09:49 skareenh Exp $
#
# iso7.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      iso7.sh
#
#    DESCRIPTION
#                  Perform TPC-C isolation test 7.
#
#    NOTES
#                  iso7.sh [options]
#
#                  options:
#          -o <output file>      : name of output file
#          -h or -help           : print list of arguments
#          -net <TWO_TASK alias> : SQL*Net V2 connect string
#                                                                  alias for remote node
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

#
# $Header: iso7.sh 7030100.2 96/04/24 15:25:27 plai Generic<base> $ Copyr (c) 1995 Oracle
#

#
#==============================================================+
#         Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                |
#                   All Rights Reserved                        |
#==============================================================+
# FILENAME
#    iso7.sh
# DESCRIPTION
#

#
# Defaults
#

OFILE="iso7.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
      exit 0
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        TT=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
  echo "Error: output file is not specified"
  echo "Use -help option to see correct usage."
  exit 1
fi

#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# stored procedures directory
#
SPRDIR=${BENCH_HOME}/tpc/tpcc/stored_proc

#
# If this is OPS isolation test, load stored procedure of
# new order that dumps what node it is executed on.
#
if [ "$TT" != "" ]
then
  sqlplus tpcc/tpcc @${SQLDIR}/new.sql
fi

#
# out_file = iso7.out1, i_id1 = 725, i_id2 = 9601
#
cat > iso7.in1 <<!
iso7.out1
725
9601
!

#
# txn = 1 (new order), w_id = 5, d_id = 9, c_id = 1234, o_ol_cnt = 5
# order_lines:      i_id        supply_w_id      quantity
#                    113           15              5
#                    725           15              5
#                   9601            15             5
#                    143           15              5
#                    725           15              5
# txn = 0 (quit)
cat > iso7.in2 <<!
1
5
9
```

```
1234
5
113
15
5
725
15
5
9601
15
5
143
15
5
725
15
5
0
!

#
# out_file = iso7.out3, i_id1 = 725, i_id2 = 9601
#
cat > iso7.in3 <<!
iso7.out3
725
9601
!

#
# out_file = iso7.out4, i_id1 = 725, i_id2 = 9601
#
cat > iso7.in4 <<!
iso7.out4
725
9601
!

#
# If this is OPS isolation test, start plsql_mon to catch trace output
# from all transactions.
#
if [ "$TT" != "" ]
then
  ${SRCDIR}/plsql_mon system/manager 5 24 | tee iso7.out5 &
fi

#
# Check initial i_price.
#
#sqlplus tpcc/tpcc @${SQLDIR}/iso7_s < iso7.in1
sqlplus tpcc/tpcc @${SQLDIR}/iso7_s iso7.out1 725 9601

#
# Start New Order T2, select i_price from item, sleep for 30 sec, then
# re-select i_price from item.
#
${SRCDIR}/tpccacid_iso7.exe < iso7.in2 | tee iso7.out2 &

#
# Sleep for 15 sec and start T3 to increase i_price of the 2 items by 10%
# (on node 2 if OPS).
#
sleep 15
if [ "$TT" != "" ]
then
#   sqlplus tpcc/tpcc@${TT} @${SQLDIR}/iso7_u < iso7.in3
   sqlplus tpcc/tpcc@${TT} @${SQLDIR}/iso7_u iso7.out3 725 9601
else
#   sqlplus tpcc/tpcc @${SQLDIR}/iso7_u < iso7.in3
   sqlplus tpcc/tpcc @${SQLDIR}/iso7_u iso7.out3 725 9601
fi

#
# Wait for all transactions to complete.
#
wait

#
# If this is OPS isolation test, restore stored procedure of new order.
#
if [ "$TT" != "" ]
then
   sqlplus tpcc/tpcc @${SPRDIR}/new.sql
fi

#
# Re-check final i_price.
#
#sqlplus tpcc/tpcc @${SQLDIR}/iso7_s < iso7.in4
sqlplus tpcc/tpcc @${SQLDIR}/iso7_s iso7.out4 725 9601

#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!
Initial i_price (T1) (on node 1 if OPS)
----------------------------------------

!

cat iso7.out1 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of New Order Transaction (T2) (on node 1 if OPS)
-----------------------------------------------------------------

!

cat iso7.out2 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Updated i_price (T3) (on node 2 if OPS)
----------------------------------------

!

cat iso7.out3 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Final i_price (T4) (on node 1 if OPS)
----------------------------------------

!

cat iso7.out4 >> ${ODIR}/${OFILE}

if [ "$TT" != "" ]
then

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 1
------------------------------------------------

!

cat iso7.out5 >> ${ODIR}/${OFILE}

fi

rm -f iso7.in1 iso7.in2 iso7.in3 iso7.in4 iso7.out1 iso7.out2 iso7.out3 \
      iso7.out4 iso7.out5
```

Iso7_s.sql

```
Rem
Rem $Header: iso7_s.sql 01-jun-98.19:19:17 skareenh Exp $
Rem
Rem iso7_s.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      iso7_s.sql
Rem
Rem    DESCRIPTION
rem       Performs TPC-C isolation test 7.
rem       Asks user to input values for item1, item2 and output file.
Rem
Rem    NOTES
rem                 Usage:   sqlplus tpcc/tpcc @iso7_s
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem

column i_price/100 format 999999.99

rem accept logfile     char     prompt  'Enter output file   =>  '

set verify off
spool &&1
set termout on
set echo on
```

```
rem accept item_id1     number  prompt  'Enter first item_id       => '
rem accept item_id2     number  prompt  'Enter second item_id      => '

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM
REM Retrieve initial values.
REM

select i_price/100 from item where i_id = &&2;
select i_price/100 from item where i_id = &&3;
commit;

spool off
exit
```

Iso7.u.sql

```
Rem
Rem $Header: iso7_u.sql 01-jun-98.19:19:35 skareenh Exp $
Rem
Rem iso7_u.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
Rem
Rem    NAME
Rem      iso7_u.sql
Rem
Rem    DESCRIPTION
rem       Performs TPC-C isolation test 7.
rem       Asks user to input values for item1, item2 and output file.
Rem
Rem    NOTES
rem                 Usage:   sqlplus tpcc/tpcc @iso7_u
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    skareenh    06/01/98 - Created
Rem

column i_price/100 format 999999.99

rem accept logfile     char     prompt  'Enter output file   => '

set verify off
spool &&1
set termout on
set echo on

rem accept item_id1     number  prompt  'Enter first item_id       => '
rem accept item_id2     number  prompt  'Enter second item_id      => '

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM
REM Get node number.
REM

select substr(value,1,5) node_num from v$parameter
   where name = 'instance_number';

update item set i_price = i_price * 1.1 where i_id = &&2;
update item set i_price = i_price * 1.1 where i_id = &&3;

select i_price/100 from item where i_id = &&2;
select i_price/100 from item where i_id = &&3;

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

commit;

spool off
exit
```

Iso8

```
SQL*Plus: Release 10.0.0.0.0 - Development on Wed Nov 6 23:08:16 2002

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Error accessing PRODUCT_USER_PROFILE
Warning:  Product user profile information not loaded!
You may need to run PUPBLD.SQL as SYSTEM

Connected to:
Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option

SQL>
SQL> rem accept ware_id     number  prompt  'Enter ware_id                    => '
SQL> rem accept dist_id     number  prompt  'Enter dist_id   => '
SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:08:16

SQL>
SQL> REM
SQL> REM Delete all new orders from that warehouse and dist.
SQL> REM
SQL> REM delete from new_order where no_w_id = &&2 and no_d_id = &&3;
SQL>
SQL> update nord set no_d_id = 11
  2                 where no_w_id = &&2 and no_d_id = &&3;

909 rows updated.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:08:16

SQL>
SQL> spool off
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option
PL/SQL Monitor started.  Hit <return> to stop.
Delivery started at 23:08:35 on node 0
Delivery found no new order at district 5 at 23:08:35
Enter txn type (0=quit, 1=new, 2=pay, 3=ord, 4=del, 5=sto):
---------- NEW ORDER TRANSACTION ----------
Enter w_id: Enter d_id: Enter c_id: Enter o_ol_cnt: Enter i_id for order line  1: Enter supply_w_id
for order line  1: Enter quantity for order line  1: Enter i_id for order line  2: Enter supply_w_id
for order line  2: Enter quantity for order line  2: Enter i_id for order line  3: Enter supply_w_id
for order line  3: Enter quantity for order line  3: Enter i_id for order line  4: Enter supply_w_id
for order line  4: Enter quantity for order line  4: Enter i_id for order line  5: Enter supply_w_id
for order line  5: Enter quantity for order line  5:
---------- NEW ORDER (in) 06-Nov-2002.23:09:04 ----------
w_id = 10
d_id = 5
c_id = 1999
o_ol_cnt = 5
   i_id = 61, sup_w_id = 13, quantity = 5
   i_id = 62, sup_w_id = 13, quantity = 5
   i_id = 63, sup_w_id = 13, quantity = 5
   i_id = 64, sup_w_id = 13, quantity = 5
   i_id = 65, sup_w_id = 13, quantity = 5
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0

---------- NEW ORDER (out) 06-Nov-2002.23:09:04 ----------
o_id = 3781
o_ol_cnt = 5
c_last = PRIESEBAR
c_credit = GC
c_discount = 0.1872
```

```
w_tax = 0.1274
d_tax = 0.1693
o_entry_d = 06-nov-02.11:09:04
total_amount = 884.61
   i_name cEIh3UoYgTXoeA6GZu6vk0, s_qty 82, bg G, i_pr 31.30, ol_amt 156.50
   i_name XVn1CP72OCUkcDkowsLM, s_qty 50, bg G, i_pr 11.93, ol_amt 59.65
   i_name ZddeMbpzpW0lEpEdfPqNQu, s_qty 100, bg G, i_pr 13.01, ol_amt 65.05
   i_name F4gEkSRRL8exJhjIdeYtme99, s_qty 53, bg G, i_pr 74.40, ol_amt 372.00
   i_name k8MThBuLS8R6dP, s_qty 70, bg G, i_pr 46.28, ol_amt 231.40
invalid item? = 0
retry = 0

-------------------------------------------------

Enter txn type (0=quit, 1=new, 2=pay, 3=ord, 4=del, 5=sto):
SQL*Plus: Release 10.0.0.0.0 - Development on Wed Nov 6 23:09:19 2002

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Error accessing PRODUCT_USER_PROFILE
Warning:  Product user profile information not loaded!
You may need to run PUPBLD.SQL as SYSTEM

Connected to:
Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option

SQL>
SQL> rem accept ware_id     number  prompt  'Enter ware_id                  => '
SQL> rem accept dist_id     number  prompt  'Enter dist_id   => '
SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:09:19

SQL>
SQL> REM
SQL> REM Select the new order just entered.
SQL> REM
SQL>
SQL>  select no_w_id, no_d_id, no_o_id
  2                 from nord
  3                 where no_w_id = &&2 and no_d_id = &&3 and no_o_id =
  4                 (select d_next_o_id - 1
  5                     from dist
  6                     where d_w_id = &&2 and d_id = &&3);

   NO_W_ID    NO_D_ID       NO_O_ID
---------- ---------- ----------
                10                  5                3781

SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:09:19

SQL>
SQL> spool off
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option
Enter txn type (0=quit, 1=new, 2=pay, 3=ord, 4=del, 5=sto):
---------- DELIVERY TRANSACTION ----------
Enter w_id: Enter o_carrier_id:
---------- DELIVERY (in) 06-Nov-2002.23:08:34 ----------
w_id = 10
o_carrier_id = 3

---------- DELIVERY (out) 06-Nov-2002.23:09:36 ----------
o_id: 2872 2872 2872 2872 0 2872 2872 2872 2872 2872
retry = 0

-------------------------------------------------

Enter txn type (0=quit, 1=new, 2=pay, 3=ord, 4=del, 5=sto): Delivery found no new order at
district 5 at 23:09:36

SQL*Plus: Release 10.0.0.0.0 - Development on Wed Nov 6 23:11:21 2002

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Error accessing PRODUCT_USER_PROFILE
Warning:  Product user profile information not loaded!
You may need to run PUPBLD.SQL as SYSTEM

Connected to:
Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option

SQL>
SQL> rem accept ware_id     number  prompt  'Enter ware_id                  => '
SQL> rem accept dist_id     number  prompt  'Enter dist_id   => '
SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:11:21

SQL>
SQL> REM
SQL> REM Restore all new orders.
SQL> REM
SQL>
SQL> update nord set no_d_id = &&3
  2                 where no_w_id = &&2 and no_d_id = 11;
update nord set no_d_id = 5
*
ERROR at line 1:
ORA-03113: end-of-file on communication channel


SQL>
SQL> commit;
commit
*
ERROR at line 1:
ORA-03114: not connected to ORACLE


SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;
select to_char (sysdate, 'HH24:MI:SS') timestamp from dual
*
ERROR at line 1:
ORA-03114: not connected to ORACLE


SQL>
SQL> spool off
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option
```

Iso8.sh

```
SQL*Plus: Release 10.0.0.0.0 - Development on Wed Nov 6 23:08:16 2002

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Error accessing PRODUCT_USER_PROFILE
Warning:  Product user profile information not loaded!
You may need to run PUPBLD.SQL as SYSTEM

Connected to:
Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option

SQL>
SQL> rem accept ware_id     number  prompt  'Enter ware_id                  => '
SQL> rem accept dist_id     number  prompt  'Enter dist_id   => '
SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:08:16

SQL>
SQL> REM
SQL> REM Delete all new orders from that warehouse and dist.
SQL> REM
```

```
SQL> REM delete from new_order where no_w_id = &&2 and no_d_id = &&3;
SQL>
SQL> update nord set no_d_id = 11
  2                 where no_w_id = &&2 and no_d_id = &&3;

909 rows updated.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:08:16

SQL>
SQL> spool off
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option
PL/SQL Monitor started.  Hit <return> to stop.
Delivery started at 23:08:35 on node 0
Delivery found no new order at district 5 at 23:08:35
Enter txn type (0=quit, 1=new, 2=pay, 3=ord, 4=del, 5=sto):
---------- NEW ORDER TRANSACTION -----------
Enter w_id: Enter d_id: Enter c_id: Enter o_ol_cnt: Enter i_id for order line  1: Enter supply_w_id
for order line  1: Enter quantity for order line  1: Enter i_id for order line  2: Enter supply_w_id
for order line  2: Enter quantity for order line  2: Enter i_id for order line  3: Enter supply_w_id
for order line  3: Enter quantity for order line  3: Enter i_id for order line  4: Enter supply_w_id
for order line  4: Enter quantity for order line  4: Enter i_id for order line  5: Enter supply_w_id
for order line  5: Enter quantity for order line  5:
---------- NEW ORDER (in) 06-Nov-2002.23:09:04 ----------
w_id = 10
d_id = 5
c_id = 1999
o_ol_cnt = 5
   i_id = 61, sup_w_id = 13, quantity = 5
   i_id = 62, sup_w_id = 13, quantity = 5
   i_id = 63, sup_w_id = 13, quantity = 5
   i_id = 64, sup_w_id = 13, quantity = 5
   i_id = 65, sup_w_id = 13, quantity = 5
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0
   i_id = 0, sup_w_id = 0, quantity = 0

---------- NEW ORDER (out) 06-Nov-2002.23:09:04 ----------
o_id = 3781
o_ol_cnt = 5
c_last = PRIESEBAR
c_credit = GC
c_discount = 0.1872
w_tax = 0.1274
d_tax = 0.1693
o_entry_d = 06-nov-02.11:09:04
total_amount = 884.61
   i_name cElh3UoYgTXoeA6GZu6vk0, s_qty 82, bg G, i_pr 31.30, ol_amt 156.50
   i_name XVn1CP72OCUkcDkowsLM, s_qty 50, bg G, i_pr 11.93, ol_amt 59.65
   i_name ZddeMbpzpW0lEpEdfPqNQu, s_qty 100, bg G, i_pr 13.01, ol_amt 65.05
   i_name F4gEkSRRL8exJhjIdeYtme99, s_qty 53, bg G, i_pr 74.40, ol_amt 372.00
   i_name k8MThBuLS8R6dP, s_qty 70, bg G, i_pr 46.28, ol_amt 231.40
invalid item? = 0
retry = 0

-------------------------------------------------

Enter txn type (0=quit, 1=new, 2=pay, 3=ord, 4=del, 5=sto):
SQL*Plus: Release 10.0.0.0.0 - Development on Wed Nov 6 23:09:19 2002

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Error accessing PRODUCT_USER_PROFILE
Warning:  Product user profile information not loaded!
You may need to run PUPBLD.SQL as SYSTEM

Connected to:
Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option

SQL>
SQL> rem accept ware_id     number  prompt  'Enter ware_id                  => '
SQL> rem accept dist_id     number  prompt  'Enter dist_id   => '
SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:09:19

SQL>
SQL> REM
SQL> REM Select the new order just entered.
SQL> REM
SQL>
SQL>  select no_w_id, no_d_id, no_o_id
  2                 from nord
  3                 where no_w_id = &&2 and no_d_id = &&3 and no_o_id =
  4                 (select d_next_o_id - 1
  5                    from dist
  6                    where d_w_id = &&2 and d_id = &&3);

   NO_W_ID    NO_D_ID       NO_O_ID
---------- ---------- ----------
                 10                  5                3781

SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:09:19

SQL>
SQL> spool off
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option
Enter txn type (0=quit, 1=new, 2=pay, 3=ord, 4=del, 5=sto):
---------- DELIVERY TRANSACTION ----------
Enter w_id: Enter o_carrier_id:
---------- DELIVERY (in) 06-Nov-2002.23:08:34 ----------
w_id = 10
o_carrier_id = 3

---------- DELIVERY (out) 06-Nov-2002.23:09:36 ----------
o_id: 2872 2872 2872 2872 0 2872 2872 2872 2872 2872
retry = 0

-------------------------------------------------

Enter txn type (0=quit, 1=new, 2=pay, 3=ord, 4=del, 5=sto): Delivery found no new order at
district 5 at 23:09:36

SQL*Plus: Release 10.0.0.0.0 - Development on Wed Nov 6 23:11:21 2002

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

Error accessing PRODUCT_USER_PROFILE
Warning:  Product user profile information not loaded!
You may need to run PUPBLD.SQL as SYSTEM

Connected to:
Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option

SQL>
SQL> rem accept ware_id     number  prompt  'Enter ware_id                  => '
SQL> rem accept dist_id     number  prompt  'Enter dist_id   => '
SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

TIMESTAM
--------
23:11:21

SQL>
SQL> REM
SQL> REM Restore all new orders.
SQL> REM
SQL>
SQL> update nord set no_d_id = &&3
  2                 where no_w_id = &&2 and no_d_id = 11;
update nord set no_d_id = 5
*
```

```
ERROR at line 1:
ORA-03113: end-of-file on communication channel


SQL>
SQL> commit;
commit
*
ERROR at line 1:
ORA-03114: not connected to ORACLE


SQL>
SQL> REM
SQL> REM Get timestamp.
SQL> REM
SQL>
SQL> select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;
select to_char (sysdate, 'HH24:MI:SS') timestamp from dual
*
ERROR at line 1:
ORA-03114: not connected to ORACLE


SQL>
SQL> spool off
SQL> exit
Disconnected from Oracle9i Enterprise Edition Release 10.0.0.0.0 - 64bit Beta
With the Partitioning option
```

Iso8.sql

```
#!/bin/sh
#
# $Header: iso8.sh 01-jun-98.19:10:09 skareenh Exp $
#
# iso8.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      iso8.sh
#
#    DESCRIPTION
#                  Perform TPC-C isolation test 8.
#
#    NOTES
#                  iso8.sh [options]
#
#                  options:
#          -o <output file>      : name of output file
#          -h or -help           : print list of arguments
#          -net <TWO_TASK alias> : SQL*Net V2 connect string
#                                                                  alias for remote node
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

#
# Defaults
#

OFILE="iso8.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
      exit 0
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        TT=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
  echo "Error: output file is not specified"
  echo "Use -help option to see correct usage."
  exit 1
fi

#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# stored procedures directory
#
SPRDIR=${BENCH_HOME}/tpc/tpcc/stored_proc

#
# Load stored procedure of delivery that dumps messages.
#
# sqlplus tpcc/tpcc @${SQLDIR}/del_iso5.sql

#
# If this is OPS isolation test, load stored procedure of
# new order that dumps what node it is executed on.
#
if [ "$TT" != "" ]
then
  sqlplus tpcc/tpcc @${SQLDIR}/new.sql
fi

#
# out_file = iso8.out0, w_id = 10, d_id = 5
#
cat > iso8.in0 <<!
iso8.out0
10
5
!

#
# txn = 4 (delivery), w_id = 10, o_carrier_id = 3, txn = 0 (quit)
#
cat > iso8.in1 <<!
4
10
3
0
!
```

```
#
# txn = 1 (new order), w_id = 10, d_id = 5, c_id = 1999, o_ol_cnt = 5
# order_lines:      i_id        supply_w_id      quantity
#                   61          13               5
#                   62          13               5
#                   63          13               5
#                   64          13               5
#                   65          13               5
# txn = 0 (quit)
cat > iso8.in2 <<!
1
10
5
1999
5
61
13
5
62
13
5
63
13
5
64
13
5
65
13
5
0
!

#
# out_file = iso8.out4, w_id = 10, d_id = 5
#
cat > iso8.in4 <<!
iso8.out4
10
5
!

#
# out_file = iso8.out5, w_id = 10, d_id = 5
#
cat > iso8.in5 <<!
iso8.out5
10
5
!

#
# Delete all new orders from warehouse 10 district 5
#
#sqlplus tpcc/tpcc @${SQLDIR}/iso8 < iso8.in0
sqlplus tpcc/tpcc @${SQLDIR}/iso8 iso8.out0 10 5

#
# Start plsql_mon to catch trace output from Delivery transaction T1.
# If this is OPS isolation test, start plsql_mon to catch trace output
# from all transactions.
#
if [ "$TT" != "" ]
then
   ${SRCDIR}/plsql_mon system/manager@${TT} 5 36 | tee iso8.out3 &
   ${SRCDIR}/plsql_mon system/manager 5 36 | tee iso8.out6 &
else
   ${SRCDIR}/plsql_mon system/manager 5 36 | tee iso8.out3 &
fi

#
# Do Delivery transaction T1 (on node 2 if OPS),
# find no new order in district 3,
# sleep for 60 sec, then re-check to see if there is still no new order.
#
sleep 15
if [ "$TT" != "" ]
then
   TWO_TASK=$TT
   export TWO_TASK
   set local=$TT
   ${SRCDIR}/tpccacid_iso8.exe < iso8.in1 | tee iso8.out1 &
   unset TWO_TASK
   unset local
else
   ${SRCDIR}/tpccacid_iso8.exe < iso8.in1 | tee iso8.out1 &
fi

#
# Sleep for 30 sec, start New Order T2 on same district and commit.
#
sleep 30
${SRCDIR}/tpcc_acid.exe < iso8.in2 | tee iso8.out2

#
# Verify the new order has been entered after 15 sec.
#
sleep 15
sqlplus tpcc/tpcc @${SQLDIR}/iso8v iso8.out4 10 5

#
# Wait for all transactions to complete.
#
wait

#
# Restore all new orders of warehouse 10 district 5
#
sqlplus tpcc/tpcc @${SQLDIR}/iso8r iso8.out5 10 5

#
# If this is OPS isolation test, restore stored procedure of new order.
#
if [ "$TT" != "" ]
then
   sqlplus tpcc/tpcc @${SPRDIR}/new.sql
fi

#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!
Input and Output for Deleting all New Orders from Warehouse 10 District 5
------------------------------------------------------------------------

!

cat iso8.out0 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of Delivery Transaction (T1) (on node 2 if OPS)
----------------------------------------------------------------

!

cat iso8.out1 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Trace output of Delivery Transaction (T1) (on node 2 if OPS)
------------------------------------------------------------

!

cat iso8.out3 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of New Order Transaction (T2) (on node 1 if OPS)
-----------------------------------------------------------------

!

cat iso8.out2 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Verification of the New Order just Entered
------------------------------------------

!

cat iso8.out4 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output for Restoring all New Orders of Warehouse 10 District 5
------------------------------------------------------------------------

!

cat iso8.out5 >> ${ODIR}/${OFILE}

if [ "$TT" != "" ]
then

cat >> ${ODIR}/${OFILE} <<!
```

```
Trace output for Transactions Executed on Node 1
------------------------------------------------

!

cat iso8.out6 >> ${ODIR}/${OFILE}

fi

rm -f iso8.in0 iso8.in1 iso8.in2 iso8.out0 iso8.out1 iso8.out2 iso8.out3 \
      iso8.in4 iso8.out4 iso8.in5 iso8.out5 iso8.out6
```

Iso8r.sql

```
rem
rem
============================================================+
rem        Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
rem                OPEN SYSTEMS PERFORMANCE GROUP                |
rem                   All Rights Reserved                        |
rem
============================================================+
rem  FILENAME
rem       iso8r.sql
rem  DESCRIPTION
rem       Performs TPC-C isolation test 8.
rem       Asks user to input values for ware_id, dist_id and output
rem       file.
rem ============================================================
rem
rem Usage:   sqlplus tpcc/tpcc @iso8r
rem

rem accept logfile     char     prompt  'Enter output file   =>  '

set verify off
spool &&1
set termout on
set echo on

rem accept ware_id     number  prompt  'Enter ware_id       =>  '
rem accept dist_id     number  prompt  'Enter dist_id       =>  '

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM
REM Restore all new orders.
REM

update nord set no_d_id = &&3
   where no_w_id = &&2 and no_d_id = 11;

commit;

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

spool off
exit
```

Iso8v.sql

```
rem
rem
============================================================+
rem        Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
rem                OPEN SYSTEMS PERFORMANCE GROUP                |
rem                   All Rights Reserved                        |
rem
============================================================+
rem  FILENAME
rem       iso8v.sql
rem  DESCRIPTION
rem       Performs TPC-C isolation test 8.
rem       Asks user to input values for ware_id, dist_id and output
rem       file.
rem ============================================================
rem
rem Usage:   sqlplus tpcc/tpcc @iso8v
rem

rem accept logfile     char     prompt  'Enter output file   =>  '

set verify off
spool &&1
set termout on
set echo on

rem accept ware_id     number  prompt  'Enter ware_id       =>  '
rem accept dist_id     number  prompt  'Enter dist_id       =>  '

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

REM
REM Select the new order just entered.
REM

 select no_w_id, no_d_id, no_o_id
   from nord
   where no_w_id = &&2 and no_d_id = &&3 and no_o_id =
   (select d_next_o_id - 1
      from dist
      where d_w_id = &&2 and d_id = &&3);

REM
REM Get timestamp.
REM

select to_char (sysdate, 'HH24:MI:SS') timestamp from dual;

spool off
exit
```

Iso9.sh

```
#!/bin/sh
#
# $Header: iso9.sh 01-jun-98.19:10:28 skareenh Exp $
#
# iso9.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      iso9.sh
#
#    DESCRIPTION
#                  Perform TPC-C isolation test 9.
#
#    NOTES
#                  iso9.sh [options]
#
#                  options:
#          -o <output file>      : name of output file
#          -h or -help           : print list of arguments
#          -net <TWO_TASK alias> : SQL*Net V2 connect string
#                                                                  alias for remote node
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

#
# Defaults
#

OFILE="iso9.v1"
TT=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
   case $1 in
      -o|-ofile)
         shift
         if [ "$1" != "" ]
         then
            OFILE=$1
            shift
         fi
         ;;
      -h|-help)
   echo "Options:"
   echo "  -o <output file>       : name of output file"
```

```
  echo "  -h or -help            : print list of arguments"
  echo "  -net <TWO_TASK alias>  : SQL*Net V2 connect string for remote node"
      exit 0
      ;;
    -net)
      shift
      if [ "$1" != "" ]
      then
        TT=$1
        shift
      fi
      ;;
    *)
      echo "Bad argument: $1"
      echo "Use -help option to see correct usage."
      exit 1
      ;;
  esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
  echo "Error: output file is not specified"
  echo "Use -help option to see correct usage."
  exit 1
fi

#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
   SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# stored procedures directory
#
SPRDIR=${BENCH_HOME}/tpc/tpcc/stored_proc

#
# Load stored procedure of order status that repeats reads & dumps messages.
#
# sqlplus tpcc/tpcc @${SQLDIR}/ord_iso9.sql

#
# If this is OPS isolation test, load stored procedures of
# new order and order status that dump what node they are executed on.
#
if [ "$TT" != "" ]
then
  echo "Running two task ..."
  sqlplus tpcc/tpcc @${SQLDIR}/new.sql
  sqlplus tpcc/tpcc @${SQLDIR}/ord.sql
fi

#
# txn = 3 (order status), w_id = 5, d_id = 1, c_id = 999, bylastname = 0,
# txn = 0 (quit)
#
cat > iso9.in1 <<!
3
5
1
999
0
0
!

#
# txn = 1 (new order), w_id = 5, d_id = 1, c_id = 999, o_ol_cnt = 5
# order_lines:      i_id        supply_w_id      quantity
#                   91          1                5
#                   92          1                5
#                   93          1                5
#                   94          1                5
#                   95          1                5
# txn = 0 (quit)
cat > iso9.in2 <<!
1
5
1
999
5
91
1
5
92
1
5
93
1
5
94
1
5
95
1
5
0
!

#
# Start plsql_mon to catch trace output from Order Status transaction T1.
# If this is OPS isolation test, start plsql_mon to catch trace output
# from all transactions.
#
if [ "$TT" != "" ]
then
  ${SRCDIR}/plsql_mon system/manager@${TT} 5 20 | tee iso9.out3 &
  ${SRCDIR}/plsql_mon system/manager 5 20 | tee iso9.out5 &
else
  ${SRCDIR}/plsql_mon system/manager 5 20 | tee iso9.out3 &
fi

#
# Do Order Status transaction T1 (on node 2 if OPS),
# find most recent order for that customer,
# sleep for 30 sec, then re-get most recent order for that customer.
#
if [ "$TT" != "" ]
then
  TWO_TASK=$TT
  export TWO_TASK
  set local=$TT
  ${SRCDIR}/tpccacid_iso9.exe  < iso9.in1 | tee iso9.out1 &
  unset TWO_TASK
  unset local
else
  ${SRCDIR}/tpccacid_iso9.exe < iso9.in1 | tee iso9.out1 &
fi

#
# Sleep for 15 sec, start New Order T2 on same customer and commit.
#
sleep 15
${SRCDIR}/tpcc_acid.exe < iso9.in2 | tee iso9.out2

#
# Do Order Status transaction T3 (on node 2 if OPS) after 30 sec.
#
sleep 30
if [ "$TT" != "" ]
then
  TWO_TASK=$TT
  export TWO_TASK
  set local=$TT
  ${SRCDIR}/tpcc_acid.exe < iso9.in1 | tee iso9.out4
  unset TWO_TASK
  unset local
else
  ${SRCDIR}/tpcc_acid.exe < iso9.in1 | tee iso9.out4
fi

#
# Wait for all transactions to complete.
#
wait

#
```

```
# If this is OPS isolation test, restore stored procedures of new order and
# order status.
#
if [ "$TT" != "" ]
then
   sqlplus tpcc/tpcc @${SPRDIR}/new.sql
   sqlplus tpcc/tpcc @${SPRDIR}/ord.sql
fi

#
# concatenate all output
#

cat > ${ODIR}/${OFILE} <<!
Input and Output of Order Status Transaction (T1) (on node 2 if OPS)
------------------------------------------------------------------------

!

cat iso9.out1 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Trace output of Order Status Transaction (T1) (on node 2 if OPS)
------------------------------------------------------------------

!

cat iso9.out3 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of New Order Transaction (T2) (on node 1 if OPS)
------------------------------------------------------------------

!

cat iso9.out2 >> ${ODIR}/${OFILE}

cat >> ${ODIR}/${OFILE} <<!


Input and Output of Order Status Transaction (T3) (on node 2 if OPS)
------------------------------------------------------------------

!

cat iso9.out4 >>  ${ODIR}/${OFILE}

if [ "$TT" != "" ]
then

cat >> ${ODIR}/${OFILE} <<!


Trace output for Transactions Executed on Node 1
-------------------------------------------------

!

cat iso9.out5 >> ${ODIR}/${OFILE}

fi

rm -f iso9.in1 iso9.in2 iso9.out1 iso9.out2 iso9.out3 iso9.out4 iso9.out5
```

Lcm.sh

```
#!/bin/sh
# echo the lcm of two numbers

if expr $2 \> $1 > /dev/null; then
  set $2 $1
# now $1 is guaranteed to be bigger
fi

lcm=$1
while expr \( \( $lcm % $1 \) + \( $lcm % $2 \) \) \> 0 > /dev/null; do
  lcm=`expr $lcm + $1`
done

echo $lcm
```

Listfiles.sh

```
#!/bin/sh
# Write out tablespace files to files.dat so the person in
# charge of the test can relink them.

#automatically export
set -a
tpcc_listfiles=t

rm $tpcc_bench/files.dat
cd $tpcc_bench
$tpcc_scripts/runscript.sh createts
echo "$tpcc_disks_location/system_001" $tpcc_system_size >> $tpcc_bench/files.dat
echo "$tpcc_disks_location/log_1" $tpcc_logfile_size >> $tpcc_bench/files.dat
echo "$tpcc_disks_location/log_2" $tpcc_logfile_size >> $tpcc_bench/files.dat
echo "$tpcc_disks_location/aux.df" $tpcc_sysaux_size >> $tpcc_bench/files.dat
echo "$tpcc_disks_location/roll01" $tpcc_undo_size >> $tpcc_bench/files.dat
echo "$tpcc_disks_location/sp_0" $tpcc_statspack_size >> $tpcc_bench/files.dat

tpcc_listfiles=f
#automatic export off
set +a

echo List of files written to $tpcc_bench/files.dat.  Stopping driver.

exit 99
```

Load_ordordl.sql

```
-- anonymous block for loading order/orderline

DECLARE
  order_idx      PLS_INTEGER;
  order_rows     PLS_INTEGER;
  ordl_rows      PLS_INTEGER;
  ordl_idx       PLS_INTEGER;
  ordl_idx_hi    PLS_INTEGER;
  local_idx      PLS_INTEGER;
BEGIN
  order_rows := :order_rows;
  ordl_rows  := :ordl_rows;
  order_idx  := 1;
  ordl_idx   := 1;

  WHILE (order_idx <= order_rows) LOOP

    INSERT INTO ordr (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(order_idx), :o_d_id(order_idx), :o_w_id(order_idx),
              :o_c_id(order_idx), SYSDATE, :o_carrier_id(order_idx),
              :o_ol_cnt(order_idx), 1);

    ordl_idx_hi := ordl_idx + :o_ol_cnt(order_idx) - 1;

    IF ( :o_id(order_idx) < 2101 ) THEN
      FORALL local_idx IN ordl_idx .. ordl_idx_hi
        INSERT INTO ordl (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(local_idx), :ol_d_id(local_idx),
                  :ol_w_id(local_idx), :ol_number(local_idx),
                  SYSDATE, :ol_i_id(local_idx),
                  :ol_supply_w_id(local_idx), 5, 0, :ol_dist_info(local_idx));
    ELSE
      FORALL local_idx IN ordl_idx .. ordl_idx_hi
        INSERT INTO ordl (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(local_idx), :ol_d_id(local_idx),
                  :ol_w_id(local_idx), :ol_number(local_idx),
                  to_date('01-Jan-1811'), :ol_i_id(local_idx),
                  :ol_supply_w_id(local_idx), 5,
                  :ol_amount(local_idx), :ol_dist_info(local_idx));
    END IF;
    ordl_idx := ordl_idx_hi + 1;
    order_idx := order_idx + 1;
  END LOOP;
END;
```

Loadcust.sh

```
#created automatically by /ORACLE/tpcc6700.100402.16K/scripts/evenload.sh Fri Oct 4
16:41:32 PDT 2002
rm loadcust*.log
cd $tpcc_bench
allprocs=
```

```
$tpcc_load -M 6700 -c  -b 1 -e 670 >> loadcust0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -c  -b 671 -e 1340 >> loadcust1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -c  -b 1341 -e 2010 >> loadcust2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -c  -b 2011 -e 2680 >> loadcust3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -c  -b 2681 -e 3350 >> loadcust4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -c  -b 3351 -e 4020 >> loadcust5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -c  -b 4021 -e 4690 >> loadcust6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -c  -b 4691 -e 5360 >> loadcust7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -c  -b 5361 -e 6030 >> loadcust8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -c  -b 6031 -e 6700 >> loadcust9.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

Loaddish.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -d > loaddist.log 2>&1
```

Loadfixordrordl.sh

```
#created automatically by /ORACLE/tpcc6700.100402.16K/scripts/evenload.sh Fri Oct 4
16:41:33 PDT 2002
rm loadfixordrordl*.log
cd $tpcc_bench
allprocs=
$tpcc_updateordrordl -M 6700 -o  -b 1 -e 670 >> loadfixordrordl0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_updateordrordl -M 6700 -o  -b 671 -e 1340 >> loadfixordrordl1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_updateordrordl -M 6700 -o  -b 1341 -e 2010 >> loadfixordrordl2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_updateordrordl -M 6700 -o  -b 2011 -e 2680 >> loadfixordrordl3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_updateordrordl -M 6700 -o  -b 2681 -e 3350 >> loadfixordrordl4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_updateordrordl -M 6700 -o  -b 3351 -e 4020 >> loadfixordrordl5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_updateordrordl -M 6700 -o  -b 4021 -e 4690 >> loadfixordrordl6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_updateordrordl -M 6700 -o  -b 4691 -e 5360 >> loadfixordrordl7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_updateordrordl -M 6700 -o  -b 5361 -e 6030 >> loadfixordrordl8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_updateordrordl -M 6700 -o  -b 6031 -e 6700 >> loadfixordrordl9.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

Loadhis.sh

```
#created automatically by /ORACLE/tpcc6700.100402.16K/scripts/evenload.sh Fri Oct 4
16:41:30 PDT 2002
rm loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 6700 -h  -b 1 -e 670 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -h  -b 671 -e 1340 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -h  -b 1341 -e 2010 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -h  -b 2011 -e 2680 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -h  -b 2681 -e 3350 >> loadhist4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -h  -b 3351 -e 4020 >> loadhist5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -h  -b 4021 -e 4690 >> loadhist6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -h  -b 4691 -e 5360 >> loadhist7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -h  -b 5361 -e 6030 >> loadhist8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -h  -b 6031 -e 6700 >> loadhist9.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loaditem.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -i > loaditem.log 2>&1
```

Loadnord.sh

```
#created automatically by /ORACLE/tpcc6700.100402.16K/scripts/evenload.sh Fri Oct 4
16:41:31 PDT 2002
rm loadnord*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 6700 -n  -b 1 -e 670 >> loadnord0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -n  -b 671 -e 1340 >> loadnord1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -n  -b 1341 -e 2010 >> loadnord2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -n  -b 2011 -e 2680 >> loadnord3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -n  -b 2681 -e 3350 >> loadnord4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -n  -b 3351 -e 4020 >> loadnord5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -n  -b 4021 -e 4690 >> loadnord6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -n  -b 4691 -e 5360 >> loadnord7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -n  -b 5361 -e 6030 >> loadnord8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -n  -b 6031 -e 6700 >> loadnord9.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

Loadordrordl.sh

```
#created automatically by /ORACLE/tpcc6700.100402.16K/scripts/evenload.sh Fri Oct 4
16:41:31 PDT 2002
rm loadordrordl*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy0.dat -b 1 -e 670 >> loadordrordl0.log 2>&1
&
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy1.dat -b 671 -e 1340 >> loadordrordl1.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy2.dat -b 1341 -e 2010 >> loadordrordl2.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy3.dat -b 2011 -e 2680 >> loadordrordl3.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy4.dat -b 2681 -e 3350 >> loadordrordl4.log
2>&1 &
allprocs="$allprocs ${!}"
```

```
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy5.dat -b 3351 -e 4020 >> loadordrordl5.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy6.dat -b 4021 -e 4690 >> loadordrordl6.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy7.dat -b 4691 -e 5360 >> loadordrordl7.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy8.dat -b 5361 -e 6030 >> loadordrordl8.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -o $tpcc_disks_location/dummy9.dat -b 6031 -e 6700 >> loadordrordl9.log
2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

Loadstok.sh

```
#created automatically by
/ORACLE/tpcc6700.100402.16K/scripts/evenload.sh Fri
Oct 4 16:41:33 PDT 2002
rm loadstok*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 6700 -S  -j 1 -k 10000 >> loadstok0.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -S  -j 10001 -k 20000 >>
loadstok1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -S  -j 20001 -k 30000 >>
loadstok2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -S  -j 30001 -k 40000 >>
loadstok3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -S  -j 40001 -k 50000 >>
loadstok4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -S  -j 50001 -k 60000 >>
loadstok5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -S  -j 60001 -k 70000 >>
loadstok6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -S  -j 70001 -k 80000 >>
loadstok7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -S  -j 80001 -k 90000 >>
loadstok8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 6700 -S  -j 90001 -k 100000 >>
loadstok9.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

Loadware.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -w > loadware.log 2>&1
```

Localoptions.sh

```
#LOCAL OPTION FILE- You must fill these in
# before the driver will work.


#oracle sid to use for the run
ORACLE_SID=tpcc3
#location of the database files (or links to raw partitions)
tpcc_disks_location=/project/oracle/dbs/tpcc3_disks

#locations of various files used in the generation scripts.
#(you can usually leave these alone.)
tpcc_sql_dir=${tpcc_bench}/scripts/sql
tpcc_log_dir=${tpcc_bench}/log
tpcc_genscripts_dir=${tpcc_bench}/scripts/generated


#Once you have filled all the options, comment
#out or delete this line.
# tpcc_no_options=t
```

New.sql

```
rem
rem
================================================================+
rem       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
rem                  OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                       All Rights Reserved                       |
rem
================================================================+
rem  FILENAME
rem       new.sql
rem  DESCRIPTION
rem       SQL script to create a stored package for new order
rem       transactions.
rem ============================================================
rem

CREATE OR REPLACE PACKAGE neworder
IS
  PROCEDURE enterorder
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id               INTEGER,
    ord_ol_cnt              INTEGER,
    ord_all_local           INTEGER,
    cust_discount       OUT NUMBER,
    cust_last           OUT VARCHAR2,
    cust_credit         OUT VARCHAR2,
    dist_tax            OUT NUMBER,
    ware_tax              OUT NUMBER,
    ord_id             IN OUT INTEGER,
    ord_entry_d        IN OUT VARCHAR2,
    retry              IN OUT INTEGER,
    cur_date                    IN                DATE
  );
END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY neworder
IS
  PROCEDURE enterorder
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id               INTEGER,
    ord_ol_cnt              INTEGER,
    ord_all_local           INTEGER,
```

```
    cust_discount       OUT NUMBER,
    cust_last           OUT VARCHAR2,
    cust_credit         OUT VARCHAR2,
    dist_tax            OUT NUMBER,
    ware_tax            OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d      IN OUT VARCHAR2,
    retry            IN OUT INTEGER,
    cur_date            IN              DATE
  )
  IS
    timestamp             DATE;
    dist_rowid      rowid;
    node_num                      varchar2(512);
    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
    SELECT substr(value,1,5)
      INTO node_num
      FROM v$parameter
      WHERE name = 'instance_number';

    plsql_mon_pack.print ('New Order started at ' ||
                  to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                  node_num);
    LOOP BEGIN
      SELECT dist.rowid, d_tax, d_next_o_id , w_tax
        INTO dist_rowid, dist_tax, ord_id, ware_tax
        FROM dist, ware
        WHERE d_id = dist_id AND d_w_id = ware_id
                  AND w_id = ware_id;
      UPDATE dist SET d_next_o_id = ord_id + 1
        WHERE rowid = dist_rowid;
      SELECT c_discount, c_last, c_credit
        INTO cust_discount, cust_last, cust_credit
        FROM cust
        WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id = ware_id;
      timestamp := cur_date;
      ord_entry_d := TO_CHAR(timestamp,'DD-MM-YYYY.HH24:MI:SS');
      INSERT INTO nord(no_o_id,no_d_id,no_w_id) VALUES
                (ord_id , dist_id, ware_id);
      INSERT INTO ordr(o_id,o_d_id,o_w_id,o_c_id,o_entry_d,o_carrier_id,
                o_ol_cnt, o_all_local)
      VALUES (ord_id , dist_id, ware_id, cust_id,
                        timestamp, 11, ord_ol_cnt, ord_all_local);
      EXIT;

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

quit;
```

Notneg.sh

```
#!/bin/sh
# echo the first and second params provided $2 is greater than -1.

if expr $2 \> -1 > /dev/null; then
  echo $1 $2
else
  echo
fi
```

Nt_audit_env

```
set BENCH_HOME=d:\\tpcc000111
set SRCHOME=%BENCH_HOME%
set ORACLE_SID=ops2
```

Options.sh

```
tpcc_os='unix'
tpcc_version='ttt'
tpcc_ldrive='1'
tpcc_scale='6700'
tpcc_np='1'
tpcc_cpu='5'
tpcc_memsize='256000'
tpcc_runlen='10'
tpcc_compress='f'
tpcc_overflow='t'

tpcc_cust_imp='cluster'
tpcc_cust_size='30064'
tpcc_cust_ext='calc'
tpcc_cust_nf='calc'
tpcc_cust_bs='auto'
tpcc_cust_used='-1'
tpcc_cust_free='0'
tpcc_cust_trans='3'
tpcc_cust_autospace='t'
tpcc_cust_flg='43'
tpcc_cust_fl='22'
tpcc_cust_rsize='auto'
tpcc_cust_hkey='auto'
tpcc_cust_hash='auto'
tpcc_cust_indices=3-2-1-
tpcc_cust_bpool=keep

tpcc_dist_imp='cluster'
tpcc_dist_size='20'
tpcc_dist_ext='calc'
tpcc_dist_nf='calc'
tpcc_dist_bs='auto'
tpcc_dist_used='-1'
tpcc_dist_free='-1'
tpcc_dist_trans='4'
tpcc_dist_autospace='t'
tpcc_dist_flg='43'
tpcc_dist_fl='22'
tpcc_dist_rsize='auto'
tpcc_dist_hkey='auto'
tpcc_dist_hash='auto'
tpcc_dist_indices=2-1-

tpcc_hist_imp='table'
tpcc_hist_size='1791'
tpcc_hist_ext='calc'
tpcc_hist_nf='calc'
tpcc_hist_bs='auto'
tpcc_hist_used='-1'
tpcc_hist_free='5'
tpcc_hist_trans='4'
tpcc_hist_autospace='t'
tpcc_hist_flg='43'
tpcc_hist_fl='22'
tpcc_hist_rsize='auto'
tpcc_hist_hkey='auto'
tpcc_hist_hash='auto'
tpcc_hist_indices=1-1-1-1-1-1-1-1-

tpcc_item_imp='cluster'
tpcc_item_size='14648'
tpcc_item_ext='calc'
tpcc_item_nf='calc'
tpcc_item_bs='auto'
tpcc_item_used='-1'
tpcc_item_free='0'
tpcc_item_trans='3'
tpcc_item_autospace='t'
tpcc_item_flg='43'
tpcc_item_fl='22'
tpcc_item_rsize='auto'
tpcc_item_hkey='auto'
tpcc_item_hash='auto'
tpcc_item_indices=1-

tpcc_nord_imp='queue'
tpcc_nord_size='178'
tpcc_nord_ext='calc'
tpcc_nord_nf='calc'
tpcc_nord_bs='auto'
tpcc_nord_used='-1'
tpcc_nord_free='5'
tpcc_nord_trans='4'
tpcc_nord_autospace='t'
tpcc_nord_flg='43'
tpcc_nord_fl='22'
tpcc_nord_rsize='auto'
tpcc_nord_hkey='auto'
tpcc_nord_hash='auto'
tpcc_nord_indices=1-2-3-

tpcc_ordl_imp='queue'
tpcc_ordl_size='21775'
tpcc_ordl_ext='calc'
tpcc_ordl_nf='calc'
```

```
tpcc_ordl_bs='16K'
tpcc_ordl_used='-1'
tpcc_ordl_free='5'
tpcc_ordl_trans='4'
tpcc_ordl_autospace='t'
tpcc_ordl_flg='43'
tpcc_ordl_fl='22'
tpcc_ordl_rsize='auto'
tpcc_ordl_hkey='auto'
tpcc_ordl_hash='auto'
tpcc_ordl_indices=1-2-3-4-

tpcc_ordr_imp='queue'
tpcc_ordr_size='1206'
tpcc_ordr_ext='calc'
tpcc_ordr_nf='calc'
tpcc_ordr_bs='16K'
tpcc_ordr_used='-1'
tpcc_ordr_free='5'
tpcc_ordr_trans='4'
tpcc_ordr_autospace='t'
tpcc_ordr_flg='43'
tpcc_ordr_fl='22'
tpcc_ordr_rsize='auto'
tpcc_ordr_hkey='auto'
tpcc_ordr_hash='auto'
tpcc_ordr_indices=2-3-1-

tpcc_stok_imp='cluster'
tpcc_stok_size='40977'
tpcc_stok_ext='calc'
tpcc_stok_nf='calc'
tpcc_stok_bs='auto'
tpcc_stok_used='-1'
tpcc_stok_free='0'
tpcc_stok_trans='3'
tpcc_stok_autospace='t'
tpcc_stok_flg='43'
tpcc_stok_fl='22'
tpcc_stok_rsize='auto'
tpcc_stok_hkey='auto'
tpcc_stok_hash='auto'
tpcc_stok_indices=1-2-
tpcc_stok_bpool=keep

tpcc_ware_imp='cluster'
tpcc_ware_size='2'
tpcc_ware_ext='calc'
tpcc_ware_nf='calc'
tpcc_ware_bs='auto'
tpcc_ware_used='-1'
tpcc_ware_free='-1'
tpcc_ware_trans='2'
tpcc_ware_autospace='t'
tpcc_ware_flg='43'
tpcc_ware_fl='22'
tpcc_ware_rsize='auto'
tpcc_ware_hkey='auto'
tpcc_ware_hash='auto'
tpcc_ware_indices=1-

tpcc_icust1_imp='index'
tpcc_icust1_size='736'
tpcc_icust1_ext='calc'
tpcc_icust1_nf='calc'
tpcc_icust1_bs='16K'
tpcc_icust1_used='-1'
tpcc_icust1_free='1'
tpcc_icust1_trans='3'
tpcc_icust1_autospace='t'
tpcc_icust1_flg='43'
tpcc_icust1_fl='22'
tpcc_icust1_rsize='auto'
tpcc_icust1_hkey='auto'
tpcc_icust1_hash='auto'
tpcc_icust1_indices=3-2-1-

tpcc_icust2_imp='index'
tpcc_icust2_size='1602'
tpcc_icust2_ext='calc'
tpcc_icust2_nf='calc'
tpcc_icust2_bs='16K'
tpcc_icust2_used='-1'
tpcc_icust2_free='1'
tpcc_icust2_trans='3'
tpcc_icust2_autospace='t'
tpcc_icust2_flg='43'
tpcc_icust2_fl='22'
tpcc_icust2_rsize='auto'
tpcc_icust2_hkey='auto'
tpcc_icust2_hash='auto'
tpcc_icust2_indices=6-3-2-7-1-

tpcc_idist_imp='index'
tpcc_idist_size='4'
tpcc_idist_ext='calc'
tpcc_idist_nf='calc'
tpcc_idist_bs='auto'
tpcc_idist_used='-1'
tpcc_idist_free='5'
tpcc_idist_trans='3'
tpcc_idist_autospace='t'
tpcc_idist_flg='43'
tpcc_idist_fl='22'
tpcc_idist_rsize='auto'
tpcc_idist_hkey='auto'
tpcc_idist_hash='auto'
tpcc_idist_indices=2-1-

tpcc_iitem_imp='index'
tpcc_iitem_size='2048'
tpcc_iitem_ext='calc'
tpcc_iitem_nf='calc'
tpcc_iitem_bs='auto'
tpcc_iitem_used='-1'
tpcc_iitem_free='5'
tpcc_iitem_trans='4'
tpcc_iitem_autospace='t'
tpcc_iitem_flg='43'
tpcc_iitem_fl='22'
tpcc_iitem_rsize='auto'
tpcc_iitem_hkey='auto'
tpcc_iitem_hash='auto'
tpcc_iitem_indices=1-

tpcc_inord_imp='none'
tpcc_inord_size='229'
tpcc_inord_ext='calc'
tpcc_inord_nf='calc'
tpcc_inord_bs='auto'
tpcc_inord_used='-1'
tpcc_inord_free='5'
tpcc_inord_trans='4'
tpcc_inord_autospace='t'
tpcc_inord_flg='43'
tpcc_inord_fl='22'
tpcc_inord_rsize='auto'
tpcc_inord_hkey='auto'
tpcc_inord_hash='auto'
tpcc_inord_indices=1-2-3-

tpcc_iordl_imp='none'
tpcc_iordl_size='8072'
tpcc_iordl_ext='calc'
tpcc_iordl_nf='calc'
tpcc_iordl_bs='auto'
tpcc_iordl_used='-1'
tpcc_iordl_free='5'
tpcc_iordl_trans='4'
tpcc_iordl_autospace='t'
tpcc_iordl_flg='43'
tpcc_iordl_fl='22'
tpcc_iordl_rsize='auto'
tpcc_iordl_hkey='auto'
tpcc_iordl_hash='auto'
tpcc_iordl_indices=1-2-3-4-

tpcc_iordr1_imp='none'
tpcc_iordr1_size='703'
tpcc_iordr1_ext='calc'
tpcc_iordr1_nf='calc'
tpcc_iordr1_bs='auto'
tpcc_iordr1_used='-1'
tpcc_iordr1_free='1'
tpcc_iordr1_trans='3'
tpcc_iordr1_autospace='t'
tpcc_iordr1_flg='43'
tpcc_iordr1_fl='22'
tpcc_iordr1_rsize='auto'
tpcc_iordr1_hkey='auto'
tpcc_iordr1_hash='auto'
tpcc_iordr1_indices=2-3-1-

tpcc_iordr2_imp='index'
tpcc_iordr2_size='1135'
tpcc_iordr2_ext='calc'
tpcc_iordr2_nf='calc'
tpcc_iordr2_bs='16K'
tpcc_iordr2_used='-1'
tpcc_iordr2_free='25'
tpcc_iordr2_trans='4'
tpcc_iordr2_autospace='t'
tpcc_iordr2_flg='43'
tpcc_iordr2_fl='22'
tpcc_iordr2_rsize='auto'
tpcc_iordr2_hkey='auto'
tpcc_iordr2_hash='auto'
tpcc_iordr2_indices=4-3-2-1-

tpcc_istok_imp='index'
```

```
tpcc_istok_size='2090'
tpcc_istok_ext='calc'
tpcc_istok_nf='calc'
tpcc_istok_bs='16K'
tpcc_istok_used='-1'
tpcc_istok_free='1'
tpcc_istok_trans='3'
tpcc_istok_autospace='t'
tpcc_istok_flg='43'
tpcc_istok_fl='22'
tpcc_istok_rsize='auto'
tpcc_istok_hkey='auto'
tpcc_istok_hash='auto'
tpcc_istok_indices=1-2-

tpcc_iware_imp='index'
tpcc_iware_size='1'
tpcc_iware_ext='calc'
tpcc_iware_nf='calc'
tpcc_iware_bs='auto'
tpcc_iware_used='-1'
tpcc_iware_free='1'
tpcc_iware_trans='3'
tpcc_iware_autospace='t'
tpcc_iware_flg='43'
tpcc_iware_fl='22'
tpcc_iware_rsize='auto'
tpcc_iware_hkey='auto'
tpcc_iware_hash='auto'
tpcc_iware_indices=1-

tpcc_temp_imp='temp'
tpcc_temp_size='16145'
tpcc_temp_ext='calc'
tpcc_temp_nf='calc'
tpcc_temp_bs='auto'
tpcc_temp_used='-1'
tpcc_temp_free='0'
tpcc_temp_trans='3'
tpcc_temp_autospace='t'
tpcc_temp_flg='43'
tpcc_temp_fl='22'
tpcc_temp_rsize='auto'
tpcc_temp_hkey='auto'
tpcc_temp_hash='auto'
tpcc_temp_indices=no
```

Ord.sql

```
rem
rem
================================================================+
rem       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
rem                  OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                     All Rights Reserved                         |
rem
================================================================+
rem  FILENAME
rem       ord.sql
rem  DESCRIPTION
rem       SQL script to create a stored package for order status
rem       transactions.
rem ================================================================
rem


CREATE OR REPLACE PACKAGE orderstatus
IS
  TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
  PROCEDURE getstatus_z
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id          IN OUT INTEGER,
    bylastname            INTEGER,
    cust_last        IN OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_balance        OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d         OUT VARCHAR2,
    ord_carrier_id      OUT INTEGER,
    ord_ol_cnt          OUT INTEGER
  );

  PROCEDURE getstatus_nz
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id          IN OUT INTEGER,
    bylastname            INTEGER,
    cust_last        IN OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_balance        OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d         OUT VARCHAR2,
    ord_carrier_id      OUT INTEGER,
    ord_ol_cnt          OUT INTEGER
  );
END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY orderstatus
IS
  PROCEDURE getstatus_z
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id          IN OUT INTEGER,
    bylastname            INTEGER,
    cust_last        IN OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_balance        OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d         OUT VARCHAR2,
    ord_carrier_id      OUT INTEGER,
    ord_ol_cnt          OUT INTEGER
  )
  IS
    cust_rowid            ROWID;
    ol                    BINARY_INTEGER;
    c_num                 BINARY_INTEGER;
    row_id                rowidarray;
    node_num              varchar2(512);
    not_serializable      EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock              EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old      EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
    CURSOR mo_cur IS
      SELECT o_id, to_char(o_entry_d, 'DD-MM-YYYY.HH24:MI:SS'),
          o_carrier_id, o_ol_cnt
      FROM ordr
      WHERE o_d_id = dist_id AND o_w_id = ware_id AND o_c_id = cust_id
      ORDER BY o_w_id, o_d_id, o_c_id, o_id DESC;
  BEGIN

   SELECT substr(value,1,5)
     INTO node_num
     FROM v$parameter
     WHERE name = 'instance_number';

   plsql_mon_pack.print ('Order Status started at ' ||
                    to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                    node_num);

   LOOP BEGIN

     SELECT c_balance, c_first, c_middle, c_last
       INTO cust_balance, cust_first, cust_middle, cust_last
       FROM cust
       WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id = ware_id;

     OPEN mo_cur;
     FETCH mo_cur INTO ord_id, ord_entry_d, ord_carrier_id, ord_ol_cnt;
     CLOSE mo_cur;

     EXIT;

     EXCEPTION
       WHEN not_serializable OR deadlock OR snapshot_too_old THEN
         ROLLBACK;
     END;
   END LOOP;

  END;

  PROCEDURE getstatus_nz
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id          IN OUT INTEGER,
    bylastname            INTEGER,
    cust_last        IN OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_balance        OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d         OUT VARCHAR2,
    ord_carrier_id      OUT INTEGER,
    ord_ol_cnt          OUT INTEGER
```

```
)
IS
  cust_rowid            ROWID;
  node_num              varchar2(512);
  ol                    BINARY_INTEGER;
  c_num                 BINARY_INTEGER;
  row_id                rowidarray;
  not_serializable      EXCEPTION;
  PRAGMA EXCEPTION_INIT(not_serializable,-8177);
  deadlock              EXCEPTION;
  PRAGMA EXCEPTION_INIT(deadlock,-60);
  snapshot_too_old      EXCEPTION;
  PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  CURSOR c_cur IS
    SELECT rowid
    FROM cust
    WHERE c_d_id = dist_id AND c_w_id = ware_id AND c_last = cust_last
    ORDER BY c_w_id, c_d_id, c_last, c_first;
  CURSOR mo_cur IS
    SELECT o_id, to_char(o_entry_d, 'DD-MM-YYYY.HH24:MI:SS'),
        o_carrier_id, o_ol_cnt
    FROM ordr
    WHERE o_d_id = dist_id AND o_w_id = ware_id AND o_c_id = cust_id
    ORDER BY o_w_id, o_d_id, o_c_id, o_id DESC;
BEGIN

  SELECT substr(value,1,5)
    INTO node_num
    FROM v$parameter
    WHERE name = 'instance_number';

  plsql_mon_pack.print ('Order Status started at ' ||
                to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                node_num);

  LOOP BEGIN

    c_num := 0;
    FOR c_id_rec IN c_cur LOOP
      c_num := c_num + 1;
      row_id(c_num) := c_id_rec.rowid;
    END LOOP;
    cust_rowid := row_id ((c_num + 1) / 2);

    SELECT c_id, c_balance, c_first, c_middle, c_last
      INTO cust_id, cust_balance, cust_first, cust_middle, cust_last
      FROM cust
      WHERE rowid = cust_rowid;

    OPEN mo_cur;
    FETCH mo_cur INTO ord_id, ord_entry_d, ord_carrier_id, ord_ol_cnt;
    CLOSE mo_cur;

    EXIT;

    EXCEPTION
      WHEN not_serializable OR deadlock OR snapshot_too_old THEN
        ROLLBACK;
    END;
  END LOOP;

 END;
END;
/
show errors;

quit;
```

P_build.ora

```
compatible = 10.0.0.0.0
db_name = tpcc
control_files = $tpcc_disks_location/control_001
parallel_max_servers = 40
recovery_parallelism = 40
db_files = 524
db_cache_size   = 5000M
db_8k_cache_size  = 500M
db_16k_cache_size  = 5000M
dml_locks = 500
log_buffer = 10485760
processes = 200
sessions = 200
transactions = 150
shared_pool_size = 350M
cursor_space_for_time = TRUE
db_block_size = 2048
undo_management = auto
undo_retention = 2
UNDO_TABLESPACE = undo_ts
_db_block_cache_protect = false
_lgwr_async_io   = FALSE
_db_file_noncontig_mblock_read_count = 1
db_block_checksum = false
db_block_checking = false
disk_asynch_io = true
#_enable_hash_overflow = true
```

P_create..ora

```
compatible = 10.0.0.0.0
db_name = tpcc
control_files = $tpcc_disks_location/control_001
db_block_size = 2048
db_cache_size = 10M
db_8k_cache_size = 10M
log_buffer = 1048576
db_16k_cache_size = 10M
undo_management = manual
```

P_run.ora

```
_db_writer_max_writes = 200
_db_writer_chunk_writes = 100
_two_pass = false
compatible = 10.0.0.0.0
db_name = tpcc
control_files = /project/oracle/dbs/tpcc3_disks/control_001
parallel_max_servers = 160
recovery_parallelism = 40
db_files = 150
db_cache_size   = 1250M
db_8k_cache_size  = 300M
db_16k_cache_size  = 7600M
db_recycle_cache_size = 20M
db_keep_cache_size = 34500M
dml_locks = 500
db_writer_processes = 1
cpu_count = 4
log_buffer = 10485760
processes = 200
sessions = 200
transactions = 180
shared_pool_size = 500M
cursor_space_for_time = TRUE
db_block_size = 2048
undo_management = auto
undo_retention = 1
log_checkpoint_interval = 0
log_checkpoint_timeout  = 0
log_checkpoints_to_alert = TRUE
UNDO_TABLESPACE = undo_ts
timed_statistics = FALSE
statistics_level = basic
disk_asynch_io = true
db_block_checksum = false
db_block_checking = false
lock_sga = true
pga_aggregate_target = 0
plsql_compiler_flags = optimize
```

Pay.sql

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

```
rem  DESCRIPTION
rem      SQL script to create a stored procedure for payment
rem      transactions.
rem ============================================================
rem

CREATE OR REPLACE PACKAGE payment
IS
  PROCEDURE dopayment_z
  (
    ware_id                INTEGER,
    dist_id                INTEGER,
    cust_w_id              INTEGER,
    cust_d_id              INTEGER,
    cust_id             IN OUT INTEGER,
    bylastname             INTEGER,
    hist_amount            INTEGER,
    cust_last           IN OUT VARCHAR2,
    ware_street_1          OUT VARCHAR2,
    ware_street_2          OUT VARCHAR2,
    ware_city              OUT VARCHAR2,
    ware_state             OUT VARCHAR2,
    ware_zip               OUT VARCHAR2,
    dist_street_1          OUT VARCHAR2,
    dist_street_2          OUT VARCHAR2,
    dist_city              OUT VARCHAR2,
    dist_state             OUT VARCHAR2,
    dist_zip               OUT VARCHAR2,
    cust_first             OUT VARCHAR2,
    cust_middle            OUT VARCHAR2,
    cust_street_1          OUT VARCHAR2,
    cust_street_2          OUT VARCHAR2,
    cust_city              OUT VARCHAR2,
    cust_state             OUT VARCHAR2,
    cust_zip               OUT VARCHAR2,
    cust_phone             OUT VARCHAR2,
    cust_since             OUT VARCHAR2,
    cust_credit         IN OUT VARCHAR2,
    cust_credit_lim        OUT NUMBER,
    cust_discount          OUT NUMBER,
    cust_balance        IN OUT NUMBER,
    cust_data              OUT VARCHAR2,
    hist_date              OUT VARCHAR2,
    retry               IN OUT INTEGER,
    cur_date                   IN DATE
  );

PROCEDURE dopayment_nz
  (
    ware_id                INTEGER,
    dist_id                INTEGER,
    cust_w_id              INTEGER,
    cust_d_id              INTEGER,
    cust_id             IN OUT INTEGER,
    bylastname             INTEGER,
    hist_amount            INTEGER,
    cust_last           IN OUT VARCHAR2,
    ware_street_1          OUT VARCHAR2,
    ware_street_2          OUT VARCHAR2,
    ware_city              OUT VARCHAR2,
    ware_state             OUT VARCHAR2,
    ware_zip               OUT VARCHAR2,
    dist_street_1          OUT VARCHAR2,
    dist_street_2          OUT VARCHAR2,
    dist_city              OUT VARCHAR2,
    dist_state             OUT VARCHAR2,
    dist_zip               OUT VARCHAR2,
    cust_first             OUT VARCHAR2,
    cust_middle            OUT VARCHAR2,
    cust_street_1          OUT VARCHAR2,
    cust_street_2          OUT VARCHAR2,
    cust_city              OUT VARCHAR2,
    cust_state             OUT VARCHAR2,
    cust_zip               OUT VARCHAR2,
    cust_phone             OUT VARCHAR2,
    cust_since             OUT VARCHAR2,
    cust_credit         IN OUT VARCHAR2,
    cust_credit_lim        OUT NUMBER,
    cust_discount          OUT NUMBER,
    cust_balance        IN OUT NUMBER,
    cust_data              OUT VARCHAR2,
    hist_date              OUT VARCHAR2,
    retry               IN OUT INTEGER,
    cur_date                   IN DATE
  );
END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY payment
IS
  PROCEDURE dopayment_z
  (
    ware_id                INTEGER,
    dist_id                INTEGER,
    cust_w_id              INTEGER,
    cust_d_id              INTEGER,
    cust_id             IN OUT INTEGER,
    bylastname             INTEGER,
    hist_amount            INTEGER,
    cust_last           IN OUT VARCHAR2,
    ware_street_1          OUT VARCHAR2,
    ware_street_2          OUT VARCHAR2,
    ware_city              OUT VARCHAR2,
    ware_state             OUT VARCHAR2,
    ware_zip               OUT VARCHAR2,
    dist_street_1          OUT VARCHAR2,
    dist_street_2          OUT VARCHAR2,
    dist_city              OUT VARCHAR2,
    dist_state             OUT VARCHAR2,
    dist_zip               OUT VARCHAR2,
    cust_first             OUT VARCHAR2,
    cust_middle            OUT VARCHAR2,
    cust_street_1          OUT VARCHAR2,
    cust_street_2          OUT VARCHAR2,
    cust_city              OUT VARCHAR2,
    cust_state             OUT VARCHAR2,
    cust_zip               OUT VARCHAR2,
    cust_phone             OUT VARCHAR2,
    cust_since             OUT VARCHAR2,
    cust_credit         IN OUT VARCHAR2,
    cust_credit_lim        OUT NUMBER,
    cust_discount          OUT NUMBER,
    cust_balance        IN OUT NUMBER,
    cust_data              OUT VARCHAR2,
    hist_date              OUT VARCHAR2,
    retry               IN OUT INTEGER,
    cur_date                   IN DATE
  )
  IS
    TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
    cust_rowid             ROWID;
    ware_rowid             ROWID;
    dist_ytd                       NUMBER(12);
    dist_name              VARCHAR2(11);
    ware_ytd                       NUMBER(12);
    ware_name              VARCHAR2(11);
    history_date           DATE;
    c_num                  BINARY_INTEGER;
    row_id                 rowidarray;
    cust_payments                  PLS_INTEGER;
    cust_ytd                       NUMBER(12);
    cust_data_temp                 VARCHAR2(500);
    node_num                       VARCHAR2(512);
    not_serializable       EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock               EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old       EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
    CURSOR c_cur IS
                SELECT rowid
      FROM cust
      WHERE c_d_id = cust_d_id AND c_w_id = cust_w_id AND c_last = cust_last
      ORDER BY c_w_id, c_d_id, c_last, c_first;
  BEGIN
    SELECT substr(value,1,5)
      INTO node_num
      FROM v$parameter
      WHERE name = 'instance_number';

    plsql_mon_pack.print ('Payment started at ' ||
                  to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                  node_num);
    LOOP BEGIN
      SELECT rowid, c_first, c_middle, c_last, c_street_1, c_street_2,
          c_city, c_state, c_zip, c_phone,
          to_char (c_since, 'DD-MM-YYYY'), c_credit, c_credit_lim,
          c_discount, c_balance - hist_amount, c_payment_cnt ,
                              c_ytd_payment + hist_amount, decode (c_credit, 'BC', c_data, ' ')
        INTO cust_rowid, cust_first, cust_middle, cust_last,
          cust_street_1, cust_street_2, cust_city, cust_state,
          cust_zip, cust_phone, cust_since, cust_credit,
          cust_credit_lim, cust_discount, cust_balance, cust_payments,
                               cust_ytd, cust_data_temp
      FROM cust
      WHERE c_id = cust_id AND c_d_id = cust_d_id AND
            c_w_id = cust_w_id;
    cust_payments := cust_payments + 1;
    IF cust_credit = 'BC' THEN
                  cust_data_temp := substr ((to_char (cust_id) || ' ' ||
                        to_char (cust_d_id) || ' ' ||
                        to_char (cust_w_id) || ' ' ||
                        to_char (dist_id) || ' ' ||
                        to_char (ware_id) || ' ' ||
                        to_char (hist_amount, '9999.99') || ' | ')
                        || cust_data_temp, 1, 500);

      UPDATE cust
        SET c_balance = cust_balance,
```

```
            c_ytd_payment = cust_ytd,
            c_payment_cnt = cust_payments,
                                    c_data = cust_data_temp
        WHERE rowid = cust_rowid;

                    cust_data := substr(cust_data_temp,1, 200);

    ELSE

      UPDATE cust
        SET c_balance = cust_balance,
            c_ytd_payment = cust_ytd,
            c_payment_cnt = cust_payments
        WHERE rowid = cust_rowid;

                    cust_data := cust_data_temp;
    END IF;

    SELECT dist.rowid, d_name, d_street_1, d_street_2, d_city,
                        d_state, d_zip, d_ytd + hist_amount,
          ware.rowid, w_name, w_street_1, w_street_2, w_city,
           w_state, w_zip, w_ytd + hist_amount
      INTO cust_rowid, dist_name, dist_street_1, dist_street_2, dist_city,
           dist_state, dist_zip, dist_ytd,
           ware_rowid, ware_name, ware_street_1, ware_street_2, ware_city,
           ware_state, ware_zip, ware_ytd
      FROM dist, ware
      WHERE d_id = dist_id
       AND d_w_id = ware_id
       AND w_id = ware_id;

    UPDATE dist
      SET d_ytd = dist_ytd
      WHERE rowid = cust_rowid;

    UPDATE ware
      SET w_ytd = ware_ytd
      WHERE rowid = ware_rowid;

    history_date := cur_date;

    INSERT INTO hist(h_c_id,h_c_d_id, h_c_w_id,h_d_id,h_w_id,h_date,
         h_amount, h_data) VALUES
       (cust_id, cust_d_id, cust_w_id, dist_id, ware_id, history_date,
        hist_amount, ware_name || '    ' || dist_name);
    COMMIT;
    hist_date := to_char (history_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;


PROCEDURE dopayment_nz
(
  ware_id                INTEGER,
  dist_id                INTEGER,
  cust_w_id              INTEGER,
  cust_d_id              INTEGER,
  cust_id             IN OUT INTEGER,
  bylastname             INTEGER,
  hist_amount            INTEGER,
  cust_last           IN OUT VARCHAR2,
  ware_street_1          OUT VARCHAR2,
  ware_street_2          OUT VARCHAR2,
  ware_city              OUT VARCHAR2,
  ware_state             OUT VARCHAR2,
  ware_zip               OUT VARCHAR2,
  dist_street_1          OUT VARCHAR2,
  dist_street_2          OUT VARCHAR2,
  dist_city              OUT VARCHAR2,
  dist_state             OUT VARCHAR2,
  dist_zip               OUT VARCHAR2,
  cust_first             OUT VARCHAR2,
  cust_middle            OUT VARCHAR2,
  cust_street_1          OUT VARCHAR2,
  cust_street_2          OUT VARCHAR2,
  cust_city              OUT VARCHAR2,
  cust_state             OUT VARCHAR2,
  cust_zip               OUT VARCHAR2,
  cust_phone             OUT VARCHAR2,
  cust_since             OUT VARCHAR2,
  cust_credit         IN OUT VARCHAR2,
  cust_credit_lim        OUT NUMBER,
  cust_discount          OUT NUMBER,
  cust_balance        IN OUT NUMBER,
  cust_data              OUT VARCHAR2,
  hist_date              OUT VARCHAR2,
  retry               IN OUT INTEGER,
  cur_date                   IN DATE
)
IS
  TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
  cust_rowid             ROWID;
  ware_rowid             ROWID;
  dist_ytd                      NUMBER(12);
  dist_name              VARCHAR2(11);
  ware_ytd                      NUMBER(12);
  ware_name              VARCHAR2(11);
  history_date           DATE;
  c_num                  BINARY_INTEGER;
  row_id                 rowidarray;
  cust_payments                 PLS_INTEGER;
  cust_ytd                      NUMBER(12);
  cust_data_temp                VARCHAR2(500);
  node_num                      VARCHAR2(512);
  not_serializable       EXCEPTION;
  PRAGMA EXCEPTION_INIT(not_serializable,-8177);
  deadlock               EXCEPTION;
  PRAGMA EXCEPTION_INIT(deadlock,-60);
  snapshot_too_old       EXCEPTION;
  PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  CURSOR c_cur IS
    SELECT rowid
    FROM cust
    WHERE c_d_id = cust_d_id AND c_w_id = cust_w_id AND c_last = cust_last
    ORDER BY c_w_id, c_d_id, c_last, c_first;
BEGIN
  SELECT substr(value,1,5)
    INTO node_num
    FROM v$parameter
    WHERE name = 'instance_number';

  plsql_mon_pack.print ('Payment started at ' ||
                  to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                  node_num);
  LOOP BEGIN

    c_num := 0;
    FOR c_id_rec IN c_cur LOOP
      c_num := c_num + 1;
      row_id(c_num) := c_id_rec.rowid;
    END LOOP;
    cust_rowid := row_id ((c_num + 1) / 2);    -- use row_id.count ?

    SELECT c_id, c_first, c_middle, c_last, c_street_1, c_street_2,
          c_city, c_state, c_zip, c_phone,
          to_char (c_since, 'DD-MM-YYYY'), c_credit, c_credit_lim,
          c_discount, c_balance - hist_amount, c_payment_cnt,
                              c_ytd_payment + hist_amount, decode (c_credit, 'BC', c_data, ' ')
      INTO cust_id, cust_first, cust_middle, cust_last,
           cust_street_1, cust_street_2, cust_city, cust_state,
           cust_zip, cust_phone, cust_since, cust_credit,
           cust_credit_lim, cust_discount, cust_balance, cust_payments,
                                cust_ytd, cust_data_temp
      FROM cust
      WHERE rowid = cust_rowid;
    cust_payments := cust_payments + 1;
    IF cust_credit = 'BC' THEN
                  cust_data_temp := substr ((to_char (cust_id) || ' ' ||
                        to_char (cust_d_id) || ' ' ||
                        to_char (cust_w_id) || ' ' ||
                        to_char (dist_id) || ' ' ||
                        to_char (ware_id) || ' ' ||
                        to_char (hist_amount/100, '9999.99') || ' | ')
                       || cust_data_temp, 1, 500);

      UPDATE cust
        SET c_balance = cust_balance,
           c_ytd_payment = cust_ytd,
           c_payment_cnt = cust_payments,
                                c_data = cust_data_temp
        WHERE rowid = cust_rowid;

                  cust_data := substr(cust_data_temp,1, 200);

    ELSE

      UPDATE cust
        SET c_balance = cust_balance,
           c_ytd_payment = cust_ytd,
           c_payment_cnt = cust_payments
        WHERE rowid = cust_rowid;

                  cust_data := cust_data_temp;

    END IF;

    SELECT dist.rowid, d_name, d_street_1, d_street_2, d_city,
                      d_state, d_zip, d_ytd + hist_amount,
          ware.rowid, w_name, w_street_1, w_street_2, w_city,
           w_state, w_zip, w_ytd + hist_amount
      INTO cust_rowid, dist_name, dist_street_1, dist_street_2, dist_city,
           dist_state, dist_zip, dist_ytd,
```

```
          ware_rowid, ware_name, ware_street_1, ware_street_2, ware_city,
          ware_state, ware_zip, ware_ytd
        FROM dist, ware
        WHERE d_id = dist_id
         AND d_w_id = ware_id
         AND w_id = ware_id;

      UPDATE dist
        SET d_ytd = dist_ytd
        WHERE rowid = cust_rowid;

      UPDATE ware
        SET w_ytd = ware_ytd
        WHERE rowid = ware_rowid;

      history_date := cur_date;

      INSERT INTO hist VALUES
        (cust_id, cust_d_id, cust_w_id, dist_id, ware_id, history_date,
         hist_amount, ware_name || '    ' || dist_name);
      COMMIT;
      hist_date := to_char (history_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;
END;
/
show errors;

quit;
```

Paynz.sql

```
DECLARE /* paynz */
    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 ware
          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 inittpcc.ware_name, :w_street_1, :w_street_2, :w_city,
               :w_state, :w_zip;

        UPDATE  cust
          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 inittpcc.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;


      IF :c_credit = 'BC' THEN
        UPDATE cust
                    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 = inittpcc.cust_rowid
      RETURNING substr(c_data,1, 200)
          INTO :c_data;

      END IF;

        UPDATE dist
          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 inittpcc.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 hist  (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);
      EXIT;

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

    END LOOP;
  END;
```

Paynz_abort.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);
    CURSOR c_cur IS
      SELECT rowid
      FROM cust
      WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last = :c_last
      ORDER BY c_w_id, c_d_id, c_last, c_first;
  BEGIN
    LOOP BEGIN
      UPDATE ware
        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 inittpcc.ware_name,
               :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


      iniitpcc.c_num := 0;
      FOR c_id_rec IN c_cur LOOP
        inittpcc.c_num := initpay.c_num + 1;
        inittpcc.row_id(initpay.c_num) := c_id_rec.rowid;
      END LOOP;
      inittpcc.cust_rowid := initpay.row_id ((initpay.c_num + 1) / 2);

      UPDATE cust
       SET c_balance = c_balance - :h_amount,
          c_ytd_payment = c_ytd_payment+ :h_amount,
          c_payment_cnt = c_payment_cnt+1
      WHERE rowid = inittpcc.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 cust
          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 = inittpcc.cust_rowid
          RETURNING substr(c_data,1, 200)
          INTO :c_data;

      END IF;

      UPDATE dist
        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 inittpcc.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 hist (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);

      ROLLBACK;
      EXIT;

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

    END LOOP;
  END;
```

Paynz_cmiit.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 ware
            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 inittpcc.ware_name, :w_street_1, :w_street_2, :w_city,
             :w_state, :w_zip;

         UPDATE  cust
           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 inittpcc.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 cust
                  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 = inittpcc.cust_rowid
        RETURNING substr(c_data,1, 200)
          INTO :c_data;

      END IF;

        UPDATE dist
          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 inittpcc.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 hist  (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);
--      COMMIT;
--      :h_date := to_char (:cr_date, 'DD-MM-YYYY.HH24:MI:SS');
      EXIT;

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

    END LOOP;
  END;
```

Paynz_std.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 ware
            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 inittpcc.ware_name, :w_street_1, :w_street_2, :w_city,
             :w_state, :w_zip;

         UPDATE  cust
           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 inittpcc.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 cust
                  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 = inittpcc.cust_rowid
        RETURNING substr(c_data,1, 200)
          INTO :c_data;

      END IF;

        UPDATE dist
          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 inittpcc.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 hist  (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);
--        COMMIT;
--        :h_date := to_char (:cr_date, 'DD-MM-YYYY.HH24:MI:SS');
        EXIT;

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

    END LOOP;
  END;
```

Payz.sql

```
DECLARE /* payz */
    not_serializable        EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock                EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old        EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  BEGIN
    LOOP BEGIN
      UPDATE ware
        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 inittpcc.ware_name,
            :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


      SELECT rowid
      BULK COLLECT INTO inittpcc.row_id
      FROM cust
      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;

     inittpcc.c_num := sql%rowcount;
     inittpcc.cust_rowid := inittpcc.row_id((inittpcc.c_num) / 2);

      UPDATE cust
       SET c_balance = c_balance - :h_amount,
          c_ytd_payment = c_ytd_payment+ :h_amount,
          c_payment_cnt = c_payment_cnt+1
      WHERE rowid = inittpcc.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 cust
          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 = inittpcc.cust_rowid
          RETURNING substr(c_data,1, 200)
          INTO :c_data;

      END IF;

      UPDATE dist
        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 inittpcc.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 hist (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);

      EXIT;

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

    END LOOP;
  END;
```

Payz_abort.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);
    CURSOR c_cur IS
      SELECT rowid
      FROM cust
      WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last = :c_last
      ORDER BY c_w_id, c_d_id, c_last, c_first;
  BEGIN
    LOOP BEGIN
      UPDATE ware
        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 inittpcc.ware_name,
            :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


      iniitpcc.c_num := 0;
      FOR c_id_rec IN c_cur LOOP
        inittpcc.c_num := initpay.c_num + 1;
        inittpcc.row_id(initpay.c_num) := c_id_rec.rowid;
      END LOOP;
      inittpcc.cust_rowid := initpay.row_id ((initpay.c_num + 1) / 2);

      UPDATE cust
       SET c_balance = c_balance - :h_amount,
          c_ytd_payment = c_ytd_payment+ :h_amount,
          c_payment_cnt = c_payment_cnt+1
      WHERE rowid = inittpcc.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 cust
          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 = inittpcc.cust_rowid
          RETURNING substr(c_data,1, 200)
          INTO :c_data;

      END IF;

      UPDATE dist
        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 inittpcc.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 hist (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);

      ROLLBACK;
      EXIT;

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

    END LOOP;
  END;
```

Payz_comnmit.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);
    CURSOR c_cur IS
      SELECT rowid
      FROM cust
      WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last = :c_last
      ORDER BY c_w_id, c_d_id, c_last, c_first;
  BEGIN
    LOOP BEGIN
      UPDATE ware
        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 inittpcc.ware_name,
            :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


      iniitpcc.c_num := 0;
      FOR c_id_rec IN c_cur LOOP
        inittpcc.c_num := initpay.c_num + 1;
        inittpcc.row_id(initpay.c_num) := c_id_rec.rowid;
      END LOOP;
      inittpcc.cust_rowid := initpay.row_id ((initpay.c_num + 1) / 2);

      UPDATE cust
       SET c_balance = c_balance - :h_amount,
          c_ytd_payment = c_ytd_payment+ :h_amount,
          c_payment_cnt = c_payment_cnt+1
      WHERE rowid = inittpcc.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 cust
          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 = inittpcc.cust_rowid
          RETURNING substr(c_data,1, 200)
          INTO :c_data;

      END IF;

      UPDATE dist
        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 inittpcc.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 hist (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);

        EXIT;

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

      END LOOP;
    END;
```

Payz_std.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 ware
        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 inittpcc.ware_name,
            :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


      SELECT rowid
      BULK COLLECT INTO inittpcc.row_id
      FROM cust
      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;

     inittpcc.c_num := sql%rowcount;
     inittpcc.cust_rowid := inittpcc.row_id((inittpcc.c_num) / 2);

      UPDATE cust
       SET c_balance = c_balance - :h_amount,
          c_ytd_payment = c_ytd_payment+ :h_amount,
          c_payment_cnt = c_payment_cnt+1
      WHERE rowid = inittpcc.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 cust
          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 = inittpcc.cust_rowid
          RETURNING substr(c_data,1, 200)
          INTO :c_data;

      END IF;

      UPDATE dist
        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 inittpcc.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 hist (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, inittpcc.ware_name || '    ' || inittpcc.dist_name);

      EXIT;

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

    END LOOP;
  END;
```

Plsql_mon.sql

```
rem
rem
================================================================+
rem        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA       |
rem                OPEN SYSTEMS PERFORMANCE GROUP                |
rem                    All Rights Reserved                       |
rem
================================================================+
rem  FILENAME
rem       plsql_mon.sql
rem  DESCRIPTION
rem       SQL script to create a stored package for PL/SQL stored
rem       procedures to dump messages.
rem ================================================================
rem
rem Usage:   sqlplus tpcc/tpcc @plsql_mon
rem

connect tpcc/tpcc;
set echo on;
CREATE OR REPLACE PACKAGE plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  );
END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  )
  IS
     s          NUMBER;
  BEGIN
    dbms_pipe.pack_message (info);
    s := dbms_pipe.send_message ('plsql_mon');
    IF (s <> 0) THEN
      raise_application_error (-20000, 'Error:' || to_char(s) ||
                        ' sending on pipe');
    END IF;
  END;
END;
/
show errors;

set echo off;
```

Remort.sql

```
set pagesize 1000
set termout on
set echo on
set timing on

Rem Total configured warehouses
select count(*) from ware;

Rem Check that the unused warehouses have not been involved
select w_id, w_ytd from ware where w_id > &&1;

Rem Get a sample of 1000 orders to prove that the supplied warehouse is not unused
select /*+ parallel(ordl, 40) */ ol_supply_w_id, ol_w_id, ol_d_id, ol_o_id from ordl
where ol_supply_w_id <> ol_w_id and ol_o_id > 3000 and rownum <= 1000;

exit;
```

Require_vars.sh

```
#!/bin/sh
# make sure each environment variable argument is defined.

for var in $* ; do
  if test -z "`eval echo \\$$var`"; then
    echo Missing variable: $var - exiting.
    exit 0
  fi
done
exit 1
```

Runchk.sh

```
#!/bin/sh
#
# $Header: runchk.sh 01-jun-98.19:11:42 skareenh Exp $
#
# runchk.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      runchk.sh
#
#    DESCRIPTION
#                  Does spot checking after a TPC-C run.
#
#    NOTES
#                  runchk.sh [output_file]
#
#                  options:
#          -o <output file>         : name of output file
#          -h or -help              : print list of arguments
#          -m <max w_id>            : maximum warehouse id in database
#          -net "list of aliases"   : list of SQL*Net V2 connect string aliases
#          -wids "list of w_id's"   : list of warehouse ids for sampling
#          -bwids "list of w_id's"  : list of beginning ids of warehouse ranges
#          -ewids "list of w_id's"  : list of ending ids of warehouse ranges
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

#
# Defaults
#

OFILE="runchk.v1"
MULT=
SNET=
WIDS=
BWIDS=
EWIDS=

#
# Parse arguments
#

while [ "$#" != "0" ]
do
  case $1 in
    -o|-ofile)
      shift
      if [ "$1" != "" ]
      then
        OFILE=$1
        shift
      fi
      ;;
    -h|-help)
  echo "Options:"
  echo "  -o <output file>       : name of output file"
  echo "  -h or -help            : print list of arguments"
  echo "  -m <max w_id>          : maximum warehouse id in database"
  echo "  -net \"list of aliases\" : list of SQL*Net V2 connect string aliases"
  echo "  -wids \"list of w_id's\" : list of warehouse ids for sampling"
  echo "  -bwids \"list of w_id's\": list of beginning ids of warehouse ranges"
  echo "  -ewids \"list of w_id's\": list of ending ids of warehouse ranges"
      exit 0
      ;;
    -m|-mult)
      shift
      if [ "$1" != "" ]
      then
        MULT=$1
        shift
```

```
        fi
        ;;
      -net)
        shift
        if [ "$1" != "" ]
        then
          SNET=$1
          shift
        fi
        ;;
      -wids)
        shift
        if [ "$1" != "" ]
        then
          WIDS=$1
          shift
        fi
        ;;
      -bwids)
        shift
        if [ "$1" != "" ]
        then
          BWIDS=$1
          shift
        fi
        ;;
      -ewids)
        shift
        if [ "$1" != "" ]
        then
          EWIDS=$1
          shift
        fi
        ;;
      *)
        echo "Bad argument: $1"
        echo "Use -help option to see correct usage."
        exit 1
        ;;
    esac
done

#
# Check arguments
#

if [ "$OFILE" = "" ]
then
  echo "Error: output file is not specified"
  echo "Use -help option to see correct usage."
  exit 1
fi

NHOSTS=0
for HOST in $SNET
do
  NHOSTS=`expr $NHOSTS + 1`
done

NWIDS=0
for WID in $WIDS
do
  NWIDS=`expr $NWIDS + 1`
done

NBWIDS=0
for BWID in $BWIDS
do
  NBWIDS=`expr $NBWIDS + 1`
done

NEWIDS=0
for EWID in $EWIDS
do
  NEWIDS=`expr $NEWIDS + 1`
done

if [ $NBWIDS != $NEWIDS ]
then
  echo "Error: # of beginning w_id's != # of ending w_id's"
  echo "Use -help option to see correct usage."
  exit 1
fi

if [ $NHOSTS != 0 ]
then
  if [ $NWIDS != 0 -a $NHOSTS != $NWIDS ]
  then
    echo "Error: # of SQL*Net V2 aliases != # of samples"
    echo "Use -help option to see correct usage."
    exit 1
  fi
  if [ $NBWIDS != 0 -a $NHOSTS != $NBWIDS ]
  then
    echo "Error: # of SQL*Net V2 aliases != # of w_id ranges"
    echo "Use -help option to see correct usage."
    exit 1
  fi
else
  if [ $NWIDS != 0 -a $NBWIDS != 0 -a $NWIDS != $NBWIDS ]
  then
    echo "Error: # of samples != # of w_id ranges"
    echo "Use -help option to see correct usage."
    exit 1
  fi
fi

#
# By default, if no SQL*Net V2 string is specified, all checks are
# run on the local node.
#

#
# By default, if no w_id ranges are specified, then ranges will be
# generated automatically based on the total number of warehouse and
# the number of SQL*Net V2 strings or the number of w_id samples.
#
if [ $NBWIDS = 0 ]
then
  if [ "$MULT" = "" ]
  then
    echo "Error: max warehouse id in this database is not specified"
    echo "Use -help option to see correct usage."
    exit 1
  fi

  if [ $NHOSTS != 0 ]
  then
    NSAMPS=$NHOSTS
  elif [ $NWIDS != 0 ]
  then
    NSAMPS=$NWIDS
  elif [ $MULT -ge 4 ]
  then
    NSAMPS=4
  else
    NSAMPS=$MULT
  fi

  EVENW=`expr $MULT % $NSAMPS`
  if [ $EVENW != 0 ]
  then
    echo "Error: cannot evenly divide # of warehouses by # of nodes/samples"
    echo "       so cannot generate w_id ranges automatically"
    echo "Use -help option to see correct usage."
    exit 1
  fi

  WPERN=`expr $MULT / $NSAMPS`
  SW=1
  EW=$WPERN
  BWIDS="$SW"
  EWIDS="$EW"
  I=2
  while [ $I -le $NSAMPS ]
  do
    SW=`expr $SW + $WPERN`
    EW=`expr $EW + $WPERN`
    BWIDS="$BWIDS $SW"
    EWIDS="$EWIDS $EW"
    I=`expr $I + 1`
  done
  NBWIDS=$NSAMPS
  NEWIDS=$NSAMPS
fi

if [ $NWIDS = 0 ]
then
  WIDS=""
  NWIDS=$NBWIDS
  I=1
  while [ $I -le $NWIDS ]
  do
    J=1
    for DUMMY in $BWIDS
    do
      if [ $J = $I ]
      then
        SW=$DUMMY
        break;
      fi
      J=`expr $J + 1`
    done

    J=1
    for DUMMY in $EWIDS
    do
      if [ $J = $I ]
      then
        EW=$DUMMY
```

```
          break;
        fi
        J=`expr $J + 1`
      done

      if [ $I = 1 ]
      then
        WIDS="$SW"
      elif [ $I = $NWIDS ]
      then
        WIDS="$WIDS $EW"
      else
        TID=`expr $SW + $EW`
        TID=`expr $TID / 2`
        WIDS="$WIDS $TID"
      fi
      I=`expr $I + 1`
    done
fi


#
# audit directory
#
if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
    SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
then
    mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

#
# run checks
#
date | tee ${ODIR}/${OFILE}
#${SDIR}/swt_temp.sh tmp

cat >> ${ODIR}/${OFILE} <<!


Number of Active Warehouses
---------------------------

!

sqlplus tpcc/tpcc > ${ODIR}/${OFILE}.0 2>&1 <<!
   set termout on
   set echo on
   set timing on

   select count(*) from ware;

   quit;
!

cat ${ODIR}/${OFILE}.0 >> ${ODIR}/${OFILE}
cat ${ODIR}/${OFILE}.0
rm -f ${ODIR}/${OFILE}.0

sqlplus tpcc/tpcc @${SQLDIR}/runchk $MULT > ${ODIR}/${OFILE}.5 &

sqlplus tpcc/tpcc @${SQLDIR}/runchk_stok $MULT > ${ODIR}/${OFILE}.6 &

sqlplus tpcc/tpcc @${SQLDIR}/remote $MULT > ${ODIR}/${OFILE}.0 &

I=1
while [ $I -le $NWIDS ]
do
   NSWID=1
   for DUMSWID in $WIDS
   do
     if [ $NSWID = $I ]
     then
       SWID=$DUMSWID
       break;
     fi
     NSWID=`expr $NSWID + 1`
   done

   J=1
   for DUMMY in $BWIDS
   do
     if [ $J = $I ]
     then
       SW=$DUMMY
       break;
     fi
     J=`expr $J + 1`
   done

   J=1
   for DUMMY in $EWIDS
   do
     if [ $J = $I ]
     then
       EW=$DUMMY
       break;
     fi
     J=`expr $J + 1`
   done

   if [ $NHOSTS = 0 ]
   then
     sqlplus tpcc/tpcc @${SQLDIR}/runchk_parallel $SWID $SW $EW | \
           tee ${ODIR}/${OFILE}.$I &
   else
     NCSTR=1
     for DUMCSTR in $SNET
     do
       if [ $NCSTR = $I ]
       then
         CSTR=$DUMCSTR
         break;
       fi
       NCSTR=`expr $NCSTR + 1`
     done
     sqlplus tpcc/tpcc@$CSTR @${SQLDIR}/runchk_parallel $SWID $SW $EW | \
           tee ${ODIR}/${OFILE}.$I &
   fi
   I=`expr $I + 1`
done

wait

I=1
while [ $I -le $NWIDS ]
do
   cat >> ${ODIR}/${OFILE} <<!



Sample $I of Run Check
----------------------

!

   cat ${ODIR}/${OFILE}.$I >> ${ODIR}/${OFILE}
   rm -f ${ODIR}/${OFILE}.$I
   I=`expr $I + 1`
done

cat >>  ${ODIR}/${OFILE} <<!


Show unused warehouses: w_id > $MUL
-----------------------

!

cat ${ODIR}/${OFILE}.0 >> ${ODIR}/${OFILE}
rm -f ${ODIR}/${OFILE}.0

cat ${ODIR}/${OFILE}.5 >> ${ODIR}/${OFILE}
rm -f ${ODIR}/${OFILE}.5

cat ${ODIR}/${OFILE}.6 >> ${ODIR}/${OFILE}
rm -f ${ODIR}/${OFILE}.6

#${SDIR}/swt_temp.sh system
date | tee -a ${ODIR}/${OFILE}
```

Runshk.sql

```
set echo on
set timing on

set heading off
select '**** MAX WAREHOUSE  **** ' from dual;
set heading on
variable max_w_id number;

exec select max(w_id) into :max_w_id from ware ;
print max_w_id;

set heading off
select '**** D_NEXT_O_ID distribution **** ' from dual;
set heading on

select d_w_id, round( avg(d_next_o_id) , 1) avg_next_oid
from dist
where d_w_id between 1 and :max_w_id
group by d_w_id
order by d_w_id;

exit ;
```

Runchk_parallel.sql

```
set echo on
set timing on

variable max_w_id number;

exec select max(w_id) into :max_w_id from ware ;
print max_w_id;

set heading off
select '**** C_PAYMENT_CNT distribution **** ' from dual;
set heading on

select /*+ index( cust, icust1 ) */ c_payment_cnt, count(*) from cust
                where c_w_id = &&1
      group by c_payment_cnt order by 1;

set heading off
select '**** O_ALL_LOCAL and MIN-MAX distribution **** ' from dual;
set heading on

select avg(o_ol_cnt) avg_ol_cnt,
       sum(o_all_local) * 100.0 / count(o_all_local) pcnt_local, count(*)
ord_cnt,
       min(o_c_id), max(o_c_id),
       min(o_d_id), max(o_d_id),
       min(o_w_id), max(o_w_id)
from ordr, tpcc_audit_tab
       where  o_w_id = &&1
                and o_entry_d >
                (select starttime from tpcc_audit_tab);

set heading off
select '**** Extract 1000 order from the run (replace X by 1000/tpmc) **** ' from dual;
set heading on


select * from ordr, tpcc_audit_tab
  where o_w_id = &&1 and o_entry_d between
     (starttime + 10/1440) and (starttime + 90/1440);


select /*+ full(hist) parallel(hist,15) */ count(*), sum (decode(h_c_w_id, h_w_id, 1, 0)) cnt_local
       from hist, tpcc_audit_tab
       where h_w_id = &&1 and h_date >
                (select starttime from tpcc_audit_tab);
exit ;
```

Runchk_stok.sql

```
set echo on
set timing on

set heading off
select '**** S_YTD distribution **** ' from dual;
set heading on


select /*+ index(stok, istok) parallel(stok,50) */ s_ytd, count(*) from stok
                where s_i_id between 1 and 1000   -- Verify only some items --
      group by s_ytd order by 1;

set heading off
select '**** S_ORDER_CNT and S_REMOTE_CNT distribution **** ' from dual;
set heading on

select /*+ index(stok, istok) parallel(stok,50) */ s_order_cnt, count(*) from stok
                where s_i_id between 1 and 1000   -- Verify only some items --
      group by s_order_cnt order by 1;

select /*+ index(stok, istok) parallel(stok,50) */ s_remote_cnt, count(*) from stok
                where s_i_id between 1 and 1000   -- Verify only some items --
      group by s_remote_cnt order by 1;

exit
```

Runscript.sh

```
#!/bin/sh
# run a script from the script directory.
# go to log directory so we don't leave a mess in the bench dir.
cd $tpcc_bench/log
/bin/sh $tpcc_genscripts_dir/${1}.sh
exit $?
```

Runsql.sh

```
#!/bin/sh
# run a sql script from the script directory.
# go to log directory so we don't leave a mess in the bench dir.
cd $tpcc_bench/log
$tpcc_sqlplus $tpcc_user_pass @$tpcc_genscripts_dir/${1}
exit $?
```

Runsqllocal.sh

```
#!/bin/sh
# run a sql script from the script directory.
$tpcc_sqlplus '/ as sysdba' @$tpcc_genscripts_dir/${1}
exit $?
```

Sampl.sh

```
#!/bin/sh
#
# $Header: sampl.sh 01-jun-98.19:12:01 skareenh Exp $
#
# sampl.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      sampl.sh
#
#    DESCRIPTION
#                Used in durability tests to sample the database after a crash.
#
#    NOTES
#                sampl.sh [output_file]
#       Need to specify the orders to sample.
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#

if [ $1 ]
then
   OFILE="$1"
else
   OFILE="sampl.v1"
fi

if [ "X$SRCHOME" = "X" -a  "X$ORACLE_HOME" != "X" ]
then
    SRCHOME=$ORACLE_HOME
fi
BENCH_HOME=$SRCHOME
TPCC_AUDIT=$BENCH_HOME/audit

#
# output directory
#
ODIR=${TPCC_AUDIT}/output
if [ ! -d $ODIR ]
```

```
then
   mkdir $ODIR
fi

#
# shell script directory
#
SDIR=${TPCC_AUDIT}/scripts

#
# SQL script directory
#
SQLDIR=${TPCC_AUDIT}/sql

#
# source directory
#
SRCDIR=${TPCC_AUDIT}/source

INPUT1="38 4 2518 3006"
INPUT2="1672 5 552 3006"
INPUT3="2354 7 1440 3008"
INPUT4="661 9 2236 3013"
INPUT5="331 5 542 3014"
INPUT6="2641 1 194 3001"

echo "" | tee ${ODIR}/${OFILE}
echo "" | tee -a ${ODIR}/${OFILE}
echo "" | tee -a ${ODIR}/${OFILE}

echo "dura.sh $INPUT1" | tee -a ${ODIR}/${OFILE}
echo "dura.sh $INPUT2" | tee -a ${ODIR}/${OFILE}
echo "dura.sh $INPUT3" | tee -a ${ODIR}/${OFILE}
echo "dura.sh $INPUT4" | tee -a ${ODIR}/${OFILE}
echo "dura.sh $INPUT5" | tee -a ${ODIR}/${OFILE}
echo "dura.sh $INPUT6" | tee -a ${ODIR}/${OFILE}

echo "" | tee -a ${ODIR}/${OFILE}
echo "" | tee -a ${ODIR}/${OFILE}
echo "" | tee -a ${ODIR}/${OFILE}

${SDIR}/dura.sh $INPUT1 | tee -a ${ODIR}/${OFILE}
${SDIR}/dura.sh $INPUT2 | tee -a ${ODIR}/${OFILE}
${SDIR}/dura.sh $INPUT3 | tee -a ${ODIR}/${OFILE}
${SDIR}/dura.sh $INPUT4 | tee -a ${ODIR}/${OFILE}
${SDIR}/dura.sh $INPUT5 | tee -a ${ODIR}/${OFILE}
${SDIR}/dura.sh $INPUT6 | tee -a ${ODIR}/${OFILE}
```

Shutdowndb.sh

```
#!/usr/bin/sh

echo "Shutting down database..."

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool shutdowndb.log;

set echo on;

alter system switch logfile;
alter system switch logfile;

shutdown immediate;

set echo off;
spool off;

exit
!
```

Space_get.sql

```
REM=============================================================================
+
REM        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
REM                  OPEN SYSTEMS PERFORMANCE GROUP                 |
REM                       All Rights Reserved                       |
REM=============================================================================
+
REM FILENAME
REM       space_get.sql
REM DESCRIPTION
REM       Get sizes of tables, indexes and tablespaces.
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @space_get [<tpm> <# of warehouses>]
REM=============================================================================
*/

set echo on;
delete from tpcc_data;
delete from tpcc_space;
delete from tpcc_totspace;

insert into tpcc_data
select substr(segment_name,1,11), substr(segment_type,1,15),
       sum(blocks), t.block_size,
       round(sum(blocks) * 0.05), 0,
       sum(blocks) + round(sum(blocks) * 0.05)
from   sys.dba_extents e, sys.dba_tablespaces t
where  owner = 'TPCC' AND ( segment_type = 'INDEX' OR
       segment_type = 'INDEX PARTITION' OR segment_type = 'CLUSTER'
       OR segment_type = 'TABLE' OR segment_type = 'TABLE PARTITION')
       AND e.tablespace_name <> 'SYSTEM' and e.tablespace_name = t.tablespace_name
group by segment_name, segment_type, t.block_size;

insert into tpcc_data
  select 'SYSTEM', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
  from   sys.dba_data_files f, sys.dba_tablespaces t
  where  f.tablespace_name = 'SYSTEM' and t.tablespace_name = 'SYSTEM'
  group by t.block_size;

insert into tpcc_data
  select 'ROLL_SEG', 'SYS',
         sum(blocks), t.block_size, 0, 0, sum(blocks)
  from   sys.dba_data_files f, sys.dba_tablespaces t
  where  f.tablespace_name like 'ROLL%' and f.tablespace_name = t.tablespace_name
  group by f.tablespace_name, t.block_size;

update tpcc_data
  set five_pct = 0,
      daily_grow = round(blocks * &&1 / 62.5 / &&2),
      total = blocks + round(blocks * &&1 / 62.5 / &&2)
  where segment = 'HIST'  OR
        segment = substr('ORDRCLUSTER_QUEUE',1,11);

insert into tpcc_space
select substr(ex$.name,1,11), sum(sp$.sz_blocks), sp$.block_size, 0, 0, 0, 0
from
 (select f.tablespace_name , sum(blocks) sz_blocks, t.block_size block_size
  from sys.dba_data_files f, sys.dba_tablespaces t
  where f.tablespace_name <> 'SYSTEM' and f.tablespace_name = t.tablespace_name
  group by f.tablespace_name, t.block_size
 ) sp$,
 (select distinct tablespace_name, segment_name name
  from sys.dba_extents
  where owner = 'TPCC'
    and (segment_type = 'CLUSTER' or segment_type = 'TABLE'
     or segment_type = 'TABLE PARTITION' or segment_type = 'INDEX'
     or segment_type = 'INDEX PARTITION')
    and tablespace_name <> 'SYSTEM'
 ) ex$
where sp$.tablespace_name = ex$.tablespace_name
group by ex$.name, sp$.block_size;

insert into tpcc_space
select substr(f.tablespace_name,1,11), sum(blocks), t.block_size, 0, 0, 0, 0
from sys.dba_data_files f, sys.dba_tablespaces t
where f.tablespace_name = 'SYSTEM' and f.tablespace_name = t.tablespace_name
group by f.tablespace_name, t.block_size;

update tpcc_space
  set required =
  (
    select sum(total)
    from   tpcc_data
    where  tpcc_data.segment = tpcc_space.segment
  )
  where segment in
  (
    select segment from tpcc_data
  );

update tpcc_space
  set static =
  (
    select sum(total)
    from   tpcc_data
    where  tpcc_data.segment = tpcc_space.segment
  )
  where segment in
  (
    select segment from tpcc_data
  );

update tpcc_space
  set static = 0,
      dynamic =
  (
    select sum(blocks)
    from   tpcc_data
    where  tpcc_data.segment = tpcc_space.segment
  )
```

```
    where segment in ('HIST',
                substr('ORDRCLUSTER_QUEUE',1,11));

  update tpcc_space
    set oversize = blocks  - required;

  insert into tpcc_totspace
    select &&1, &&2, sum(static * block_size)/1024, sum(dynamic * block_size)/1024,
sum(oversize * block_size)/1024, 0, 0, 0
    from   tpcc_space;

  update tpcc_totspace
    set daily_grow =
    (
      select sum(daily_grow * block_size)/1024
      from   tpcc_data
    );
  update tpcc_totspace
    set space60 = static + 60 * daily_grow;
  set echo off;
```

Space_init.sql

```
REM============================================================
+
REM FILENAME
REM       space_init.sql
REM DESCRIPTION
REM       Create tables for space calculations.
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @space_init.sql
REM============================================================
*/
  set echo on;
  drop table tpcc_data;
  drop table tpcc_space;
  drop table tpcc_totspace;
  create table tpcc_data (
    segment      varchar2(11),
    type         varchar2(15),
    blocks       number,
    block_size   number,
    five_pct     number,
    daily_grow   number,
    total        number
  );
  create table tpcc_space (
    segment       varchar2(11),
    blocks        number,
    block_size    number,
    required      number,
    static        number,
    dynamic       number,
    oversize      number
  );
  create table tpcc_totspace (
    tpm           number,
    nware         number,
    static        number,
    dynamic       number,
    oversize      number,
    daily_grow    number,
    daily_spre    number,
    space60       number
  );
  create unique index itpcc_data on tpcc_data (segment);
  create unique index itpcc_space on tpcc_space (segment);
  set echo off;
```

Space.sh

```
#!/bin/ksh

sqlplus tpcc/tpcc << !!
@space_init
@space_get 80570 6700
@space_rpt
exit;
!!
```

Space_rpt.sql

```
REM============================================================
+
REM        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA     |
REM                  OPEN SYSTEMS PERFORMANCE GROUP                |
REM                       All Rights Reserved                      |
REM============================================================
+
REM FILENAME
REM       space_rpt.sql
REM DESCRIPTION
REM       Generate space report and save it in space.rpt
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @space_rpt.sql
REM============================================================
*/
  set space 2
  set pagesize 2000
  set echo off
  set termout off
  set verify off
  set feedback off
  set pagesize 60 linesize 120
  spool space.rpt
  select tpm, nware from tpcc_totspace;
  select * from tpcc_data order by segment;
  select * from tpcc_space order by segment;
  select 'Unit of Total Space Reported is 1K Blocks' "Measurement units"
         from dual;
  select static, dynamic, oversize, daily_grow, daily_spre, space60
    from tpcc_totspace;
  spool off;
```

Statupdb.sh

```
#!/usr/bin/sh

echo "Starting up database using $1..."

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool startdb.log

set echo on

startup pfile=${1}.ora open

spool off
set echo off
exit sql.sqlcode
!
```

Stepenv.sh

```
# forces any env variables we set to be exported
set -a
tpcc_kit=t
tpcc_bench=$PWD
tpcc_scripts=$tpcc_bench/scripts
tpcc_require=$tpcc_scripts/require_vars.sh
tpcc_lcm=$tpcc_scripts/lcm.sh
tpcc_tokilobytes=$tpcc_scripts/tokilobytes.sh
tpcc_fromkilobytes=$tpcc_scripts/fromkilobytes.sh
tpcc_notneg=$tpcc_scripts/notneg.sh
tpcc_isneg=$tpcc_scripts/isneg.sh

if test -x /usr/bin/bc; then
tpcc_bcexpr=$tpcc_scripts/bcexpr.sh
else
tpcc_bcexpr=expr
fi


# the ksh version is a bit faster, so we want
# to use it if we have ksh.  Otherwise we have
# a compatible version.
if test -x /usr/bin/ksh; then
tpcc_createts=$tpcc_scripts/createts.ksh
else
tpcc_createts=$tpcc_scripts/createts.sh
fi

tpcc_tabledata=$tpcc_scripts/tabledata.sh
tpcc_load=$tpcc_bench/benchrun/bin/tpccload.exe
tpcc_createtablespaces=$tpcc_scripts/createtablespaces.sh

##
tpcc_sqlplus=cat
tpcc_sqlplus_args='/nolog'
tpcc_internal_connect='connect / as sysdba'
```

```
tpcc_user_pass='tpcc/tpcc'
tpcc_dba_user_pass='system/manager'
oracle_dba=system
oracle_dba_password=manager
tpcc_sqlplus_args=
tpcc_user_pass=
tpcc_sqlplus=sqlplus
tpcc_user_pass='tpcc/tpcc'

# import options generated by gui
. ${tpcc_bench}/options.sh

#8gb oracle filesize limit (in k)
tpcc_fsize_limit_k=8388608
#2gb - 1k oracle extent limit (in k)
tpcc_extent_limit_k=2097151

# Runlen calculations are in minutes, so multiply by
# 60, and 8 times.
tpcc_runlen=`$tpcc_bcexpr 8 \* 60 \* $tpcc_runlen`

tpcc_system_size=200M
tpcc_logfile_size=`$tpcc_bcexpr 20 + \( $tpcc_scale \)`M

tpcc_undo_size=`$tpcc_bcexpr 2 \* $tpcc_scale`M
tpcc_undo_bs=8K

tpcc_statspack_size=`$tpcc_bcexpr 1 \* $tpcc_scale`M
tpcc_sysaux_size=120M

# fixed table params

#table list (note temp is always at the end since it may use numbers from other tables, and it's not
included in these lists)
tpcc_table_list='ware cust dist hist stok item ordr ordl nord'
tpcc_index_list='iware icust1 icust2 idist istok iitem iordr1 iordr2 iordl inord'
#for these I use average row length, calculated from multi-blocksize stats.
#we figure out how many new rows we will gain in a run (in createtablesspaces.sh)
#and add that much to the base tablespace size.
tpcc_hist_growth=51
tpcc_ordr_growth=35
tpcc_nord_growth=13
#tpcc_ordl_growth=660
tpcc_ordl_growth=900

#i started indices at 1/10th... need an exact figure
tpcc_iordr1_growth=20
tpcc_iordr2_growth=20
tpcc_iordl_growth=66
tpcc_inord_growth=2

tpcc_item_growth=0
tpcc_iitem_growth=0
tpcc_temp_growth=0

tpcc_cust_growth=regular
tpcc_icust1_growth=regular
tpcc_icust2_growth=regular

tpcc_stok_growth=regular
tpcc_istok_growth=regular

tpcc_ware_growth=regular
tpcc_iware_growth=regular

tpcc_dist_growth=regular
tpcc_idist_growth=regular

# minimum size of temp tablespace
tpcc_tempts_min=10240

# for Linux, set appropriate tablespace heuristics
# to set high io tables to have 64 files, and minimize
# others.
if expr $tpcc_os = linux > /dev/null; then
  for table in $tpcc_table_list $tpcc_index_list temp; do
    eval "tpcc_${table}_tsfileinc=1"
  done
  tpcc_os=unix

  tpcc_stok_tsfileinc=64
  tpcc_cust_tsfileinc=64
  tpcc_iordl2_tsfileinc=16
  tpcc_icust2_tsfileinc=16
  tpcc_iordl_tsfileinc=16
else
#in case someone changes out of linux, and the shell is stuck
  tpcc_stok_tsfileinc=
  tpcc_cust_tsfileinc=
  tpcc_iordl2_tsfileinc=
  tpcc_icust2_tsfileinc=
  tpcc_iordl_tsfileinc=
fi

# import local options
. ${tpcc_bench}/localoptions.sh

if expr `echo x$tpcc_no_options` = xt > /dev/null; then
  echo Please modify ${tpcc_bench}/localoptions.sh to configure the generator.
  exit 1
fi

tpcc_fixordrordl=${tpcc_genscripts_dir}/loadfixordrordl.sh
tpcc_updateordrordl=${tpcc_scripts}/updateordrordl.sh

#tp- get table param.  (that is, $tpcc_tablename_tableparam)
tp(){
  eval echo \"\"\$tpcc_$1_$2"\"
}

# automatically generated variables
if expr `echo $tpcc_version | cut -b1` = t > /dev/null; then
  tpcc_auto_undo=t
else
  tpcc_auto_undo=f
fi
if expr `echo $tpcc_version | cut -b2` = t > /dev/null; then
  tpcc_autospace_avail=t
else
  tpcc_autospace_avail=f
fi
if expr `echo $tpcc_version | cut -b3` = t > /dev/null; then
  tpcc_queue_avail=t
else
  tpcc_queue_avail=f
fi

# used for loading program
if test x$tpcc_overflow = xt; then
  tpcc_hash_overflow=t
fi

tpcc_create_steps="buildcreatets buildcreatedb \
buildcreatetable-ware buildcreatetable-cust buildcreatetable-dist buildcreatetable-hist
buildcreatetable-stok buildcreatetable-item buildcreatetable-ordr buildcreatetable-ordl
buildcreatetable-nord \
buildloadware buildloaddist buildloaditem buildloadhist buildloadnord buildloadordrordl
buildloadcust buildloadstok buildfixoo \
buildcreateindex-iware buildcreateindex-icust1 buildcreateindex-icust2 buildcreateindex-idist
buildcreateindex-istok buildcreateindex-iitem buildcreateindex-iordr1 buildcreateindex-iordr2
buildcreateindex-iordl buildcreateindex-inord \
listfiles
"

tpcc_steps="runsqllocal-createdb shutdowndb startupdb-p_build createuser runscript-createts
assigntemp ddview \
  runsql-createtable_ware runsql-createtable_cust runsql-createtable_dist runsql-createtable_hist
runsql-createtable_stok runsql-createtable_item runsql-createtable_ordr runsql-createtable_ordl
runsql-createtable_nord \
runscript-loadware runscript-loaddist runscript-loaditem runscript-loadhist runscript-loadnord
runscript-loadordrordl runscript-loadcust runscript-loadstok \
 runsql-createindex_iware runsql-createindex_icust1 runsql-createindex_icust2 runsql-
createindex_idist runsql-createindex_istok runsql-createindex_iitem runsql-createindex_iordr1
runsql-createindex_iordr2 runsql-createindex_iordl runsql-createindex_inord \
analyze runscript-loadfixordrordl createstats createstoredprocs createspacestats createmisc"

# no longer automatically exports env variables
set +a

# check for problems with configuration
badconf=
for table in $tpcc_table_list; do
  if expr `tp $table imp` = queue > /dev/null; then
    if expr $tpcc_queue_avail = f > /dev/null; then
                echo Table $table may not be a queue, since queues are
                echo are unavailable in the selected Oracle version.
                badconf=t
    fi
  fi
  if expr $tpcc_autospace_avail = f \& `tp $table autospace` = t > /dev/null; then
    echo Table $table may not use bitmapped space management
    echo since it is not available in the selected Oracle version.
    badconf=t
  fi
done

if test -n "$badconf"; then
  exit 1
fi

# make sure we have everything
if $tpcc_require ORACLE_SID \
 tpcc_tokilobytes tpcc_createts tpcc_lcm\
 tpcc_sqlplus tpcc_internal_connect\
 tpcc_np tpcc_cpu tpcc_os tpcc_runlen tpcc_ldrive tpcc_scale tpcc_disks_location
tpcc_auto_undo tpcc_tempts_min\
 tpcc_system_size tpcc_logfile_size\
 tpcc_undo_size tpcc_undo_bs\
 oracle_dba oracle_dba_password tpcc_dba_user_pass
```

```
then exit 1; fi
```

Tabledata.sh

```
# tabledata- (not a script) data and functions for
# getting columns and info for creating tables and indices.

# forces any env variables we set to be exported
set -a

# note- clustcols will be an option- move to stepenv file!

#tp- get table param.  (that is, $tpcc_tablename_tableparam)
tp(){
  eval echo \"\"\$tpcc_$1_$2"\"
}

#nullauto- check if $1 null or auto, if not echo it, otherwise echo $2.
nullauto(){
  if expr "x$1" = x \| "x$1" = xauto \| "x$1" = xno > /dev/null; then
    echo "$2"
  else
    echo "$1"
  fi
}

tpcc_item_warecol=none
tpcc_item_clustcols=t
tpcc_item_hkey="`nullauto "$tpcc_item_hkey" 100000`"
tpcc_item_hash="`nullauto "$tpcc_item_hash" i_id`"
tpcc_item_rsize="`nullauto "$tpcc_item_rsize" 120`"

tpcc_item_cols=`cat <<!
        i_id-number(6,0)
        i_name-varchar2(24)
        i_price-number
        i_data-varchar2(50)
        i_im_id-number
!
`

tpcc_iitem_cols=$tpcc_item_cols
tpcc_iitem_warecol=none
tpcc_iitem_indexon=item

if expr x$tpcc_compress = xt > /dev/null; then
  tpcc_stok_compress=ffffffttttttttttt
  tpcc_stok_rsize="`nullauto "$tpcc_stok_rsize" 256`"
tpcc_stok_cols=`cat <<!
        s_i_id-number
        s_w_id-number
        s_quantity-number
        s_ytd-number
        s_order_cnt-number
        s_remote_cnt-number
        s_data-varchar2(50)
        s_dist_01-varchar2(24)
        s_dist_02-varchar2(24)
        s_dist_03-varchar2(24)
        s_dist_04-varchar2(24)
        s_dist_05-varchar2(24)
        s_dist_06-varchar2(24)
        s_dist_07-varchar2(24)
        s_dist_08-varchar2(24)
        s_dist_09-varchar2(24)
        s_dist_10-varchar2(24)
!
`
else
tpcc_stok_rsize="`nullauto "$tpcc_stok_rsize" 350`"
tpcc_stok_cols=`cat <<!
        s_i_id-number
        s_w_id-number
        s_quantity-number
        s_ytd-number
        s_order_cnt-number
        s_remote_cnt-number
        s_data-varchar2(50)
        s_dist_01-char(24)
        s_dist_02-char(24)
        s_dist_03-char(24)
        s_dist_04-char(24)
        s_dist_05-char(24)
        s_dist_06-char(24)
        s_dist_07-char(24)
        s_dist_08-char(24)
        s_dist_09-char(24)
        s_dist_10-char(24)
!
`
fi
tpcc_stok_clustcols=tt
tpcc_stok_hkey_def=`$tpcc_bcexpr 100000 \\* $tpcc_scale`
tpcc_stok_hkey="`nullauto "$tpcc_stok_hkey" "$tpcc_stok_hkey_def"`"
if expr $tpcc_stok_imp = parttable \| $tpcc_stok_imp = partiot > /dev/null; then
  tmpdiv="$tpcc_scale/$tpcc_np"
  tpcc_stok_hash_def="(abs(s_i_id - 1) * $tmpdiv + mod((s_w_id - 1), $tmpdiv) + trunc ((s_w_id
- 1) / $tmpdiv) * $tmpdiv * 100000)"
else
  tpcc_stok_hash_def="(s_i_id * $tpcc_scale + s_w_id)"
fi
tpcc_stok_hash="`nullauto "$tpcc_stok_hash" "$tpcc_stok_hash_def"`"

tpcc_stok_warecol=s_w_id

tpcc_istok_cols=$tpcc_stok_cols
tpcc_istok_warecol=$tpcc_stok_warecol
tpcc_istok_indexon=stok

tpcc_ware_clustcols=t
tpcc_ware_hkey="`nullauto "$tpcc_ware_hkey" "$tpcc_scale"`"
tpcc_ware_hash="`nullauto "$tpcc_ware_hash" '(w_id)'`"
tpcc_ware_rsize="`nullauto "$tpcc_ware_rsize" 1536`"
tpcc_ware_warecol=none
tpcc_ware_cols=`cat <<!
        w_id-number(5,0)
        w_ytd-number
        w_tax-number
        w_name-varchar2(10)
        w_street_1-varchar2(20)
        w_street_2-varchar2(20)
        w_city-varchar2(20)
        w_state-char(2)
        w_zip-char(9)
!
`

tpcc_iware_cols=$tpcc_ware_cols
tpcc_iware_warecol=none
tpcc_iware_indexon=ware

tpcc_dist_clustcols=tt
tpcc_dist_hkey_def=`$tpcc_bcexpr 10 \* $tpcc_scale`
tpcc_dist_hkey="`nullauto "$tpcc_dist_hkey" $tpcc_dist_hkey_def`"
tpcc_dist_hash="`nullauto "$tpcc_dist_hash" '((d_w_id * 10) + d_id)'`"
tpcc_dist_rsize="`nullauto "$tpcc_dist_rsize" 1536`"
tpcc_dist_warecol=d_w_id
tpcc_dist_cols=`cat <<!
        d_id-number
        d_w_id-number
        d_ytd-number
        d_next_o_id-number
        d_tax-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)
!
`

tpcc_idist_cols=$tpcc_dist_cols
tpcc_idist_warecol=d_w_id
tpcc_idist_indexon=dist

#NOTE- needs to change SQL code
tpcc_ordr_clustcols=ttttfffft
tpcc_ordr_queuesort=2t-3t-1tt-9tt-
tpcc_ordr_queuenames=2-3-1-
#NOTE- check if this is right for hash keys
tpcc_ordr_hkey_def=`$tpcc_bcexpr 10 \* $tpcc_scale`
tpcc_ordr_hkey="`nullauto "$tpcc_ordr_hkey" "$tpcc_ordr_hkey_def"`"
tpcc_ordr_hash="`nullauto "$tpcc_ordr_hash" '(o_w_id - 1) * 10 + o_d_id - 1'`"
tpcc_ordr_rsize="`nullauto "$tpcc_ordr_rsize" 1490`"
tpcc_ordr_warecol=o_w_id

if expr $tpcc_ordr_imp = queue > /dev/null; then
# buildcreatetable sets cols to this after tablecols
# are fetched from normal ordr_cols, so we can get the
# o_number field out for queuesort. (bit of a hack, but
# it would be difficult to redo it another way)
tpcc_ordr_qcols=`cat <<!
        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
        o_number-number
!
`

tpcc_ordr_cols=`cat <<!
        o_id-number-sort
```

```
        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
!
`
else
tpcc_ordr_cols=`cat <<!
        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
!
`
fi

tpcc_iordr1_warecol=$tpcc_ordr_warecol
tpcc_iordr1_cols=$tpcc_ordr_cols
tpcc_iordr1_indexon=ordr

tpcc_iordr2_warecol=$tpcc_ordr_warecol
tpcc_iordr2_cols=$tpcc_ordr_cols
tpcc_iordr2_indexon=ordr

#ordl uses ordr's cluster, so we represent this
#with another variable
tpcc_ordl_usecluster=', constraint ordl_uk primary key (ol_w_id, ol_d_id, ol_o_id, ol_number  ))
CLUSTER ordrcluster_queue(ol_w_id, ol_d_id, ol_o_id, ol_number)'
tpcc_ordl_othercluster=ordr

tpcc_ordl_warecol=ol_w_id
if test $tpcc_ordl_imp = queue; then
tpcc_ordl_cols=`cat <<!
        ol_w_id-number
        ol_d_id-number
        ol_o_id-number-sort
        ol_number-number-sort
        ol_i_id-number
        ol_delivery_d-date
        ol_amount-number
        ol_supply_w_id-number
        ol_quantity-number
        ol_dist_info-char(24)
!
`
else
tpcc_ordl_cols=`cat <<!
        ol_w_id-number
        ol_d_id-number
        ol_o_id-number
        ol_number-number
        ol_i_id-number
        ol_delivery_d-date
        ol_amount-number
        ol_supply_w_id-number
        ol_quantity-number
        ol_dist_info-char(24)
!
`
fi


tpcc_iordl_cols=$tpcc_ordl_cols
tpcc_iordl_warecol=$tpcc_ordl_warecol
tpcc_iordl_indexon=ordl

tpcc_nord_clustcols=ttt
tpcc_nord_queuesort=1t-2t-3tt-
tpcc_nord_queuenames=1-2-3-
#NOTE- check if this is right for hash keys
tpcc_nord_hkey_def=`$tpcc_bcexpr 10 \* $tpcc_scale`
tpcc_nord_hkey="`nullauto "$tpcc_nord_hkey" "$tpcc_nord_hkey_def"`"
tpcc_nord_hash="`nullauto "$tpcc_nord_hash" '(no_w_id - 1) * 10 + no_d_id - 1'`"
tpcc_nord_rsize="`nullauto "$tpcc_nord_rsize" 190`"
tpcc_nord_warecol=no_w_id
tpcc_nord_cols=`cat <<!
        no_w_id-number
        no_d_id-number
        no_o_id-number
!
`

tpcc_inord_cols=$tpcc_nord_cols
tpcc_inord_warecol=$tpcc_nord_warecol
tpcc_inord_indexon=nord

if expr x$tpcc_overflow = xt > /dev/null; then
  if expr x$tpcc_compress = xt > /dev/null; then
    tpcc_cust_compress=ffffffffffffttffffff
    tpcc_cust_rsize="`nullauto "$tpcc_cust_rsize" 160`"
  else
    tpcc_cust_rsize="`nullauto "$tpcc_cust_rsize" 180`"
  fi

tpcc_cust_cols=`cat <<!
        c_id-number
        c_d_id-number
        c_w_id-number
        c_discount-number
        c_credit-char(2)
        c_last-varchar2(16)
        c_first-varchar2(16)
        c_credit_lim-number
        c_balance-number
        c_ytd_payment-number
        c_payment_cnt-number
        c_delivery_cnt-number
        c_street_1-varchar2(20)
        c_street_2-varchar2(20)
        c_city-varchar2(20)
        c_state-char(2)
        c_zip-char(9)
        c_phone-char(16)
        c_since-date
        c_middle-char(2)
        c_data-varchar2(500)
!
`
else
  tpcc_cust_rsize="`nullauto "$tpcc_cust_rsize" 850`"
tpcc_cust_cols=`cat <<!
        c_id-number
        c_d_id-number
        c_w_id-number
        c_discount-number
        c_credit-char(2)
        c_last-varchar2(16)
        c_first-varchar2(16)
        c_credit_lim-number
        c_balance-number
        c_ytd_payment-number
        c_payment_cnt-number
        c_delivery_cnt-number
        c_street_1-varchar2(20)
        c_street_2-varchar2(20)
        c_city-varchar2(20)
        c_state-char(2)
        c_zip-char(9)
        c_phone-char(16)
        c_since-date
        c_middle-char(2)
        c_data-varchar2(500)
!
`
fi

tpcc_cust_clustcols=ttt
tpcc_cust_hkey_def=`$tpcc_bcexpr 30000 \* $tpcc_scale`
tpcc_cust_hkey="`nullauto "$tpcc_cust_hkey" "$tpcc_cust_hkey_def"`"
if expr $tpcc_np \> 1 > /dev/null; then
  tpcc_cust_hash_def='(c_w_id * 30000 + c_id * 10 + c_d_id - 30011)'
else
  tpcc_cust_hash_def="(c_id * ( $tpcc_scale * 10 ) + c_w_id * 10 + c_d_id)"
fi
tpcc_cust_hash="`nullauto "$tpcc_cust_hash" "$tpcc_cust_hash_def"`"
tpcc_cust_rsize="`nullauto "$tpcc_cust_rsize" 850`"
tpcc_cust_warecol=c_w_id

tpcc_icust1_cols=$tpcc_cust_cols
tpcc_icust1_warecol=$tpcc_cust_warecol
tpcc_icust1_indexon=cust

tpcc_icust2_cols=$tpcc_cust_cols
tpcc_icust2_warecol=$tpcc_cust_warecol
tpcc_icust2_indexon=cust

tpcc_hist_warecol=h_w_id
tpcc_hist_cols=`cat <<!
        h_c_id-number
        h_c_d_id-number
        h_c_w_id-number
        h_d_id-number
        h_w_id-number
        h_date-date
        h_amount-number
        h_data-varchar2(24)
!
`

set +a
```

```
#indexnames- with colums for table $1, pull out the desired indices
#and echo each.  Also allowed in format, two characters after the
#index specifying if type and SORT should be listed as well-
#example 1t-4t-2tt-3tt gives the type on each, but only SORT on the
#two entries.
#echos blank if cannot index.
indexnames(){
  tabcols=`tp $1 cols`
  if expr x$2 = xno \| x$2 = x > /dev/null; then
    echo ""
  else
    colsleft=$2
    addcomma=
    while test -n "$colsleft"; do
      desiredcol=`echo $colsleft | cut -f1 -d-`
      yestype=x`echo $desiredcol | cut -b2`
      yessort=x`echo $desiredcol | cut -b3`
      desiredcol=`echo $desiredcol | cut -b1`

      desiredcol=`echo $tabcols | cut -d' ' -f$desiredcol`
      if expr $yestype = xt > /dev/null; then
        firstcol=`echo $desiredcol | cut -f1 -d-`
        desiredcol="$firstcol `echo $desiredcol | cut -f2 -d-`"
      else
        desiredcol=`echo $desiredcol | cut -f1 -d-`
      fi
      if expr $yessort = xt > /dev/null; then
        desiredcol="$desiredcol SORT"
      fi

      echo $addcomma `echo $desiredcol | cut -f1 -d-`
      addcomma=,
      colsleft=`echo $colsleft | cut -f2- -d-`
    done
  fi
}

#colize- get columns desired for table $1, print with commas between.
#second argument is an string like ttfftft, representing
#which colums are to be displayed.
# $3 true if only name should be printed, otherwise name and value are printed
# $4 has all columns that need to have SORT after them (for queue)
# $5 has all the columns that need to have compress after them (for hash tables)
colize(){
  if expr $2 = all > /dev/null; then
    allcols=t
  else
    enabledcols=$2
  fi
  sortcols=$4
  compresscols=$5

  tabcols=`tp $1 cols`
  addcomma=
  for curcol in $tabcols ; do
    if expr x`echo $sortcols | cut -b1` = xt > /dev/null; then
      sortsql=SORT
    else
      sortsql=
    fi

    if expr x`echo $compresscols | cut -b1` = xt > /dev/null; then
      compresssql=compress
    else
      compresssql=
    fi

    if expr x`echo $enabledcols | cut -b1` = xt \| x$allcols = xt > /dev/null; then
      if expr x$3 = xt > /dev/null; then
        echo $addcomma `echo $curcol | cut -f1 -d-`
      else
        if expr x$allcols = xt > /dev/null; then
          echo $addcomma `echo $curcol | cut -f1 -d-` `echo $curcol | cut -f2 -d-` `echo $curcol | cut -
f3 -d- | tr . ' '` $sortsql $compresssql
        else
          echo $addcomma `echo $curcol | cut -f1 -d-` `echo $curcol | cut -f2 -d-` $sortsql
$compresssql
        fi
      fi
    fi
    addcomma=,
    enabledcols=`echo $enabledcols | cut -b2-`
    sortcols=`echo $sortcols | cut -b2-`
    compresscols=`echo $compresscols | cut -b2-`
  done
}

#createpart- create the sql to generate partitions based on warehouse.
createpart(){
  warecol=`tp $1 warecol`

  #check if unavailable for partitioning
  if test -z "$warecol"; then
    exit 1
  fi
  if expr $warecol = none > /dev/null; then
    exit 1
  fi

  echo "partition by range( $warecol ) ("

  partcount=0
  waredivide=0
  wareinc=`expr $tpcc_scale / $tpcc_np`
  wareextra=`expr $tpcc_scale \% $tpcc_np`
  addcomma=
  while expr $partcount \< $tpcc_np > /dev/null; do
    waredivide=`expr $wareinc + $waredivide + \( $wareextra / $tpcc_np \)`

    echo $addcomma partition ${1}_$partcount values less than \( `expr 1 + $waredivide` \)
tablespace ${1}_$partcount
    addcomma=,

    wareextra=`expr $wareextra + 1`
    partcount=`expr $partcount + 1`
  done
  echo ')'
}

#createindexpart- create the sql to generate partitions based on warehouse,
# this time for indices (so no calculation is required.)
createindexpart(){
  echo 'local ('
  partcount=0
  addcomma=
  while expr $partcount \< $tpcc_np > /dev/null; do
    echo $addcomma partition ${1}_$partcount tablespace ${1}_$partcount
    addcomma=,
    partcount=`expr $partcount + 1`
  done
  echo ')'
}
```

Tkvcbnew.sql

```
DECLARE   /* new order */
    not_serializable        EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock                EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old        EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  BEGIN
    LOOP BEGIN
      SELECT c_discount, c_last, c_credit
        INTO :c_discount, :c_last, :c_credit
        FROM cust
       WHERE c_id = :c_id
         AND c_d_id = :d_id
         AND c_w_id = :w_id;

      UPDATE wh_dist SET d_next_o_id = d_next_o_id + 1, d_tax=d_tax+0
       WHERE  d_id = :d_id
         AND  w_id = :w_id
       RETURNING d_tax, d_next_o_id-1, w_tax
        INTO :d_tax, :o_id, :w_tax;

      INSERT INTO nord  (no_o_id, no_d_id, no_w_id)
          VALUES (:o_id , :d_id, :w_id);
      INSERT INTO ordr (o_id, o_w_id, o_d_id, o_c_id, o_carrier_id,
                  o_ol_cnt, o_all_local,o_entry_d)
          VALUES (:o_id, :w_id, :d_id, :c_id, 11,
                :o_ol_cnt, :o_all_local, :cr_date);
      EXIT;

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

Tkvcini.sql

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

CREATE OR REPLACE PACKAGE inittpcc
AS
 TYPE intarray IS TABLE OF INTEGER INDEX BY BINARY_INTEGER;
```

```
 TYPE distarray IS TABLE OF VARCHAR(24) INDEX BY BINARY_INTEGER;
 nulldate      DATE;
 TYPE rowidarray IS TABLE OF ROWID INDEX BY PLS_INTEGER;
 s_dist                                        distarray;
 idx1arr                        intarray;
 s_remote                       intarray;
 dist               intarray;
 row_id               rowidarray;
 cust_rowid            rowid;
 dist_name             VARCHAR2(11);
 ware_name              VARCHAR2(11);
 c_num               PLS_INTEGER;

 PROCEDURE init_no(idxarr intarray);
 PROCEDURE init_del;
 PROCEDURE init_pay;
END inittpcc;
/
show errors;

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

  PROCEDURE init_del
  IS
  BEGIN
    FOR i IN 1 .. 10 LOOP
      dist(i) := i;
    END LOOP;
  END init_del;

  PROCEDURE init_pay IS
  BEGIN
    NULL;
  END init_pay;

END inittpcc;
/
show errors
exit
```

Tkvcpdel_iso5.sql

```
declare
  TYPE numarray IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
  TYPE numlist is varray (10) of number;
  dist numarray;
  amt numarray ;
  cnt pls_integer;
  node_num      VARCHAR2(10);

  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
  SELECT substr(value,1,5)
        INTO node_num
        FROM v$parameter
        WHERE name = 'instance_number';

  plsql_mon_pack.print ('Delivery started at ' ||
                        to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                        node_num);

  LOOP BEGIN
    FORALL d IN 1..10
      DELETE FROM nord
        WHERE no_d_id = inittpcc.dist(d)
          AND no_w_id = :w_id
          AND ROWNUM <= 1
        RETURNING no_d_id, no_o_id BULK COLLECT INTO :d_id, :order_id;

    :ordcnt := SQL%ROWCOUNT;

    FORALL o in 1.. :ordcnt
      UPDATE ordr SET o_carrier_id = :carrier_id
      WHERE o_id = :order_id (o)
        AND o_d_id = :d_id(o)
        AND o_w_id = :w_id
      RETURNING o_c_id BULK COLLECT INTO :o_c_id;

    FORALL o in 1.. :ordcnt
      UPDATE ordl SET ol_delivery_d = :now
      WHERE ol_w_id = :w_id
        AND ol_d_id = :d_id(o)
        AND ol_o_id = :order_id(o)
      RETURNING sum(ol_amount) BULK COLLECT INTO  :sums;

    FORALL c IN 1.. :ordcnt
      UPDATE /*+ VECTOR_READ */ cust
        SET c_balance = c_balance + :sums(c),
                    c_delivery_cnt = c_delivery_cnt + 1
      WHERE c_w_id = :w_id
        AND c_d_id = :d_id(c)
        AND c_id = :o_c_id(c);
    plsql_mon_pack.print ('Delivery [dist 5] sleep before commit at '
                          || to_char (sysdate, 'HH24:MI:SS'));
    plsql_mon_pack.print ('   cust_id = ' || to_char (:o_c_id(5)) ||
                          '   amount_sum = ' ||
                          to_char (:sums(5)/100, '9999999999.99'));
    dbms_lock.sleep (60);
    plsql_mon_pack.print ('Delivery [dist 5] wake up at ' ||
                          to_char (sysdate, 'HH24:MI:SS'));
    COMMIT;
    EXIT;
    EXCEPTION
      WHEN not_serializable OR deadlock OR snapshot_too_old
      THEN
        ROLLBACK;
        :retry := :retry + 1;
    END;

  END LOOP; -- for retry
END;
```

Tkvcpdel_iso6.sql

```
declare
  TYPE numarray IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
  TYPE numlist is varray (10) of number;
  dist numarray;
  amt numarray ;
  cnt pls_integer;
  node_num      VARCHAR2(10);

  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
  SELECT substr(value,1,5)
        INTO node_num
        FROM v$parameter
        WHERE name = 'instance_number';

  plsql_mon_pack.print ('Delivery started at ' ||
                        to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                        node_num);

  LOOP BEGIN
    FORALL d IN 1..10
      DELETE FROM nord
        WHERE no_d_id = inittpcc.dist(d)
          AND no_w_id = :w_id
          AND ROWNUM <= 1
        RETURNING no_d_id, no_o_id BULK COLLECT INTO :d_id, :order_id;

    :ordcnt := SQL%ROWCOUNT;

    FORALL o in 1.. :ordcnt
      UPDATE ordr SET o_carrier_id = :carrier_id
      WHERE o_id = :order_id (o)
        AND o_d_id = :d_id(o)
        AND o_w_id = :w_id
      RETURNING o_c_id BULK COLLECT INTO :o_c_id;

    FORALL o in 1.. :ordcnt
      UPDATE ordl SET ol_delivery_d = :now
      WHERE ol_w_id = :w_id
        AND ol_d_id = :d_id(o)
        AND ol_o_id = :order_id(o)
      RETURNING sum(ol_amount) BULK COLLECT INTO  :sums;

    FORALL c IN 1.. :ordcnt
      UPDATE  cust
        SET c_balance = c_balance + :sums(c),
                    c_delivery_cnt = c_delivery_cnt + 1
      WHERE c_w_id = :w_id
        AND c_d_id = :d_id(c)
        AND c_id = :o_c_id(c);
    plsql_mon_pack.print ('Delivery [dist 5] sleep before commit at '
```

```
                        || to_char (sysdate, 'HH24:MI:SS'));
   plsql_mon_pack.print ('   cust_id = ' || to_char (:o_c_id(5)) ||
                        '   amount_sum = ' ||
                        to_char (:sums(5)/100, '9999999999.99'));
   dbms_lock.sleep (60);
   plsql_mon_pack.print ('Delivery [dist 5] wake up at ' ||
                        to_char (sysdate, 'HH24:MI:SS'));
   ROLLBACK;
   EXIT;
   EXCEPTION
    WHEN not_serializable OR deadlock OR snapshot_too_old
    THEN
     ROLLBACK;
     :retry := :retry + 1;
   END;

  END LOOP; -- for retry
END;
```

Tkvcpdel_iso8.sql

```
declare
  TYPE numarray IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
  TYPE numlist is varray (10) of number;
  dist numarray;
  amt numarray ;
  cnt pls_integer;
  no_ordr binary_integer;
  order_number binary_integer;
  node_num      VARCHAR2(10);

  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
  SELECT substr(value,1,5)
       INTO node_num
       FROM v$parameter
       WHERE name = 'instance_number';

  plsql_mon_pack.print ('Delivery started at ' ||
                       to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                       node_num);

  LOOP BEGIN
   FORALL d IN 1..10
    DELETE FROM nord
     WHERE no_d_id = inittpcc.dist(d)
      AND no_w_id = :w_id
      AND ROWNUM <= 1
     RETURNING no_d_id, no_o_id BULK COLLECT INTO :d_id, :order_id;

   :ordcnt := SQL%ROWCOUNT;

   FOR i IN 1..10 LOOP
    IF (sql%BULK_rowcount(i) = 0) THEN
       no_ordr := i;
       plsql_mon_pack.print ('Delivery found no new order at district '
                || to_char(i) || ' at ' || to_char (sysdate, 'HH24:MI:SS'));
    END IF;
   END LOOP;

   dbms_lock.sleep (60);

   BEGIN
     SELECT no_o_id into order_number FROM nord
      WHERE no_d_id = no_ordr
       AND no_w_id = :w_id
       AND ROWNUM <= 1;
    EXCEPTION
      WHEN NO_DATA_FOUND THEN NULL;
   END;

   IF (SQL%FOUND) THEN
       plsql_mon_pack.print ('Delivery found new order(s) at district '
                        || to_char(no_ordr) || ' at '
                        || to_char (sysdate, 'HH24:MI:SS'));
   ELSE
       plsql_mon_pack.print ('Delivery found no new order at district '
                        || to_char(no_ordr) || ' at '
                        || to_char (sysdate, 'HH24:MI:SS'));
   END IF;

   FORALL o in 1.. :ordcnt
    UPDATE ordr SET o_carrier_id = :carrier_id
    WHERE o_id = :order_id (o)
     AND o_d_id = :d_id(o)
     AND o_w_id = :w_id
    RETURNING o_c_id BULK COLLECT INTO :o_c_id;

   FORALL o in 1.. :ordcnt
    UPDATE ordl SET ol_delivery_d = :now
    WHERE ol_w_id = :w_id
     AND ol_d_id = :d_id(o)
     AND ol_o_id = :order_id(o)
    RETURNING sum(ol_amount) BULK COLLECT INTO  :sums;

   FORALL c IN 1.. :ordcnt
    UPDATE /*+ VECTOR_READ */ cust
      SET c_balance = c_balance + :sums(c),
                c_delivery_cnt = c_delivery_cnt + 1
     WHERE c_w_id = :w_id
      AND c_d_id = :d_id(c)
      AND c_id = :o_c_id(c);
   COMMIT;
   EXIT;
   EXCEPTION
    WHEN not_serializable OR deadlock OR snapshot_too_old
    THEN
     ROLLBACK;
     :retry := :retry + 1;
   END;

  END LOOP; -- for retry
END;
```

Tkvcpnew.sql

```
-- New Order Anonymous block

 DECLARE
    idx                 PLS_INTEGER;
    dummy_local         PLS_INTEGER;
    cache_ol_cnt        PLS_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 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                          THEN s_quantity +91
                          ELSE s_quantity
                     END) - :ol_quantity(idx)
       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,
                 i_price*:ol_quantity(idx),
        CASE WHEN i_data NOT LIKE '%ORIGINAL%'
             THEN 'G'
             ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                        THEN 'G'
                        ELSE 'B'
                        END)
         END
      BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                        :ol_amount,:brand_generic;
   END u1;

   PROCEDURE u2 IS
   BEGIN
      FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                          THEN s_quantity +91
                          ELSE s_quantity
                     END) - :ol_quantity(idx)
       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,
                 i_price*:ol_quantity(idx),
        CASE WHEN i_data NOT LIKE '%ORIGINAL%'
             THEN 'G'
             ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                        THEN 'G'
                        ELSE 'B'
                        END)
         END
```

```
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u2;

PROCEDURE u3 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u3;

PROCEDURE u4 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u4;

PROCEDURE u5 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u5;

PROCEDURE u6 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u6;

PROCEDURE u7 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u7;

PROCEDURE u8 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u8;

PROCEDURE u9 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
           THEN 'G'
           ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                THEN 'G'
                ELSE 'B'
                END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u9;

PROCEDURE u10 IS
BEGIN
    FORALL idx IN 1 .. cache_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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                      THEN s_quantity +91
                      ELSE s_quantity
                    END) - :ol_quantity(idx)
      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,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                  THEN 'G'
                  ELSE 'B'
                  END)
       END
     BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                  :ol_amount,:brand_generic;
END u10;

PROCEDURE fix_items IS
  rows_lost                 PLS_INTEGER;
  max_index                 PLS_INTEGER;
  temp_index                PLS_INTEGER;
BEGIN
  idx := 1;
  rows_lost := 0;
  max_index := dummy_local;

  WHILE (max_index != cache_ol_cnt) LOOP

  WHILE (idx <= sql%rowcount AND
              sql%bulk_rowcount(idx + rows_lost) = 1)
  LOOP
    idx := idx + 1;
  END LOOP;

  temp_index := max_index;
  WHILE (temp_index >= idx + rows_lost) LOOP
    :ol_amount(temp_index + 1)      :=  :ol_amount(temp_index);
    :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);
    inittpcc.s_dist(temp_index + 1) :=  inittpcc.s_dist(temp_index);
    :brand_generic(temp_index + 1)  :=  :brand_generic(temp_index);
    temp_index := temp_index - 1;
  END LOOP;

  IF (idx + rows_lost <= cache_ol_cnt) THEN
    :i_price(idx + rows_lost)       :=  0;
    :i_name(idx + rows_lost)        :=  'NO ITEM';
    :s_quantity(idx + rows_lost)    :=  0;
    inittpcc.s_dist(idx + rows_lost) := NULL;
    :brand_generic(idx + rows_lost)  := ' ';
    :ol_amount(idx + rows_lost)     := 0;
    rows_lost := rows_lost + 1;
    max_index := max_index + 1;
  END IF;

 END LOOP;
END fix_items;

BEGIN
  LOOP BEGIN
    cache_ol_cnt := :o_ol_cnt;

    UPDATE dist 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 cust , ware
     WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = w_id
     AND  w_id = :w_id;


    INSERT INTO nord (no_o_id, no_d_id, no_w_id)
     VALUES (:o_id, :d_id, :w_id);

    INSERT INTO ordr  (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 < 6) THEN
      IF (dummy_local < 3) THEN
        IF (dummy_local = 1) THEN
          u1;
        ELSE
          u2;
        END IF;
      ELSE
        IF (dummy_local = 3) THEN
          u3;
        ELSIF (dummy_local = 4) then
          u4;
        ELSE
          u5;
        END IF;
      END IF;
    ELSE
      IF (dummy_local < 8) THEN
        IF (dummy_local = 6) THEN
          u6;
        ELSE
          u7;
        END IF;
      ELSE
        IF (dummy_local = 8) THEN
          u8;
        ELSIF (dummy_local = 9) then
          u9;
        ELSE
          u10;
        END IF;
      END IF;
    END IF;

    dummy_local := sql%rowcount;

    IF (dummy_local != cache_ol_cnt )  THEN fix_items; END IF;

    FORALL idx IN 1..dummy_local
     INSERT INTO ordl
       (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, inittpcc.idx1arr(idx), inittpcc.nulldate,
            :ol_i_id(idx), :ol_supply_w_id(idx),
            :ol_quantity(idx), :ol_amount(idx), inittpcc.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;
```

Tkvcpnew_iso7.sql

```
-- New Order Anonymous block

 DECLARE
    idx                 PLS_INTEGER;
    dummy_local             PLS_INTEGER;
    cache_ol_cnt            PLS_INTEGER;
    not_serializable        EXCEPTION;
    node_num      VARCHAR2(10);
    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 .. cache_ol_cnt
       UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                       THEN s_quantity +91
                       ELSE s_quantity
                     END) - :ol_quantity(idx)
       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,
               i_price*:ol_quantity(idx),
        CASE WHEN i_data NOT LIKE '%original%'
             THEN 'G'
             ELSE (CASE WHEN s_data NOT LIKE '%original%'
```

```
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u1;

PROCEDURE u2 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%original%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%original%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u2;

PROCEDURE u3 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%original%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%original%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u3;

PROCEDURE u4 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%original%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%original%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u4;

PROCEDURE u5 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%original%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%original%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u5;

PROCEDURE u6 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%original%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%original%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u6;

PROCEDURE u7 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%original%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%original%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u7;

PROCEDURE u8 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      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,
            i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%original%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%original%'
                THEN 'G'
                ELSE 'B'
                END)
        END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u8;
```

```
PROCEDURE u9 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                      THEN s_quantity +91
                      ELSE s_quantity
                 END) - :ol_quantity(idx)
      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,
             i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%original%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u9;

PROCEDURE u10 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE /*+ VECTOR_READ */ 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 = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                      THEN s_quantity +91
                      ELSE s_quantity
                 END) - :ol_quantity(idx)
      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,
             i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%original%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u10;

PROCEDURE fix_items IS
  rows_lost                 PLS_INTEGER;
  max_index                   PLS_INTEGER;
  temp_index                  PLS_INTEGER;
BEGIN
  idx := 1;
  rows_lost := 0;
  max_index := dummy_local;

  WHILE (max_index != cache_ol_cnt) LOOP

  WHILE (idx <= sql%rowcount AND
             sql%bulk_rowcount(idx + rows_lost) = 1)
  LOOP
    idx := idx + 1;
  END LOOP;

  temp_index := max_index;
  WHILE (temp_index >= idx + rows_lost) LOOP
    :ol_amount(temp_index + 1)      :=  :ol_amount(temp_index);
    :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);
    inittpcc.s_dist(temp_index + 1) :=  inittpcc.s_dist(temp_index);
    :brand_generic(temp_index + 1)  :=  :brand_generic(temp_index);
    temp_index := temp_index - 1;
  END LOOP;

  IF (idx + rows_lost <= cache_ol_cnt) THEN
    :i_price(idx + rows_lost)       :=  0;
    :i_name(idx + rows_lost)        :=  'NO ITEM';
    :s_quantity(idx + rows_lost)    :=  0;
    inittpcc.s_dist(idx + rows_lost) := NULL;
    :brand_generic(idx + rows_lost)  := ' ';
    :ol_amount(idx + rows_lost)     := 0;
    rows_lost := rows_lost + 1;
    max_index := max_index + 1;
  END IF;

 END LOOP;
END fix_items;

BEGIN

SELECT substr(value,1,5)
   INTO node_num
   FROM v$parameter
   WHERE name = 'instance_number';

 plsql_mon_pack.print ('New Order started at ' ||
                  to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                  node_num);

 LOOP BEGIN
   cache_ol_cnt := :o_ol_cnt;

   UPDATE dist 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 cust , ware
    WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = w_id
    AND  w_id = :w_id;


   INSERT INTO nord (no_o_id, no_d_id, no_w_id)
    VALUES (:o_id, :d_id, :w_id);

   INSERT INTO ordr  (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);


   -- copying :d_id in local variable is important - lots of instr.
   dummy_local :=  :d_id;

   IF (dummy_local < 6) THEN
    IF (dummy_local < 3) THEN
     IF (dummy_local = 1) THEN
       u1;
     ELSE
       u2;
     END IF;
    ELSE
     IF (dummy_local = 3) THEN
       u3;
     ELSIF (dummy_local = 4) then
       u4;
     ELSE
       u5;
     END IF;
    END IF;
   ELSE
    IF (dummy_local < 8) THEN
     IF (dummy_local = 6) THEN
       u6;
     ELSE
       u7;
     END IF;
    ELSE
     IF (dummy_local = 8) THEN
       u8;
     ELSIF (dummy_local = 9) then
       u9;
     ELSE
       u10;
     END IF;
    END IF;
   END IF;

   dummy_local := sql%rowcount;

               plsql_mon_pack.print('Item table read at: '|| to_char (sysdate, 'HH24:MI:SS'));
               FOR idx IN 1 .. dummy_local LOOP
     plsql_mon_pack.print('i_id = ' || to_char(:ol_i_id(idx)) || ','
                     || i_price = '|| to_char(:i_price(idx)));
               END LOOP;

   dbms_lock.sleep (30);

               plsql_mon_pack.print('Item table read at: '|| to_char (sysdate, 'HH24:MI:SS'));
               FOR idx IN 1 .. dummy_local LOOP
     plsql_mon_pack.print('i_id = ' || to_char(:ol_i_id(idx)) || ','
                     || i_price = '|| to_char(:i_price(idx)));
               END LOOP;

   IF (dummy_local != cache_ol_cnt )  THEN fix_items; END IF;

   FORALL idx IN 1..dummy_local
    INSERT INTO ordl
      (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, inittpcc.idx1arr(idx), inittpcc.nulldate,
          :ol_i_id(idx), :ol_supply_w_id(idx),
```

```
            :ol_quantity(idx), :ol_amount(idx), inittpcc.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;
```

Tokilobytes.sh

```
#!/bin/sh
# convert k, m, g, t into kilobytes, echo result
# no units means kilobytes is assumed.

amount=`echo $1 | sed -e's/^\(.*\)\(.\)/\1/'`
unit=`echo $1 | sed -e's/^.*\(.\)/\1/'`
if expr $unit = k \| $unit = K> /dev/null; then
  result=$amount
elif expr $unit = m \| $unit = M> /dev/null; then
  result=`$tpcc_bcexpr $amount \* 1024`
elif expr $unit = g \| $unit = G> /dev/null; then
  result=`$tpcc_bcexpr $amount \* 1024 \* 1024`
elif expr $unit = t \| $unit = T> /dev/null; then
  result=`$tpcc_bcexpr $amount \* 1024 \* 1024 \* 1024`
else
  #assume it's already correct.
  result=$result
fi

if test -n "$result"; then
  echo $result
else
#check if it's still valid
  echo $1
  expr $1 + 0 > /dev/null
  exit $?
fi
```

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>

#ifndef boolean
#define boolean int
#endif

#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 void TPCexit ();
extern int TPCdumpinit ();
extern void TPCdumpnew ();
extern void TPCdumppay ();
extern void TPCdumpord ();
extern void TPCdumpdel ();
extern void TPCdumpsto ();
extern void TPCdumpexit ();
extern void userlog(char* fmtp, ...);


/* 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.hh24:mi:ss"
#define SHORTDATE "dd-mm-yyyy"



#define DELRT 80.0

extern int tkvcninit ();
extern int tkvcpinit ();
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 void 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];
/* xnie - begin */
extern OCIRowid *o_rowid;
/* xnie - end */

/* 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(x)  (x  *)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));



/* bind arrays for sql */
#define OCIBNDRA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
        OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\

(progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));


/* use with callback data */
#define OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,\
              cbf_nodata,cbf_data) \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
         OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar), \
                  strlen((sqlvar)),0,(progvl),(ftype), \
                  indp,0,0,0,0,OCI_DATA_AT_EXEC)); \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
        OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)));



/* bind in/out for plsql without indicator and rcode */
#define OCIBNDPL(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen) \
    DISCARD ocierror(__FILE__,__LINE__,(errp), \
     OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
    DISCARD ocierror(__FILE__,__LINE__,(errp), \
      OCIBindByName((stmp),&(bndp),(errp),(CONST text *)(sqlvar), \
       (sb4)strlen((CONST char *)(sqlvar)), (dvoid*)(progv),(progvl),(ftype),\
           NULLP(dvoid),(alen), NULLP(ub2), 0,NULLP(ub4),OCI_DEFAULT));

/* bind  in values for plsql with indicator and rcode */
#define OCIBNDR(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
                                    (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0, \
                OCI_DEFAULT));

/* bind in/out for plsql arrays witout indicator and rcode */
#define OCIBNDPLA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen,ms,cu) \
      DISCARD ocierror(__FILE__,__LINE__, (errp), \
        OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));\
      DISCARD ocierror(__FILE__,__LINE__,(errp),\
        OCIBindByName((stmp),&(bndp),(errp),(CONST text *)(sqlvar), \
            (sb4)strlen((CONST char *) (sqlvar)),(void *)(progv), \
            (progvl),(ftype),NULL,(alen),NULL,(ms),(cu),OCI_DEFAULT));

/* bind in/out values for plsql with indicator and rcode */
#define OCIBNDRAA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode,\
```

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

#define OCIDEFINE(stmp,dfnp,errp,pos,progv,progvl,ftype)\
    OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),(ftype),\
                                    0,0,0,OCI_DEFAULT);


#define OCIDEF(stmp,dfnp,errp,pos,progv,progvl,ftype) \
    OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                                                        (dvoid**)0);\
    OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),\
                (ftype),NULL,NULL,NULL,OCI_DEFAULT); \


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

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



/* New order */

struct newinstruct {
  int w_id;
  int d_id;
  int c_id;
  int ol_i_id[15];
  int ol_supply_w_id[15];
  int ol_quantity[15];
};

struct newoutstruct {
  int terror;
  int o_id;
  int o_ol_cnt;
  char c_last[17];
  char c_credit[3];
  float c_discount;
  float w_tax;
  float d_tax;
  char o_entry_d[20];
  float total_amount;
  char i_name[15][25];
  int s_quantity[15];
  char brand_generic[15];
  float i_price[15];
  float ol_amount[15];
  char status[26];
  int retry;
};

struct newstruct {
  struct newinstruct newin;
  struct newoutstruct newout;
};


/* Payment */

struct payinstruct {
  int w_id;
  int d_id;
  int c_w_id;
  int c_d_id;
  int c_id;
  int bylastname;
  int h_amount;
  char c_last[17];
};

struct payoutstruct {
  int terror;
  char w_street_1[21];
  char w_street_2[21];
  char w_city[21];
  char w_state[3];
  char w_zip[10];
  char d_street_1[21];
  char d_street_2[21];
  char d_city[21];
  char d_state[3];
  char d_zip[10];
  int  c_id;
  char c_first[17];
  char c_middle[3];
  char c_last[17];
  char c_street_1[21];
  char c_street_2[21];
  char c_city[21];
  char c_state[3];
  char c_zip[10];
  char c_phone[17];
  char c_since[11];
  char c_credit[3];
  double c_credit_lim;
  float  c_discount;
  double c_balance;
  char c_data[201];
  char h_date[20];
  int retry;
};

struct paystruct {
  struct payinstruct payin;
  struct payoutstruct payout;
};


/* Order status */

struct ordinstruct {
  int w_id;
  int d_id;
  int c_id;
  int bylastname;
  char c_last[17];
};

struct ordoutstruct {
  int terror;
  int c_id;
  char c_last[17];
  char c_first[17];
  char c_middle[3];
  double c_balance;
  int o_id;
  char o_entry_d[20];
  int o_carrier_id;
  int o_ol_cnt;
  int ol_supply_w_id[15];
  int ol_i_id[15];
  int ol_quantity[15];
  float ol_amount[15];
  char ol_delivery_d[15][11];
  int retry;
};

struct ordstruct {
  struct ordinstruct ordin;
  struct ordoutstruct ordout;
};


/* Delivery */

struct delinstruct {
  int w_id;
  int o_carrier_id;
  double qtime;
  int in_timing_int;
  int plsqlflag;
};

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
```

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 wares> [options]
|          options: -A  load all tables
|                   -w  load ware table
|                   -d  load dist table
|                   -c  load cust table
|                   -i  load item table
|                   -s  load stok table (cluster around s_w_id)
|                   -S  load stok table (cluster around s_i_id)
|                   -h  load hist table
|                   -n  load new-order table
|                   -o <oline file> load order and order-line table
|                   -b <ware#>  beginning ware number
|                   -e <ware#>  ending ware 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 <time.h>
#include <sys/types.h>
#include "tpcc.h"

#ifdef ORA_NT
#undef boolean
#include <process.h>
#include "dpbcore.h"
# define gettime dpbtimef
# define getcpu  dpbcpu
#define lrand48() ((long)rand() <<15 | rand())
#ifdef __STDC__
#  define PROTO(args)    args
#else
#  define PROTO(args)    ()
#endif
#endif

#define DISTARR 10                      /* dist insert array size       */
#define CUSTARR 100                     /* cust insert array size       */
#define STOCARR 100                     /* stok insert array size       */
#define ITEMARR 100                     /* item insert array size       */
#define HISTARR 100           /* hist insert array size    */
#define ORDEARR 100            /* order insert array size     */
#define NEWOARR 100            /* new order insert array size  */

#define DISTFAC 10                      /* max. dist id */
#define CUSTFAC 3000                    /* max. cust id*/
#define STOCFAC 100000                  /* max. stok 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 NOT_SERIALIZABLE  8177  /* ORA-08177: transaction not serializable */
#define SNAPSHOT_TOO_OLD  1555  /* ORA-01555: snapshot too old */
#define RECOVERR -10
#define IRRECERR -20

#define SQLTXTW "INSERT INTO ware (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 dist (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 cust (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 hist (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 stok (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 ordr (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 ordr (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 ordl (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 ordl (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 nord (no_o_id, no_d_id, no_w_id) VALUES (:no_o_id,
:no_d_id, :no_w_id)"

#define SQLTXTENHA "alter session set \"_enable_hash_overflow\"=true"
#define SQLTXTDIHA "alter session set \"_enable_hash_overflow\"=false"

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];

void initperm();
void randstr();
void randdatastr();
void randnum();
void randlastname (char*, int);
int NURand();
void sysdate();

OCIEnv *tpcenv;
```

```
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;

OCIStmt *curw;
OCIStmt *curd;
OCIStmt *curc;
OCIStmt *curh;
OCIStmt *curs;
OCIStmt *curi;
OCIStmt *curo1;
OCIStmt *curo2;
OCIStmt *curol1;
OCIStmt *curol2;
OCIStmt *curno;

OCIBind *w_id_bp = (OCIBind *) 0;
OCIBind *w_name_bp = (OCIBind *) 0;
OCIBind *w_street1_bp = (OCIBind *) 0;
OCIBind *w_street2_bp = (OCIBind *) 0;
OCIBind *w_city_bp = (OCIBind *) 0;
OCIBind *w_state_bp = (OCIBind *) 0;
OCIBind *w_zip_bp = (OCIBind *) 0;
OCIBind *w_tax_bp = (OCIBind *) 0;

OCIBind *d_id_bp = (OCIBind *) 0;
OCIBind *d_w_id_bp = (OCIBind *) 0;
OCIBind *d_name_bp = (OCIBind *) 0;
OCIBind *d_street1_bp = (OCIBind *) 0;
OCIBind *d_street2_bp = (OCIBind *) 0;
OCIBind *d_city_bp = (OCIBind *) 0;
OCIBind *d_state_bp = (OCIBind *) 0;
OCIBind *d_zip_bp = (OCIBind *) 0;
OCIBind *d_tax_bp = (OCIBind *) 0;

OCIBind *c_id_bp = (OCIBind *) 0;
OCIBind *c_d_id_bp = (OCIBind *) 0;
OCIBind *c_w_id_bp = (OCIBind *) 0;
OCIBind *c_first_bp = (OCIBind *) 0;
OCIBind *c_last_bp = (OCIBind *) 0;
OCIBind *c_street1_bp = (OCIBind *) 0;
OCIBind *c_street2_bp = (OCIBind *) 0;
OCIBind *c_city_bp = (OCIBind *) 0;
OCIBind *c_state_bp = (OCIBind *) 0;
OCIBind *c_zip_bp = (OCIBind *) 0;
OCIBind *c_phone_bp = (OCIBind *) 0;
OCIBind *c_discount_bp = (OCIBind *) 0;
OCIBind *c_credit_bp = (OCIBind *) 0;
OCIBind *c_data_bp = (OCIBind *) 0;

OCIBind *i_id_bp = (OCIBind *) 0;
OCIBind *i_im_id_bp = (OCIBind *) 0;
OCIBind *i_name_bp = (OCIBind *) 0;
OCIBind *i_price_bp = (OCIBind *) 0;
OCIBind *i_data_bp = (OCIBind *) 0;

OCIBind *s_i_id_bp = (OCIBind *) 0;
OCIBind *s_w_id_bp = (OCIBind *) 0;
OCIBind *s_quantity_bp = (OCIBind *) 0;
OCIBind *s_dist_01_bp = (OCIBind *) 0;
OCIBind *s_dist_02_bp = (OCIBind *) 0;
OCIBind *s_dist_03_bp = (OCIBind *) 0;
OCIBind *s_dist_04_bp = (OCIBind *) 0;
OCIBind *s_dist_05_bp = (OCIBind *) 0;
OCIBind *s_dist_06_bp = (OCIBind *) 0;
OCIBind *s_dist_07_bp = (OCIBind *) 0;
OCIBind *s_dist_08_bp = (OCIBind *) 0;
OCIBind *s_dist_09_bp = (OCIBind *) 0;
OCIBind *s_dist_10_bp = (OCIBind *) 0;
OCIBind *s_data_bp = (OCIBind *) 0;

OCIBind *h_c_id_bp = (OCIBind *) 0;
OCIBind *h_c_d_id_bp = (OCIBind *) 0;
OCIBind *h_c_w_id_bp = (OCIBind *) 0;
OCIBind *h_d_id_bp = (OCIBind *) 0;
OCIBind *h_w_id_bp = (OCIBind *) 0;
OCIBind *h_data_bp = (OCIBind *) 0;

OCIBind *ol_o_id_bp = (OCIBind *) 0;
OCIBind *ol_d_id_bp = (OCIBind *) 0;
OCIBind *ol_w_id_bp = (OCIBind *) 0;
OCIBind *ol_i_id_bp = (OCIBind *) 0;
OCIBind *ol_number_bp = (OCIBind *) 0;
OCIBind *ol_supply_w_id_bp = (OCIBind *) 0;
OCIBind *ol_dist_info_bp = (OCIBind *) 0;
OCIBind *ol_amount_bp = (OCIBind *) 0;

OCIBind *o_id_bp = (OCIBind *) 0;
OCIBind *o_d_id_bp = (OCIBind *) 0;
OCIBind *o_w_id_bp = (OCIBind *) 0;
OCIBind *o_c_id_bp = (OCIBind *) 0;
OCIBind *o_carrier_id_bp = (OCIBind *) 0;
OCIBind *o_ol_cnt_bp  = (OCIBind *) 0;
OCIBind *o_ocnt_bp = (OCIBind *) 0;
OCIBind *o_olcnt_bp  = (OCIBind *) 0;


OCIBind *no_o_id_bp  = (OCIBind *) 0;
OCIBind *no_d_id_bp  = (OCIBind *) 0;
OCIBind *no_w_id_bp  = (OCIBind *) 0;

void 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 ware table\n");
   fprintf (stderr, "\t-d :\tload dist table\n");
   fprintf (stderr, "\t-c :\tload cust table\n");
   fprintf (stderr, "\t-i :\tload item table\n");
   fprintf (stderr, "\t-s :\tload stok table (cluster around s_w_id)\n");
   fprintf (stderr, "\t-S :\tload stok table (cluster around s_i_id)\n");
   fprintf (stderr, "\t-h :\tload hist 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 ware number\n");
   fprintf (stderr, "\t-e <ware#> :\tending ware 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,"\t   $tpcc_bench must be set to the location of the kit\n");
   fprintf (stderr, "\n");
   exit(1);
}

int sqlfile(fnam,linebuf)
char   *fnam;
text   *linebuf;
{
    FILE *fd;
    int nulpt = 0;
    char realfile[512];

    sprintf(realfile,"%s",fnam);
    fd = fopen(realfile,"r");
    if (!fd)
    {
      return (0);
    }
    while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE, fd))
    {
        nulpt = strlen((char *)linebuf);
    }
    return(nulpt);
}

void quit()
{
   OCIERROR(errhp,OCISessionEnd ( tpcsvc,errhp, tpcusr, OCI_DEFAULT));
   OCIERROR(errhp,OCIServerDetach ( tpcsrv, errhp, OCI_DEFAULT));
   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);
}

void main (argc, argv)
int argc;
char *argv[];
{
   char *uid="tpcc";
   char *pwd="tpcc";
   int scale=0;
   int i, j;
   int loop;
   int loopcount;
   int cid;
   int dwid;
   int cdid;
   int cwid;
   int sid;
   int swid;
   int olcnt;
   int nrows;
   int row;

   int w_id;
   char w_name[11];
   char w_street_1[21];
   char w_street_2[21];
   char w_city[21];
   char w_state[2];
   char w_zip[9];
   float w_tax;

   int d_id[10];
   int d_w_id[10];
```

```
char d_name[10][11];
char d_street_1[10][21];
char d_street_2[10][21];
char d_city[10][21];
char d_state[10][2];
char d_zip[10][9];
float d_tax[10];

int c_id[100];
int c_d_id[100];
int c_w_id[100];
char c_first[100][17];
char c_last[100][17];
char c_street_1[100][21];
char c_street_2[100][21];
char c_city[100][21];
char c_state[100][2];
char c_zip[100][9];
char c_phone[100][16];
char c_credit[100][2];
float c_discount[100];
char c_data[100][501];

int i_id[100];
int i_im_id[100];
int i_price[100];
char i_name[100][25];
char i_data[100][51];

int s_i_id[100];
int s_w_id[100];
int s_quantity[100];
char s_dist_01[100][24];
char s_dist_02[100][24];
char s_dist_03[100][24];
char s_dist_04[100][24];
char s_dist_05[100][24];
char s_dist_06[100][24];
char s_dist_07[100][24];
char s_dist_08[100][24];
char s_dist_09[100][24];
char s_dist_10[100][24];
char s_data[100][51];

int h_w_id[100];
int h_d_id[100];
int h_c_id[100];
char h_data[100][25];

int o_id[100];
int o_d_id[100];
int o_w_id[100];
int o_c_id[100];
int o_carrier_id[100];
int o_ol_cnt[100];

int ol_o_id[1500];
int ol_d_id[1500];
int ol_w_id[1500];
int ol_number[1500];
int ol_i_id[1500];
int ol_supply_w_id[1500];
int ol_amount[1500];
char ol_dist_info[1500][24];
int o_cnt;
int ol_cnt;

ub2 ol_o_id_len[1500];
ub2 ol_d_id_len[1500];
ub2 ol_w_id_len[1500];
ub2 ol_number_len[1500];
ub2 ol_i_id_len[1500];
ub2 ol_supply_w_id_len[1500];
ub2 ol_dist_info_len[1500];
ub2 ol_amount_len[1500];

ub4 ol_o_id_clen;
ub4 ol_d_id_clen;
ub4 ol_w_id_clen;
ub4 ol_number_clen;
ub4 ol_i_id_clen;
ub4 ol_supply_w_id_clen;
ub4 ol_dist_info_clen;
ub4 ol_amount_clen;

ub2 o_id_len[100];
ub2 o_d_id_len[100];
ub2 o_w_id_len[100];
ub2 o_c_id_len[100];
ub2 o_carrier_id_len[100];
ub2 o_ol_cnt_len[100];

ub4 o_id_clen;
ub4 o_d_id_clen;
ub4 o_w_id_clen;
ub4 o_c_id_clen;
ub4 o_carrier_id_clen;
ub4 o_ol_cnt_clen;

text stmbuf[16*1024];

int no_o_id[100];
int no_d_id[100];
int no_w_id[100];

char sdate[30];

#ifdef ORA_NT
  clock_t begin_time, end_time;
  clock_t begin_cpu, end_cpu;

  char *arg_ptr,  **end_args;
#else
  double begin_time, end_time;
  double begin_cpu, end_cpu;
  double gettime(), getcpu();

  extern int getopt();
  extern char *optarg;
  extern int optind, opterr;
  int opt;
#endif

  char            *argstr="M:AwdcisShno:b:e:j:k:g";
  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];
  char* basename;
  int status;
#ifdef ORA_NT
  char fname[100];
  FILE *logfile;
#endif /* ORA_NT */

/*---------------------------------------------------------------+
 | Parse command line -- look for scale factor.                  |
 +---------------------------------------------------------------*/

  if (argc == 1) {
    myusage ();
  }

#ifdef ORA_NT
  end_args = argv + argc;
  for (++argv; argv < end_args; )
  {
    arg_ptr = *argv++;

      if (*arg_ptr != '-')
      {
        myusage ();
      } else
      {
     switch (arg_ptr[1]) {
      case '?': myusage ();
                break;
      case 'M': scale = atoi (*argv++);
                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, *argv++);
                break;
        case 'b': bware = atoi (*argv++);
                break;
        case 'e': eware = atoi (*argv++);
                break;
        case 'j': bitem = atoi (*argv++);
                break;
        case 'k': eitem = atoi (*argv++);
                break;
        case 'g': gen = 1;
                strcpy (fname, *argv++);
                break;
        case 'l': logfile=fopen(*argv++,"w");
                break;
        default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
                fprintf (stderr, "(reached default case in getopt ())\n");
                myusage ();
      }
       }
  }


#else

  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 ();
    }
  }

# endif /* ORA_NT */

/*----------------------------------------------------------------*|
|                   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 (do_o) {
    if ((basename = getenv ("tpcc_bench")) == NULL)
    {
      fprintf (stderr, "$tpcc_bench is not set");
      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 */

    OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(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_SERVER, errhp);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
    OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
            (ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
    OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
          OCI_ATTR_PASSWORD, errhp);
    OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));

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

    fprintf (stderr, "'\nConnected to Oracle userid '%s/%s'.\n", uid, pwd);

    /* open cursors and parse statement */
    if (do_A || do_w)  {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curw), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curw, errhp, (text *)SQLTXTW,
              strlen((char *)SQLTXTW), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_d) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curd), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curd, errhp, (text *)SQLTXTD,
              strlen((char *)SQLTXTD), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_c) {
```

```
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curc), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curc, errhp, (text *)SQLTXTC,
            strlen((char *)SQLTXTC), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_h) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curh), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curh, errhp, (text *)SQLTXTH,
            strlen((char *)SQLTXTH), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_s || do_S) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curs), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curs, errhp, (text *)SQLTXTS,
            strlen((char *)SQLTXTS), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_i) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curi), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curi, errhp, (text *)SQLTXTI,
            strlen((char *)SQLTXTI), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_o) {
      int stat;
      char fname[160];
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curo1), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      DISCARD strcpy(fname,basename);
      DISCARD strcat(fname, "/");
      DISCARD strcat(fname, "benchrun/blocks/load_ordordl.sql");
      stat = sqlfile(fname, stmbuf);
      if (!stat)
      {
       fprintf (stderr, "unable to open %s \n",fname);
       quit();
       exit(1);
      }
      OCIERROR(errhp,OCIStmtPrepare(curo1, errhp, stmbuf,
            strlen((char *)stmbuf), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_n) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curno), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curno, errhp, (text *)SQLTXTNO,
            strlen((char *)SQLTXTNO), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    /* bind variables */

    /* warehouse */

    if (do_A || do_w) {
      OCIERROR(errhp, OCIBindByName(curw, &w_id_bp, errhp, (text *)(":w_id"),
strlen((":w_id")),
               (ub1 *)&(w_id), sizeof(w_id), SQLT_INT, (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_name_bp, errhp,(text *)":w_name",
strlen(":w_name"),
               (ub1 *)w_name, 11, SQLT_STR, (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_street1_bp, errhp, (text *)":w_street_1",
                                    strlen(":w_street_1"), (ub1 *)w_street_1, 21, SQLT_STR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_street2_bp, errhp, (text *)":w_street_2",
                                    strlen(":w_street_2"), (ub1 *)w_street_2, 21, SQLT_STR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_city_bp, errhp, (text *)":w_city",
                                    strlen(":w_city"), (ub1 *)w_city, 21, SQLT_STR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_state_bp, errhp, (text *)":w_state",
                                    strlen(":w_state"), (ub1 *)w_state, 2, SQLT_CHR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_zip_bp, errhp, (text *)":w_zip",
                                    strlen(":w_zip"), (ub1 *)w_zip, 9, SQLT_CHR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_tax_bp, errhp, (text *)":w_tax",
                                    strlen(":w_tax"), (ub1 *) & w_tax, sizeof(w_tax), SQLT_FLT,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* district */

    if (do_A || do_d) {
      OCIERROR(errhp, OCIBindByName(curd, &d_id_bp, errhp, (text *)":d_id",
                                    strlen(":d_id"), (ub1 *)d_id, sizeof(int), SQLT_INT,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_w_id_bp, errhp, (text *)":d_w_id",
                                    strlen(":d_w_id"), (ub1 *)d_w_id, sizeof(int), SQLT_INT,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_name_bp, errhp, (text *)":d_name",
                                    strlen(":d_name"), (ub1 *)d_name, 11, SQLT_STR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_street1_bp, errhp, (text *)":d_street_1",
                                    strlen(":d_street_1"), (ub1 *)d_street_1, 21, SQLT_STR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_street2_bp, errhp, (text *)":d_street_2",
                                    strlen(":d_street_2"), (ub1 *)d_street_2, 21, SQLT_STR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_city_bp, errhp, (text *)":d_city",
                                    strlen(":d_city"), (ub1 *)d_city, 21, SQLT_STR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_state_bp, errhp, (text *)":d_state",
               strlen(":d_state"), (ub1 *)d_state, 2, SQLT_CHR,
               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_zip_bp, errhp, (text *)":d_zip",
               strlen(":d_zip"), (ub1 *)d_zip, 9, SQLT_CHR,
               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_tax_bp, errhp, (text *)":d_tax",
               strlen(":d_tax"), (ub1 *)d_tax, sizeof(float), SQLT_FLT,
               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* customer */

    if (do_A || do_c) {
      OCIERROR(errhp, OCIBindByName(curc, &c_id_bp, errhp, (text *)":c_id",
               strlen(":c_id"), (ub1 *)c_id, sizeof(int), SQLT_INT,
               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_d_id_bp, errhp, (text *)":c_d_id",
               strlen(":c_d_id"), (ub1 *)c_d_id, sizeof(int), SQLT_INT,
               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_w_id_bp, errhp, (text *)":c_w_id",
               strlen(":c_w_id"), (ub1 *)c_w_id, sizeof(int), SQLT_INT,
               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_first_bp, errhp, (text *)":c_first",
                                    strlen(":c_first"), (ub1 *)c_first, 17, SQLT_STR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_last_bp, errhp, (text *)":c_last",
                                    strlen(":c_last"), (ub1 *)c_last, 17, SQLT_STR,
                                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_street1_bp, errhp, (text *)":c_street_1",
               strlen(":c_street_1"), (ub1 *)c_street_1, 21, SQLT_STR,
               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_street2_bp, errhp, (text *)":c_street_2",
               strlen(":c_street_2"), (ub1 *)c_street_2, 21, SQLT_STR,
               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_city_bp, errhp, (text *)":c_city",
               strlen(":c_city"), (ub1 *)c_city, 21, SQLT_STR,
               (dvoid *) 0, (ub2 *)0, (ub2 *)0,
               (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
```

```
  OCIERROR(errhp, OCIBindByName(curc, &c_state_bp, errhp, (text *)":c_state",
          strlen(":c_state"), (ub1 *)c_state, 2, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curc, &c_zip_bp, errhp, (text *)":c_zip",
          strlen(":c_zip"), (ub1 *)c_zip, 9, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curc, &c_phone_bp, errhp, (text *)":c_phone",
          strlen(":c_phone"), (ub1 *)c_phone, 16, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curc, &c_credit_bp, errhp, (text *)":c_credit",
          strlen(":c_credit"), (ub1 *)c_credit, 2, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curc, &c_discount_bp, errhp, (text *)":c_discount",
          strlen(":c_discount"), (ub1 *)c_discount, sizeof(float), SQLT_FLT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curc, &c_data_bp, errhp, (text *)":c_data",
          strlen(":c_data"), (ub1 *)c_data, 501, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

/* item */

if (do_A || do_i) {
  OCIERROR(errhp, OCIBindByName(curi, &i_id_bp, errhp, (text *)":i_id",
          strlen(":i_id"), (ub1 *)i_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curi, &i_im_id_bp, errhp, (text *)":i_im_id",
          strlen(":i_im_id"), (ub1 *)i_im_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curi, &i_name_bp, errhp, (text *)":i_name",
          strlen(":i_name"), (ub1 *)i_name, 25, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curi, &i_price_bp, errhp, (text *)":i_price",
          strlen(":i_price"), (ub1 *)i_price, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curi, &i_data_bp, errhp, (text *)":i_data",
          strlen(":i_data"), (ub1 *)i_data, 51, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

/* stock */

if (do_A || do_s || do_S) {
  OCIERROR(errhp, OCIBindByName(curs, &s_i_id_bp, errhp, (text *)":s_i_id",
          strlen(":s_i_id"), (ub1 *)s_i_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curs, &s_w_id_bp, errhp, (text *)":s_w_id",
          strlen(":s_w_id"), (ub1 *)s_w_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curs, &s_quantity_bp, errhp, (text *)":s_quantity",
          strlen(":s_quantity"), (ub1 *)s_quantity, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curs, &s_dist_01_bp, errhp, (text *)":s_dist_01",
          strlen(":s_dist_01"), (ub1 *)s_dist_01, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


  OCIERROR(errhp, OCIBindByName(curs, &s_dist_02_bp, errhp, (text *)":s_dist_02",
          strlen(":s_dist_02"), (ub1 *)s_dist_02, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


  OCIERROR(errhp, OCIBindByName(curs, &s_dist_03_bp, errhp, (text *)":s_dist_03",
          strlen(":s_dist_03"), (ub1 *)s_dist_03, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curs, &s_dist_04_bp, errhp, (text *)":s_dist_04",
          strlen(":s_dist_04"), (ub1 *)s_dist_04, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


  OCIERROR(errhp, OCIBindByName(curs, &s_dist_05_bp, errhp, (text *)":s_dist_05",
          strlen(":s_dist_05"), (ub1 *)s_dist_05, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


  OCIERROR(errhp, OCIBindByName(curs, &s_dist_06_bp, errhp, (text *)":s_dist_06",
          strlen(":s_dist_06"), (ub1 *)s_dist_06, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


  OCIERROR(errhp, OCIBindByName(curs, &s_dist_07_bp, errhp, (text *)":s_dist_07",
          strlen(":s_dist_07"), (ub1 *)s_dist_07, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


  OCIERROR(errhp, OCIBindByName(curs, &s_dist_08_bp, errhp, (text *)":s_dist_08",
          strlen(":s_dist_08"), (ub1 *)s_dist_08, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


  OCIERROR(errhp, OCIBindByName(curs, &s_dist_09_bp, errhp, (text *)":s_dist_09",
          strlen(":s_dist_09"), (ub1 *)s_dist_09, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


  OCIERROR(errhp, OCIBindByName(curs, &s_dist_10_bp, errhp, (text *)":s_dist_10",
          strlen(":s_dist_10"), (ub1 *)s_dist_10, 24, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curs, &s_data_bp, errhp, (text *)":s_data",
          strlen(":s_data"), (ub1 *)s_data, 51, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

/* history */

if (do_A || do_h) {
  OCIERROR(errhp, OCIBindByName(curh, &h_c_id_bp, errhp, (text *)":h_c_id",
          strlen(":h_c_id"), (ub1 *)h_c_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curh, &h_c_d_id_bp, errhp, (text *)":h_c_d_id",
          strlen(":h_c_d_id"), (ub1 *)h_d_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curh, &h_c_w_id_bp, errhp, (text *)":h_c_w_id",
          strlen(":h_c_w_id"), (ub1 *)h_w_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curh, &h_d_id_bp, errhp, (text *)":h_d_id",
          strlen(":h_d_id"), (ub1 *)h_d_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curh, &h_w_id_bp, errhp, (text *)":h_w_id",
          strlen(":h_w_id"), (ub1 *)h_w_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curh, &h_data_bp, errhp, (text *)":h_data",
          strlen(":h_data"), (ub1 *)h_data, 25, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
          (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

}

/* order and order_line (delivered) */

if (do_A || do_o) {

  for (i = 0; i < ORDEARR; i++ ) {
    o_id_len[i] = sizeof(int);
    o_d_id_len[i] = sizeof(int);
    o_w_id_len[i] = sizeof(int);
    o_c_id_len[i] = sizeof(int);
    o_carrier_id_len[i] = sizeof(int);
    o_ol_cnt_len[i] = sizeof(int);
```

```
    }

    OCIERROR(errhp, OCIBindByName(curo1, &ol_o_id_bp, errhp, (text *)":ol_o_id",
            strlen(":ol_o_id"), (ub1 *)ol_o_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)ol_o_id_len, (ub2 *)0,
            (ub4) 15*ORDEARR, (ub4 *)&ol_o_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &ol_d_id_bp, errhp, (text *)":ol_d_id",
            strlen(":ol_d_id"), (ub1 *)ol_d_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)ol_d_id_len, (ub2 *)0,
            (ub4) 15*ORDEARR, (ub4 *) &ol_d_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &ol_w_id_bp, errhp, (text *)":ol_w_id",
            strlen(":ol_w_id"), (ub1 *)ol_w_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)ol_w_id_len, (ub2 *)0,
            (ub4) 15*ORDEARR, (ub4 *) &ol_w_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &ol_number_bp, errhp, (text *)":ol_number",
            strlen(":ol_number"), (ub1 *)ol_number, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)ol_number_len, (ub2 *)0,
            (ub4) 15*ORDEARR, (ub4 *) &ol_number_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &ol_i_id_bp, errhp, (text *)":ol_i_id",
            strlen(":ol_i_id"), (ub1 *)ol_i_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)ol_i_id_len, (ub2 *)0,
            (ub4) 15*ORDEARR, (ub4 *) &ol_i_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &ol_supply_w_id_bp, errhp, (text
*)":ol_supply_w_id",
            strlen(":ol_supply_w_id"), (ub1 *)ol_supply_w_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)ol_supply_w_id_len, (ub2 *)0,
            (ub4) 15*ORDEARR, (ub4 *) &ol_supply_w_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &ol_dist_info_bp, errhp, (text
*)":ol_dist_info",
            strlen(":ol_dist_info"), (ub1 *)ol_dist_info, 24, SQLT_CHR,
            (dvoid *) 0, (ub2 *)ol_dist_info_len, (ub2 *)0,
            (ub4) 15*ORDEARR, (ub4 *) &ol_dist_info_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &ol_amount_bp, errhp, (text *)":ol_amount",
            strlen(":ol_amount"), (ub1 *)ol_amount, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)ol_amount_len, (ub2 *)0,
            (ub4) 15*ORDEARR, (ub4 *) &ol_amount_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &o_id_bp, errhp, (text *)":o_id",
            strlen(":o_id"), (ub1 *)o_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)o_id_len, (ub2 *)0,
            (ub4) ORDEARR, (ub4 *) &o_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &o_d_id_bp, errhp, (text *)":o_d_id",
            strlen(":o_d_id"), (ub1 *)o_d_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)o_d_id_len, (ub2 *)0,
            (ub4) ORDEARR, (ub4 *) &o_d_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &o_w_id_bp, errhp, (text *)":o_w_id",
            strlen(":o_w_id"), (ub1 *)o_w_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)o_w_id_len, (ub2 *)0,
            (ub4) ORDEARR, (ub4 *) &o_w_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &o_c_id_bp, errhp, (text *)":o_c_id",
            strlen(":o_c_id"), (ub1 *)o_c_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)o_c_id_len, (ub2 *)0,
            (ub4) ORDEARR, (ub4 *) &o_c_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &o_carrier_id_bp, errhp, (text
*)":o_carrier_id",
            strlen(":o_carrier_id"), (ub1 *)o_carrier_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)o_carrier_id_len, (ub2 *)0,
            (ub4) ORDEARR, (ub4 *) &o_carrier_id_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &o_ol_cnt_bp, errhp, (text *)":o_ol_cnt",
            strlen(":o_ol_cnt"), (ub1 *)o_ol_cnt, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)o_ol_cnt_len, (ub2 *)0,
            (ub4) ORDEARR, (ub4 *) &o_ol_cnt_clen, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &o_ocnt_bp, errhp, (text *)":order_rows",
            strlen(":order_rows"), (ub1 *)&o_cnt, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &o_olcnt_bp, errhp, (text *)":ordl_rows",
            strlen(":ordl_rows"), (ub1 *)&ol_cnt, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
  }

  /* new order */

  if (do_A || do_n) {
    OCIERROR(errhp, OCIBindByName(curno, &no_o_id_bp, errhp, (text *)":no_o_id",
            strlen(":no_o_id"), (ub1 *)no_o_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curno, &no_d_id_bp, errhp, (text *)":no_d_id",
            strlen(":no_d_id"), (ub1 *)no_d_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curno, &no_w_id_bp, errhp, (text *)":no_w_id",
            strlen(":no_w_id"), (ub1 *)no_w_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }
  }

/*---------------------------------------------------------------+
 | Initialize random number generator                            |
 +---------------------------------------------------------------*/

  srand (SEED);
#ifndef ORA_NT
  srand48 (SEED);
#endif
  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 = (float) ((lrand48 () % 2001) * 0.0001);
      randstr (w_name, 6, 10);
      randstr (w_street_1, 10, 20);
      randstr (w_street_2, 10, 20);
      randstr (w_city, 10, 20);
      randstr (str2, 2, 2);
      randnum (num9, 9);
      num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

      if (gen) {
        printf ("%d 30000000 %6.4f %s %s %s %s %s\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);

                    status = OCIStmtExecute(tpcsvc, curw, errhp, (ub4) 1, (ub4) 0,
                  (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                  (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
                    if (status != OCI_SUCCESS) {
                      fprintf (stderr, "Error at ware %d\n", loop);
                      OCIERROR(errhp, status);
                      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] = (float) ((lrand48 () % 2001) * 0.0001);
```

```
        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 3000000 %6.4f 3001 %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 {
        status = OCIStmtExecute(tpcsvc, curd, errhp, (ub4) DISTARR, (ub4) 0,
                 (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                 (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
                                  fprintf (stderr, "Aborted at ware %d, dist 1\n", dwid);
          OCIERROR(errhp, status);
          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);

    if (getenv("tpcc_hash_overflow")) {
     fprintf(stderr,"Hash overflow is enabled\n");
     OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
     sprintf ((char *) stmbuf, SQLTXTENHA);
     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);
     fprintf (stderr,"Customer loaded for horizontal partitioning\n");
    }
    else
    {
     fprintf (stderr,"Customer not loaded for horizontal partitioning\n");
    }
    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 dist cycle */
          if (cdid > DISTFAC) {
            cdid = 1;
            cwid++;                 /* shift ware 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 (lrand48 () % 10)
          c_credit[i][0] = 'G';
        else
          c_credit[i][0] = 'B';
        c_discount[i] = (float)((lrand48 () % 5001) * 0.0001);
        randstr (c_first[i], 8, 16);
        randstr (c_street_1[i], 10, 20);
        randstr (c_street_2[i], 10, 20);
        randstr (c_city[i], 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
        randnum (num16, 16);
        randstr (c_data[i], 300, 500);

        if (gen) {
          printf ("%d %d %d %s OE %s %s %s %s %s %s %s %cC 5000000 %6.4f -1000
1000 1 0 %s\n",
                  cid, cdid, cwid, c_first[i], c_last[i],
                  c_street_1[i], c_street_2[i], c_city[i], str2, num9,
                  num16, sdate, c_credit[i][0], c_discount[i], c_data[i]);
        }
        else {
          strncpy (c_state[i], str2, 2);
          strncpy (c_zip[i], num9, 9);
          strncpy (c_phone[i], num16, 16);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        status = OCIStmtExecute(tpcsvc, curc, errhp, (ub4) CUSTARR, (ub4) 0,
                 (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                 (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);

        if (status != OCI_SUCCESS) {
                                  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]);
          OCIERROR(errhp, status);
          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);
    if (getenv("tpcc_hash_overflow")) {
     fprintf(stderr,"Hash overflow is disabled\n");
     OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
     sprintf ((char *) stmbuf, SQLTXTDIHA);
     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);
    }
  }


/*-------------------------------------------------------------+
 | 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] = (lrand48 () % 10000) + 1;
        i_price[i] = ((lrand48 () % 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 {
        status = OCIStmtExecute(tpcsvc, curi, errhp, (ub4) ITEMARR, (ub4) 0,
                 (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                 (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
                                  fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
            OCIERROR(errhp, status);
            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; ) {
      /* added row < nrows condition on next line - alex.ni */
                for (i = 0; (i < STOCARR) && (row < nrows); i++, row++) {
        if (++sid > STOCFAC) {        /* cheap mod */
          sid = 1;
          swid++;
        }
        s_quantity[i] = (lrand48 () % 91) + 10;
        randstr (str24[0], 24, 24);
        randstr (str24[1], 24, 24);
        randstr (str24[2], 24, 24);
        randstr (str24[3], 24, 24);
        randstr (str24[4], 24, 24);
        randstr (str24[5], 24, 24);
        randstr (str24[6], 24, 24);
        randstr (str24[7], 24, 24);
        randstr (str24[8], 24, 24);
        randstr (str24[9], 24, 24);
        randdatastr (s_data[i], 26, 50);

        if (gen) {
          printf ("%d %d %d %s %s %s %s %s %s %s %s %s 0 0 0 %s\n",
                sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
                str24[3], str24[4], str24[5], str24[6], str24[7],
                str24[8], str24[9], s_data[i]);
        }
        else {
          s_i_id[i] = sid;
          s_w_id[i] = swid;
          strncpy (s_dist_01[i], str24[0], 24);
          strncpy (s_dist_02[i], str24[1], 24);
          strncpy (s_dist_03[i], str24[2], 24);
          strncpy (s_dist_04[i], str24[3], 24);
          strncpy (s_dist_05[i], str24[4], 24);
          strncpy (s_dist_06[i], str24[5], 24);
          strncpy (s_dist_07[i], str24[6], 24);
          strncpy (s_dist_08[i], str24[7], 24);
          strncpy (s_dist_09[i], str24[8], 24);
          strncpy (s_dist_10[i], str24[9], 24);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
                    /* Changed to STOCKARR to i - alex.ni */
        status = OCIStmtExecute(tpcsvc, curs, errhp, (ub4) i, (ub4) 0,
                 (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                 (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
                      fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0], s_i_id[0]);
            OCIERROR(errhp, status);
            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] = (lrand48 () % 91) + 10;
        randstr (str24[0], 24, 24);
        randstr (str24[1], 24, 24);
        randstr (str24[2], 24, 24);
        randstr (str24[3], 24, 24);
        randstr (str24[4], 24, 24);
        randstr (str24[5], 24, 24);
        randstr (str24[6], 24, 24);
        randstr (str24[7], 24, 24);
        randstr (str24[8], 24, 24);
        randstr (str24[9], 24, 24);
        randdatastr (s_data[i], 26, 50);

        if (gen) {
          printf ("%d %d %d %s %s %s %s %s %s %s %s %s 0 0 0 %s\n",
                sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
                str24[3], str24[4], str24[5], str24[6], str24[7],
                str24[8], str24[9], s_data[i]);
        }
        else {
          s_i_id[i] = sid;
          s_w_id[i] = swid;
          strncpy (s_dist_01[i], str24[0], 24);
          strncpy (s_dist_02[i], str24[1], 24);
          strncpy (s_dist_03[i], str24[2], 24);
          strncpy (s_dist_04[i], str24[3], 24);
          strncpy (s_dist_05[i], str24[4], 24);
          strncpy (s_dist_06[i], str24[5], 24);
          strncpy (s_dist_07[i], str24[6], 24);
          strncpy (s_dist_08[i], str24[7], 24);
          strncpy (s_dist_09[i], str24[8], 24);
          strncpy (s_dist_10[i], str24[9], 24);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        status = OCIStmtExecute(tpcsvc, curs, errhp, (ub4) STOCARR, (ub4) 0,
                 (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                 (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0], s_i_id[0]);
            OCIERROR(errhp, status);
```

```
            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 %s 1000 %s\n", cid, cdid, cwid, cdid,
                  cwid, sdate, h_data[i]);
          }
        }

        if (gen) {
          fflush (stdout);
        }
        else {
          status = OCIStmtExecute(tpcsvc, curh, errhp, (ub4) HISTARR, (ub4) 0,
                  (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                  (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
          if (status != OCI_SUCCESS) {
                                    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]);
            OCIERROR(errhp, status);
            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) {

      int batch_olcnt;

      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; ) {

        batch_olcnt = 0;

        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] = lrand48 () % 10 + 1;
          o_ol_cnt[i] = olcnt = lrand48 () % 11 + 5;

          if (gen) {
            if (cid < 2101) {
              printf ("%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 %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];
            if (cid >= 2101 ) {
              o_carrier_id[i] = 11;
            }
          }

          for (j = 0; j < o_ol_cnt[i]; j++, batch_olcnt++ ) {
            ol_i_id[batch_olcnt] = sid = lrand48 () % 100000 + 1;
            if (cid < 2101)
              ol_amount[batch_olcnt] = 0;
            else
              ol_amount[batch_olcnt] = (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[batch_olcnt], cwid,
                      ol_amount[batch_olcnt], 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[batch_olcnt], cwid,
                      ol_amount[batch_olcnt], str24[j]);
              }
            }
            else {
              ol_o_id[batch_olcnt] = cid;
              ol_d_id[batch_olcnt] = cdid;
              ol_w_id[batch_olcnt] = cwid;
              ol_number[batch_olcnt] = j + 1;
              ol_supply_w_id[batch_olcnt] = cwid;
              strncpy (ol_dist_info[batch_olcnt], str24[j], 24);
            }
          }
          if (gen) {
            fflush (olfp);
          }
        }

        o_cnt =  ORDEARR;
        ol_cnt =  batch_olcnt;

        for (j = 0; j < batch_olcnt; j++) {
          ol_o_id_len[j] = sizeof(int);
          ol_d_id_len[j] = sizeof(int);
          ol_w_id_len[j] = sizeof(int);
          ol_number_len[j] = sizeof(int);
          ol_i_id_len[j] = sizeof(int);
```

```
      ol_supply_w_id_len[j] = sizeof(int);
      ol_dist_info_len[j] = 24;
      ol_amount_len[j] = sizeof(int);
    }
    for (j = batch_olcnt; j < 15*ORDEARR; j++) {
      ol_o_id_len[j] = 0;
      ol_d_id_len[j] = 0;
      ol_w_id_len[j] = 0;
      ol_number_len[j] = 0;
      ol_i_id_len[j] = 0;
      ol_supply_w_id_len[j] = 0;
      ol_dist_info_len[j] = 0;
      ol_amount_len[j] = 0;
    }

    o_id_clen = ORDEARR;
    o_d_id_clen = ORDEARR;
    o_w_id_clen = ORDEARR;
    o_c_id_clen = ORDEARR;
    o_carrier_id_clen = ORDEARR;
    o_ol_cnt_clen = ORDEARR;

    ol_o_id_clen = batch_olcnt;
    ol_d_id_clen = batch_olcnt;
    ol_w_id_clen = batch_olcnt;
    ol_number_clen = batch_olcnt;
    ol_i_id_clen = batch_olcnt;
    ol_supply_w_id_clen = batch_olcnt;
    ol_dist_info_clen = batch_olcnt;
    ol_amount_clen = batch_olcnt;

    OCIERROR(errhp, OCIStmtExecute(tpcsvc, curo1, errhp, (ub4) 1, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS ));

    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 {
      status = OCIStmtExecute(tpcsvc, curno, errhp, (ub4) NEWOARR, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
        fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n", cwid,
cdid, cid + 2100);
        OCIERROR(errhp, status);
        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);

}

void 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 = lrand48 () % i;
    temp = randperm3000[i - 1];
    randperm3000[i - 1] = randperm3000[pos];
    randperm3000[pos] = temp;
  }
}

void randstr (str, x, y)
char *str;
int x;
int y;
{
  int i, j;
  int len;

  len = (lrand48 () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = lrand48 () % 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';

}

void randdatastr (str, x, y)
char *str;
int x;
int y;
{
  int i, j;
  int len;
  int pos;

  len = (lrand48 () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = lrand48 () % 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 ((lrand48 () % 10) == 0) {
    pos = (lrand48 () % (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';
  }
}

void randnum (str, len)
char *str;
int len;
{
  int i;

  for (i = 0; i < len; i++)
    str[i] = (char) (lrand48 () % 10 + '0');
  str[len] = '\0';

}

void 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]);
}

int 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);

}

void sysdate (sdate)
char *sdate;
{
  time_t tp;
  struct tm *tmptr;

  time (&tp);
  tmptr = localtime (&tp);
  strftime (sdate, 29, "%d-%b-%Y", tmptr);
}

int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
 text errbuf[512];
 sb4 errcode;
 sb4 lstat;
 ub4 recno=2;

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

Unix_audit_env.sh

```
BENCH_HOME=/xnieq/tpccQ
SRCHOME=$BENCH_HOME
ORACLE_SID=tpccQ
export ORACLE_SID BENCH_HOME SRCHOME
```

Upd_date.sh

```
#!/bin/sh
#
# $Header: upd_date.sh 01-jun-98.19:12:54 skareenh Exp $
#
# upd_date.sh
#
#  Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#    NAME
#      upd_date.sh
#
#    DESCRIPTION
#                  Insert start time into tpcc_audit_tab.
#
#    NOTES
#      <other useful comments, qualifications, etc.>
#
#    MODIFIED   (MM/DD/YY)
#    skareenh    06/01/98 - Creation
#


sqlplus tpcc/tpcc <<!
  delete from tpcc_audit_tab;
  insert into tpcc_audit_tab (starttime)
    select sysdate from dual;
  commit;
  quit;
!
```

Updateordrordl.sh

```
#!/usr/bin/sh
$tpcc_sqlplus $tpcc_user_pass <<!

declare
  wid  number;
  did  number;
  oid  number;
begin
  for wid in ${5}..${7} loop
    for did in 1..10 loop
      for oid in 1..2100 loop
        update ordr set o_entry_d=sysdate
                where o_w_id = wid and o_d_id = did and o_id = oid;
```

```
        update ordl set ol_delivery_d = sysdate
                where ol_w_id = wid and ol_d_id = did and ol_o_id = oid;
        commit;
      end loop;
    end loop;
  end loop;
end;
/

exit 0
!
```

Views.sql

```
connect tpcc/tpcc;
set echo on;

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 cust c, ware 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 dist d, ware 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 stok s, item i
 where i.i_id = s.s_i_id;

set echo off;
```

Appendix C Tunable Parameters

The HP-UX operating system tunable parameters employed to generate the kernel for the hp server rx5670 and the 7 HP 9000 Model hp server rp2470 clients are listed below. Included as well are the Oracle10i Database Standard Edition and TUXEDO 8.0 parameters.

C.1 HP-UX Configuration - Clients

Config/Client2/ostune.ver

```
****************************************
* $Source: /usr/local/kcs/sys.ROSE_800/filesets.info/CORE-KRN/RCS/generic,v $
* $Revision: 1.3.106.2 $        $Author: kcs $
* $State: Exp $                 $Locker: CRT $
* $Date: 97/07/12 21:51:58 $
*
****************************************
* Additional drivers required in every machine-type to create a complete
* system file during cold install.  This list is every driver that the
* master.d/ files do not force on the system or is not identifiable by
* ioscan.
* Other CPU-type specific files can exist for their special cases.
* see create_sysfile (1m).
****************************************
*
* Drivers/Subsystems
audio
sdisk
sctl
asio0
c720
*graph3
cdfs
nfs_core
*nfs_client
*nfs_server
*btlan5
*fcgsc_lan
*fc_arp
dlpi
inet
uipc
tun
telm
tels
netdiag1
nms
vxbase
lvm
lv
hpstreams
clone
strlog
sad
echo
sc
timod
tirdwr
pipedev
pipemod
ffs
ldterm
ptem
pts
ptm
pckt
sba
lba
sapic
diag1
superio
SCentIf
side
hcd
usbd
hub
hid
autofsc
nfsm
*btlan6
btlan3
maclan
diag2
dmem
dev_config
beep
token_arp
rpcmod
dump lvol

STRMSGSZ       65535
bufpages       8192
create_fastlinks    1
dbc_min_pct         0
dbc_max_pct         0
default_disk_ir     1
fs_async            1

maxfiles       2048
maxfiles_lim    2048
maxdsiz        0x80000000
maxssiz        0X10000000
maxswapchunks                  4096
maxuprc        200
nproc          (100+MAXUPRC)
max_thread_proc 1050
nkthread       15000

msgmni         (NKTHREAD)
msgtql         (NKTHREAD)
msgseg         (MSGMNI*2)
msgssz         512
msgmap         (MSGSEG)
msgmax         32768
msgmnb         (MSGMAX*2)
nfile          (NKTHREAD+NPROC*5)
ninode         (NKTHREAD+NPROC*5)
nflocks        4000
npty           128
nstrpty        200

semmni         32
semmns         NKTHREAD
semmnu         (SEMMNS)
semume         4
semvmx         40960

shmmax         0X40000000
shmmni         16
shmseg         16

swapmem_on      0
unlockable_mem  1
```

C.2 HP-UX Configurtion – Server

Config/Server/ostune.ver

```
* Generated file. Do not edit
****************************************
* Source: /ux/core/kern/filesets.info/CORE-KRN/generic
* @(#)B.11.11_LR
*
****************************************
* Additional drivers required in every machine-type to create a complete
* system file during cold install.  This list is every driver that the
* master.d/ files do not force on the system or is not identifiable by
* ioscan.
* Other CPU-type specific files can exist for their special cases.
* see create_sysfile (1m).
****************************************
*
* Drivers/Subsystems
sba
lba
asio0
btlan
c720
sdisk
sctl
td
stape
gelan
cdfs
cxperf
diag0
diag1
diag2
```

```
dmem
dev_config
iomem
nfs_core
nfs_client
nfs_server
maclan
dlpi
token_arp
inet
uipc
tun
telm
tels
netdiag1
nms
hpstreams
clone
strlog
sad
echo
sc
timod
tirdwr
pipedev
pipemod
ffs
ldterm
ptem
pts
ptm
pckt
vxfs
vxportal
lvm
lv
nfsm
rpcmod
autofsc
cachefsc
cifs
fddi4
GSCtoPCI
iop_drv
bs_osm
hd_fabric
tape2
olar_psm
olar_psm_if
dev_olar
STRMSGSZ                65535
nstrpty                 60
dump lvol

asyncdsk
* coke
*
maxfiles          2048
maxfiles_lim      2048
nflocks           2048
fs_async          0
bufpages          25000
eqmemsize         10000
maxusers          1024
maxuprc           3084
max_async_ports   3200
nproc             3200
nhtbl_scale       1
maxswapchunks     16384
swchunk           16384
nfile             700000
ninode            30000
npty              10
shmmni            104
semmni            10240
semmns            20480
semmnu            2548
semvmx            32768
shmmax            0x4000000000
shmseg            16
maxssiz           0x10000000
maxdsiz           0x30000000
maxssiz_64bit     0x300000000
maxdsiz_64bit     0x300000000
timezone          480
maxvgs            128
msgmax            32768
msgmnb            32768
msgmni            50
msgseg            7168
msgssz            8
msgtql            256
unlockable_mem    1
swapmem_on        0
```

C.3 Oracle10i Database Standard Edition Parameters

Config/Server/dbtune.ver

```
_db_file_noncontig_mblock_read_count = 1
db_cache_advice                 = off
statistics_level                = basic
_log_simultaneous_copies        = 64
log_parallelism                                 = 4
_disable_incremental_checkpoints  = true
control_files                   = (?/dbs/tpcc_disks/control01, ?/dbs/tpcc_disks/control02)
db_block_checksum               = false
_lgwr_async_io                  = false
timed_statistics                = false
db_writer_processes             = 8
parallel_max_servers            = 700
cpu_count                       = 64
disk_asynch_io                  = TRUE
lock_sga                        = false
compatible                      = 9.2.0.1.0
db_name                         = tpcc
instance_name                   = tpcc
db_files                        = 390
db_block_size                   = 2048
db_16k_cache_size               = 67000M
db_8k_cache_size                = 800M
db_cache_size                   = 4500M
db_keep_cache_size              = 162000M
db_recycle_cache_size           = 3500M
_db_block_max_dirty_target      = 90000000
_db_writer_chunk_writes         = 250
_db_writer_max_writes           = 250
dml_locks                       = 500
enqueue_resources               = 60000
hash_join_enabled               = FALSE
log_archive_start               = FALSE
log_checkpoint_timeout          = 0
log_checkpoint_interval         = 1000000000
log_checkpoints_to_alert        = TRUE
log_buffer                      = 33554432
open_cursors                    = 2000
processes                       = 2000
sessions                        = 2000
transactions                    = 6000
shared_pool_size                = 700M
shared_pool_reserved_size       = 100M
cursor_space_for_time           = TRUE
transaction_auditing            = FALSE
replication_dependency_tracking   = FALSE
db_block_checking               = FALSE
max_dump_file_size              = 10K
transactions_per_rollback_segment = 1
max_rollback_segments           = 1200
rollback_segments               = (large_rbs,s1,s2,s3,s4,s5,s6,s7,s8,s9,
s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,
s20,s21,s22,s23,s24,s25,s26,s27,s28,s29,
s30,s31,s32,s33,s34,s35,s36,s37,s38,s39,
s40,s41,s42,s43,s44,s45,s46,s47,s48,s49,
s50,s51,s52,s53,s54,s55,s56,s57,s58,s59,
s60,s61,s62,s63,s64,s65,s66,s67,s68,s69,
s70,s71,s72,s73,s74,s75,s76,s77,s78,s79,
s80,s81,s82,s83,s84,s85,s86,s87,s88,s89,
s90,s91,s92,s93,s94,s95,s96,s97,s98,s99,
s100,s101,s102,s103,s104,s105,s106,s107,s108,s109,
s110,s111,s112,s113,s114,s115,s116,s117,s118,s119,
s120,s121,s122,s123,s124,s125,s126,s127,s128,s129,
s130,s131,s132,s133,s134,s135,s136,s137,s138,s139,
s140,s141,s142,s143,s144,s145,s146,s147,s148,s149,
s150,s151,s152,s153,s154,s155,s156,s157,s158,s159,
s160,s161,s162,s163,s164,s165,s166,s167,s168,s169,
s170,s171,s172,s173,s174,s175,s176,s177,s178,s179,
s180,s181,s182,s183,s184,s185,s186,s187,s188,s189,
s190,s191,s192,s193,s194,s195,s196,s197,s198,s199,
s200)
rollback_segments          = (s201,s202,s203,s204,s205,s206,s207,s208,s209,
s210,s211,s212,s213,s214,s215,s216,s217,s218,s219,
s220,s221,s222,s223,s224,s225,s226,s227,s228,s229,
s230,s231,s232,s233,s234,s235,s236,s237,s238,s239,
s240,s241,s242,s243,s244,s245,s246,s247,s248,s249,
s250,s251,s252,s253,s254,s255,s256,s257,s258,s259,
s260,s261,s262,s263,s264,s265,s266,s267,s268,s269,
s270,s271,s272,s273,s274,s275,s276,s277,s278,s279,
s280,s281,s282,s283,s284,s285,s286,s287,s288,s289,
s290,s291,s292,s293,s294,s295,s296,s297,s298,s299,
s300,s301,s302,s303,s304,s305,s306,s307,s308,s309,
s310,s311,s312,s313,s314,s315,s316,s317,s318,s319,
s320,s321,s322,s323,s324,s325,s326,s327,s328,s329,
```

```
s330,s331,s332,s333,s334,s335,s336,s337,s338,s339,
s340,s341,s342,s343,s344,s345,s346,s347,s348,s349,
s350,s351,s352,s353,s354,s355,s356,s357,s358,s359,
s360,s361,s362,s363,s364,s365,s366,s367,s368,s369,
s370,s371,s372,s373,s374,s375,s376,s377,s378,s379,
s380,s381,s382,s383,s384,s385,s386,s387,s388,s389,
s390,s391,s392,s393,s394,s395,s396,s397,s398,s399,
s400)
rollback_segments       = (s401,s402,s403,s404,s405,s406,s407,s408,s409,
s410,s411,s412,s413,s414,s415,s416,s417,s418,s419,
s420,s421,s422,s423,s424,s425,s426,s427,s428,s429,
s430,s431,s432,s433,s434,s435,s436,s437,s438,s439,
s440,s441,s442,s443,s444,s445,s446,s447,s448,s449,
s450,s451,s452,s453,s454,s455,s456,s457,s458,s459,
s460,s461,s462,s463,s464,s465,s466,s467,s468,s469,
s470,s471,s472,s473,s474,s475,s476,s477,s478,s479,
s480,s481,s482,s483,s484,s485,s486,s487,s488,s489,
s490,s491,s492,s493,s494,s495,s496,s497,s498,s499,
s500,s501,s502,s503,s504,s505,s506,s507,s508,s509,
s510,s511,s512,s513,s514,s515,s516,s517,s518,s519,
s520,s521,s522,s523,s524,s525,s526,s527,s528,s529,
s530,s531,s532,s533,s534,s535,s536,s537,s538,s539,
s540,s541,s542,s543,s544,s545,s546,s547,s548,s549,
s550,s551,s552,s553,s554,s555,s556,s557,s558,s559,
s560,s561,s562,s563,s564,s565,s566,s567,s568,s569,
s570,s571,s572,s573,s574,s575,s576,s577,s578,s579,
s580,s581,s582,s583,s584,s585,s586,s587,s588,s589,
s590,s591,s592,s593,s594,s595,s596,s597,s598,s599,
s600)
rollback_segments       = (s601,s602,s603,s604,s605,s606,s607,s608,s609,
s610,s611,s612,s613,s614,s615,s616,s617,s618,s619,
s620,s621,s622,s623,s624,s625,s626,s627,s628,s629,
s630,s631,s632,s633,s634,s635,s636,s637,s638,s639,
s640,s641,s642,s643,s644,s645,s646,s647,s648,s649,
s650,s651,s652,s653,s654,s655,s656,s657,s658,s659,
s660,s661,s662,s663,s664,s665,s666,s667,s668,s669,
s670,s671,s672,s673,s674,s675,s676,s677,s678,s679,
s680,s681,s682,s683,s684,s685,s686,s687,s688,s689,
s690,s691,s692,s693,s694,s695,s696,s697,s698,s699,
s700,s701,s702,s703,s704,s705,s706,s707,s708,s709,
s710,s711,s712,s713,s714,s715,s716,s717,s718,s719,
s720,s721,s722,s723,s724,s725,s726,s727,s728,s729,
s730,s731,s732,s733,s734,s735,s736,s737,s738,s739,
s740,s741,s742,s743,s744,s745,s746,s747,s748,s749,
s750,s751,s752,s753,s754,s755,s756,s757,s758,s759,
s760,s761,s762,s763,s764,s765,s766,s767,s768,s769,
s770,s771,s772,s773,s774,s775,s776,s777,s778,s779,
s780,s781,s782,s783,s784,s785,s786,s787,s788,s789,
s790,s791,s792,s793,s794,s795,s796,s797,s798,s799,
s800)
rollback_segments       = (s801,s802,s803,s804,s805,s806,s807,s808,s809,
s810,s811,s812,s813,s814,s815,s816,s817,s818,s819,
s820,s821,s822,s823,s824,s825,s826,s827,s828,s829,
s830,s831,s832,s833,s834,s835,s836,s837,s838,s839,
s840,s841,s842,s843,s844,s845,s846,s847,s848,s849,
s850,s851,s852,s853,s854,s855,s856,s857,s858,s859,
s860,s861,s862,s863,s864,s865,s866,s867,s868,s869,
s870,s871,s872,s873,s874,s875,s876,s877,s878,s879,
s880,s881,s882,s883,s884,s885,s886,s887,s888,s889,
s890,s891,s892,s893,s894,s895,s896,s897,s898,s899,
s900,s901,s902,s903,s904,s905,s906,s907,s908,s909,
s910,s911,s912,s913,s914,s915,s916,s917,s918,s919,
s920,s921,s922,s923,s924,s925,s926,s927,s928,s929,
s930,s931,s932,s933,s934,s935,s936,s937,s938,s939,
s940,s941,s942,s943,s944,s945,s946,s947,s948,s949,
s950,s951,s952,s953,s954,s955,s956,s957,s958,s959,
s960,s961,s962,s963,s964,s965,s966,s967,s968,s969,
s970,s971,s972,s973,s974,s975,s976,s977,s978,s979,
s980,s981,s982,s983,s984,s985,s986,s987,s988,s989,
s990,s991,s992,s993,s994,s995,s996,s997,s998,s999,s1000)
rollback_segments     = (s1001,s1002,s1003,s1004,s1005,s1006,s1007,s1008,s1009,
s1010,s1011,s1012,s1013,s1014,s1015,s1016,s1017,s1018,s1019,
s1020,s1021,s1022,s1023,s1024,s1025,s1026,s1027,s1028,s1029,
s1030,s1031,s1032,s1033,s1034,s1035,s1036,s1037,s1038,s1039,
s1040,s1041,s1042,s1043,s1044,s1045,s1046,s1047,s1048,s1049,
s1050,s1051,s1052,s1053,s1054,s1055,s1056,s1057,s1058,s1059,
s1060,s1061,s1062,s1063,s1064,s1065,s1066,s1067,s1068,s1069,
s1070,s1071,s1072,s1073,s1074,s1075,s1076,s1077,s1078,s1079,
s1080,s1081,s1082,s1083,s1084,s1085,s1086,s1087,s1088,s1089,
s1090,s1091,s1092,s1093,s1094,s1095,s1096,s1097,s1098,s1099,
s1100,s1101,s1102,s1103,s1104,s1105,s1106,s1107,s1108,s1109,
s1110,s1111,s1112,s1113,s1114,s1115,s1116,s1117,s1118,s1119,
s1120,s1121,s1122,s1123,s1124,s1125,s1126,s1127,s1128,s1129,
s1130,s1131,s1132,s1133,s1134,s1135,s1136,s1137,s1138,s1139,
s1140,s1141,s1142,s1143,s1144,s1145,s1146,s1147,s1148,s1149,
s1150,s1151,s1152,s1153,s1154,s1155,s1156,s1157,s1158,s1159,
s1160,s1161,s1162,s1163,s1164,s1165,s1166,s1167,s1168,s1169,
s1170,s1171,s1172,s1173,s1174,s1175,s1176,s1177,s1178,s1179,
s1180,s1181,s1182,s1183,s1184,s1185,s1186,s1187,s1188,s1189,
s1190,s1191,s1192,s1193,s1194,s1195,s1196,s1197,s1198,s1199,
s1200)
#rollback_segments     = (s1201,s1202,s1203,s1204,s1205,s1206,s1207,s1208,s1209,
#s1210,s1211,s1212,s1213,s1214,s1215,s1216,s1217,s1218,s1219,
#s1220,s1221,s1222,s1223,s1224,s1225,s1226,s1227,s1228,s1229,
#s1230,s1231,s1232,s1233,s1234,s1235,s1236,s1237,s1238,s1239,
#s1240,s1241,s1242,s1243,s1244,s1245,s1246,s1247,s1248,s1249,
#s1250,s1251,s1252,s1253,s1254,s1255,s1256,s1257,s1258,s1259,
#s1260,s1261,s1262,s1263,s1264,s1265,s1266,s1267,s1268,s1269,
#s1270,s1271,s1272,s1273,s1274,s1275,s1276,s1277,s1278,s1279,
#s1280,s1281,s1282,s1283,s1284,s1285,s1286,s1287,s1288,s1289,
#s1290,s1291,s1292,s1293,s1294,s1295,s1296,s1297,s1298,s1299,
#s1300,s1301,s1302,s1303,s1304,s1305,s1306,s1307,s1308,s1309,
#s1310,s1311,s1312,s1313,s1314,s1315,s1316,s1317,s1318,s1319,
#s1320,s1321,s1322,s1323,s1324,s1325,s1326,s1327,s1328,s1329,
#s1330,s1331,s1332,s1333,s1334,s1335,s1336,s1337,s1338,s1339,
#s1340,s1341,s1342,s1343,s1344,s1345,s1346,s1347,s1348,s1349,
#s1350,s1351,s1352,s1353,s1354,s1355,s1356,s1357,s1358,s1359,
#s1360,s1361,s1362,s1363,s1364,s1365,s1366,s1367,s1368,s1369,
#s1370,s1371,s1372,s1373,s1374,s1375,s1376,s1377,s1378,s1379,
#s1380,s1381,s1382,s1383,s1384,s1385,s1386,s1387,s1388,s1389,
#s1390,s1391,s1392,s1393,s1394,s1395,s1396,s1397,s1398,s1399,
#s1400)
#rollback_segments     = (s1401,s1402,s1403,s1404,s1405,s1406,s1407,s1408,s1409,
#s1410,s1411,s1412,s1413,s1414,s1415,s1416,s1417,s1418,s1419,
#s1420,s1421,s1422,s1423,s1424,s1425,s1426,s1427,s1428,s1429,
#s1430,s1431,s1432,s1433,s1434,s1435,s1436,s1437,s1438,s1439,
#s1440,s1441,s1442,s1443,s1444,s1445,s1446,s1447,s1448,s1449,
#s1450,s1451,s1452,s1453,s1454,s1455,s1456,s1457,s1458,s1459,
#s1460,s1461,s1462,s1463,s1464,s1465,s1466,s1467,s1468,s1469,
#s1470,s1471,s1472,s1473,s1474,s1475,s1476,s1477,s1478,s1479,
#s1480,s1481,s1482,s1483,s1484,s1485,s1486,s1487,s1488,s1489,
#s1490,s1491,s1492,s1493,s1494,s1495,s1496,s1497,s1498,s1499,
#s1500,s1501,s1502,s1503,s1504,s1505,s1506,s1507,s1508,s1509,
#s1510,s1511,s1512,s1513,s1514,s1515,s1516,s1517,s1518,s1519,
#s1520,s1521,s1522,s1523,s1524,s1525,s1526,s1527,s1528,s1529,
#s1530,s1531,s1532,s1533,s1534,s1535,s1536,s1537,s1538,s1539,
#s1540,s1541,s1542,s1543,s1544,s1545,s1546,s1547,s1548,s1549,
#s1550,s1551,s1552,s1553,s1554,s1555,s1556,s1557,s1558,s1559,
#s1560,s1561,s1562,s1563,s1564,s1565,s1566,s1567,s1568,s1569,
#s1570,s1571,s1572,s1573,s1574,s1575,s1576,s1577,s1578,s1579,
#s1580,s1581,s1582,s1583,s1584,s1585,s1586,s1587,s1588,s1589,
#s1590,s1591,s1592,s1593,s1594,s1595,s1596,s1597,s1598,s1599,
#s1600)
#rollback_segments     = (s1601,s1602,s1603,s1604,s1605,s1606,s1607,s1608,s1609,
#s1610,s1611,s1612,s1613,s1614,s1615,s1616,s1617,s1618,s1619,
#s1620,s1621,s1622,s1623,s1624,s1625,s1626,s1627,s1628,s1629,
#s1630,s1631,s1632,s1633,s1634,s1635,s1636,s1637,s1638,s1639,
#s1640,s1641,s1642,s1643,s1644,s1645,s1646,s1647,s1648,s1649,
#s1650,s1651,s1652,s1653,s1654,s1655,s1656,s1657,s1658,s1659,
#s1660,s1661,s1662,s1663,s1664,s1665,s1666,s1667,s1668,s1669,
#s1670,s1671,s1672,s1673,s1674,s1675,s1676,s1677,s1678,s1679,
#s1680,s1681,s1682,s1683,s1684,s1685,s1686,s1687,s1688,s1689,
#s1690,s1691,s1692,s1693,s1694,s1695,s1696,s1697,s1698,s1699,
#s1700,s1701,s1702,s1703,s1704,s1705,s1706,s1707,s1708,s1709,
#s1710,s1711,s1712,s1713,s1714,s1715,s1716,s1717,s1718,s1719,
#s1720,s1721,s1722,s1723,s1724,s1725,s1726,s1727,s1728,s1729,
#s1730,s1731,s1732,s1733,s1734,s1735,s1736,s1737,s1738,s1739,
#s1740,s1741,s1742,s1743,s1744,s1745,s1746,s1747,s1748,s1749,
#s1750,s1751,s1752,s1753,s1754,s1755,s1756,s1757,s1758,s1759,
#s1760,s1761,s1762,s1763,s1764,s1765,s1766,s1767,s1768,s1769,
#s1770,s1771,s1772,s1773,s1774,s1775,s1776,s1777,s1778,s1779,
#s1780,s1781,s1782,s1783,s1784,s1785,s1786,s1787,s1788,s1789,
#s1790,s1791,s1792,s1793,s1794,s1795,s1796,s1797,s1798,s1799,
#s1800)
```

C.4 Tuxedo UBBconfig

Config/Client2/tmcfg.ver

```
# This is a UBBconfig for a client1-server configuration.
#
# This UBBconfig requires settings for:
# SERVER_NAME CLIENT_NAME MASTER_NAME SERVER_ADDR CLIENT_ADDR
NODE_NAMES
# TLISTEN_PORT TBRIDGE_PORT
# In addition, it requires setting the things all UBBconfig.gens need:
#               IPCKEY                  some decent IPCKEY, should be different for each
config
#               ROOTDIR
#               TUXCONFIG
#               APPDIR
#               ULOGDIR
#

#------------------------------------------------------------------
*RESOURCES
#------------------------------------------------------------------
                IPCKEY      40001
                PERM        0666
                MASTER      client1

  MAXACCESSERS          13050           # 1024 or more
  MAXGTT    1024
```

```
   MAXSERVERS              70
   MAXSERVICES             335   # MAXSERVERS * #-of-services-each-server + 10( for BBL)
   MODEL       SHM
   LDBAL       Y

# During benchmark, don't want to scan too often.  In particular, while
# the client1s are stabilizing in virtual memory, we don't want to sanity
# scan; and if we do sanity scan, we want large timeouts, since the BRIDGE
# the BBL, the DBBL, and the client1s aren't getting much CPU time during that
# period.  Current settings:
#                 *  scan servers every 5 minutes (maximum allowed by TUXEDO);
#                 *  wait 1 minute for sanity  responses (maximum allowed by TUXEDO);
#                 *  scan all the BBLs from DBBL every 30 minutes (want one scan in the
#                    audited results);
#                 *  timeout a blocking call after 5 minutes (the maximum).
                  SCANUNIT    60
                  SANITYSCAN                  5
                  DBBLWAIT    1
                  BBLQUERY    30
                  BLOCKTIME 5
#
#--------------------------------------------------------------------
*MACHINES
#--------------------------------------------------------------------
DEFAULT:
            TUXCONFIG="/project/iti/confs/TUXconfig.client1"
                              ROOTDIR="/project/iti"
                              APPDIR="/project/tpcc/bin"
                              ULOGPFX="/tmp/TUXEDO_LOG"

# for debugging, put both into the same log on the same machine
#                             ULOGPFX="/home/iti/confs/tpcc/ULOG"
# but for a big run, need some space, and want them local to the
# machine rather than across the net.

# Leave TUXCONFIG alone on the MASTER machine; over-ride for each
# other machine?
client1           LMID=client1
      TUXCONFIG="/project/iti/confs/TUXconfig.client1"
#--------------------------------------------------------------------
*GROUPS
#--------------------------------------------------------------------
group1          LMID=client1
                GRPNO=1
group2          LMID=client1
                GRPNO=2
group3          LMID=client1
                GRPNO=3
group4          LMID=client1
                GRPNO=4
group5          LMID=client1
                GRPNO=5
group6          LMID=client1
                GRPNO=6
group7          LMID=client1
                GRPNO=7
group8          LMID=client1
                GRPNO=8
group9          LMID=client1
                GRPNO=9
group10           LMID=client1
                GRPNO=10
group11           LMID=client1
                GRPNO=11
group12           LMID=client1
                GRPNO=12
group13           LMID=client1
                GRPNO=13

#--------------------------------------------------------------------
#--------------------------------------------------------------------

#--------------------------------------------------------------------
*SERVERS
#--------------------------------------------------------------------
#
# "--" is application-specific arguments to be passed to server
# "-n" is designed to specify server-id

service SRVGRP=group1
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n1"
      RQADDR=tpcc_1 SRVID=1

service SRVGRP=group1
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n2"
      RQADDR=tpcc_2 SRVID=2

service SRVGRP=group1
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n3"
      RQADDR=tpcc_3 SRVID=3

service SRVGRP=group1
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n4"
      RQADDR=tpcc_4 SRVID=4

service SRVGRP=group1
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n5"
      RQADDR=tpcc_5 SRVID=5

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n6"
      RQADDR=tpcc_6 SRVID=6

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n7"
      RQADDR=tpcc_7 SRVID=7

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n8"
      RQADDR=tpcc_8 SRVID=8

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n9"
      RQADDR=tpcc_9 SRVID=9

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n10"
      RQADDR=tpcc_10 SRVID=10

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n11"
      RQADDR=tpcc_11 SRVID=11

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n12"
      RQADDR=tpcc_12 SRVID=12

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n13"
      RQADDR=tpcc_13 SRVID=13

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n14"
      RQADDR=tpcc_14 SRVID=14

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n15"
      RQADDR=tpcc_15 SRVID=15

service SRVGRP=group4
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n16"
      RQADDR=tpcc_16 SRVID=16

service SRVGRP=group4
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n17"
      RQADDR=tpcc_17 SRVID=17

service SRVGRP=group4
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n18"
      RQADDR=tpcc_18 SRVID=18

service SRVGRP=group4
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n19"
      RQADDR=tpcc_19 SRVID=19

service SRVGRP=group4
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n20"
      RQADDR=tpcc_20 SRVID=20

service SRVGRP=group5
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n21"
      RQADDR=tpcc_21 SRVID=21

service SRVGRP=group5
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n22"
      RQADDR=tpcc_22 SRVID=22

service SRVGRP=group5
```

```
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n23"
    RQADDR=tpcc_23 SRVID=23

service SRVGRP=group5
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n24"
    RQADDR=tpcc_24 SRVID=24

service SRVGRP=group5
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n25"
    RQADDR=tpcc_25 SRVID=25

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n26"
    RQADDR=tpcc_26 SRVID=26

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n27"
    RQADDR=tpcc_27 SRVID=27

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n28"
    RQADDR=tpcc_28 SRVID=28

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n29"
    RQADDR=tpcc_29 SRVID=29

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n30"
    RQADDR=tpcc_30 SRVID=30

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n31"
    RQADDR=tpcc_31 SRVID=31

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n32"
    RQADDR=tpcc_32 SRVID=32

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n33"
    RQADDR=tpcc_33 SRVID=33

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n34"
    RQADDR=tpcc_34 SRVID=34

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n35"
    RQADDR=tpcc_35 SRVID=35

service SRVGRP=group8
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n36"
    RQADDR=tpcc_36 SRVID=36

service SRVGRP=group8
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n37"
    RQADDR=tpcc_37 SRVID=37

service SRVGRP=group8
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n38"
    RQADDR=tpcc_38 SRVID=38

service SRVGRP=group8
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n39"
    RQADDR=tpcc_39 SRVID=39

service SRVGRP=group8
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n40"
    RQADDR=tpcc_40 SRVID=40

service SRVGRP=group9
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n41"
    RQADDR=tpcc_41 SRVID=41

service SRVGRP=group9
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n42"
    RQADDR=tpcc_42 SRVID=42

service SRVGRP=group9
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n43"
    RQADDR=tpcc_43 SRVID=43

service SRVGRP=group9
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n44"
    RQADDR=tpcc_44 SRVID=44

service SRVGRP=group9
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n45"
    RQADDR=tpcc_45 SRVID=45

service SRVGRP=group10
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n46"
    RQADDR=tpcc_46 SRVID=46

service SRVGRP=group10
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n47"
    RQADDR=tpcc_47 SRVID=47

service SRVGRP=group10
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n48"
    RQADDR=tpcc_48 SRVID=48

service SRVGRP=group10
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n49"
    RQADDR=tpcc_49 SRVID=49

service SRVGRP=group10
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n50"
    RQADDR=tpcc_50 SRVID=50

service SRVGRP=group11
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n51"
    RQADDR=tpcc_51 SRVID=51

service SRVGRP=group11
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n52"
    RQADDR=tpcc_52 SRVID=52

service SRVGRP=group11
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n53"
    RQADDR=tpcc_53 SRVID=53

service SRVGRP=group11
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n54"
    RQADDR=tpcc_54 SRVID=54

service SRVGRP=group11
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n55"
    RQADDR=tpcc_55 SRVID=55

service SRVGRP=group12
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n56"
    RQADDR=tpcc_56 SRVID=56

service SRVGRP=group12
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n57"
    RQADDR=tpcc_57 SRVID=57

service SRVGRP=group12
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n58"
    RQADDR=tpcc_58 SRVID=58

service SRVGRP=group12
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n59"
    RQADDR=tpcc_59 SRVID=59

service SRVGRP=group12
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n60"
    RQADDR=tpcc_60 SRVID=60

service SRVGRP=group13
```

```
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n61"
        RQADDR=tpcc_61 SRVID=61

service SRVGRP=group13
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n62"
        RQADDR=tpcc_62 SRVID=62

service SRVGRP=group13
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n63"
        RQADDR=tpcc_63 SRVID=63

service SRVGRP=group13
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n64"
        RQADDR=tpcc_64 SRVID=64

service SRVGRP=group13
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n65"
        RQADDR=tpcc_65 SRVID=65
#------------------------------------------------------------------
*SERVICES
#------------------------------------------------------------------
*ROUTING
#------------------------------------------------------------------

# This is a UBBconfig for a client1-server configuration.
#
# This UBBconfig requires settings for:
# SERVER_NAME CLIENT_NAME MASTER_NAME SERVER_ADDR CLIENT_ADDR
NODE_NAMES
# TLISTEN_PORT TBRIDGE_PORT
# In addition, it requires setting the things all UBBconfig.gens need:
#               IPCKEY                  some decent IPCKEY, should be different for each
config
#               ROOTDIR
#               TUXCONFIG
#               APPDIR
#               ULOGDIR
#

#------------------------------------------------------------------
*RESOURCES
#------------------------------------------------------------------
                IPCKEY          40001
                PERM            0666
                MASTER          client1

  MAXACCESSERS          13050           # 1024 or more
  MAXGTT    1024
  MAXSERVERS            70
  MAXSERVICES           335   # MAXSERVERS * #-of-services-each-server + 10( for BBL)
  MODEL     SHM
  LDBAL     Y

# During benchmark, don't want to scan too often.  In particular, while
# the client1s are stabilizing in virtual memory, we don't want to sanity
# scan; and if we do sanity scan, we want large timeouts, since the BRIDGE
# the BBL, the DBBL, and the client1s aren't getting much CPU time during that
# period.  Current settings:
#               *  scan servers every 5 minutes (maximum allowed by TUXEDO);
#               *  wait 1 minute for sanity  responses (maximum allowed by TUXEDO);
#               *  scan all the BBLs from DBBL every 30 minutes (want one scan in the
#                  audited results);
#               *  timeout a blocking call after 5 minutes (the maximum).
                SCANUNIT    60
                SANITYSCAN              5
                DBBLWAIT    1
                BBLQUERY    30
                BLOCKTIME 5
#
#------------------------------------------------------------------
*MACHINES
#------------------------------------------------------------------
DEFAULT:
        TUXCONFIG="/project/iti/confs/TUXconfig.client1"
                        ROOTDIR="/project/iti"
                        APPDIR="/project/tpcc/bin"
                        ULOGPFX="/tmp/TUXEDO_LOG"

# for debugging, put both into the same log on the same machine
#                       ULOGPFX="/home/iti/confs/tpcc/ULOG"
# but for a big run, need some space, and want them local to the
# machine rather than across the net.

# Leave TUXCONFIG alone on the MASTER machine; over-ride for each
# other machine?
client1         LMID=client1
        TUXCONFIG="/project/iti/confs/TUXconfig.client1"
#------------------------------------------------------------------
*GROUPS
#------------------------------------------------------------------
group1          LMID=client1
                GRPNO=1
group2          LMID=client1
                GRPNO=2
group3          LMID=client1
                GRPNO=3
group4          LMID=client1
                GRPNO=4
group5          LMID=client1
                GRPNO=5
group6          LMID=client1
                GRPNO=6
group7          LMID=client1
                GRPNO=7
group8          LMID=client1
                GRPNO=8
group9          LMID=client1
                GRPNO=9
group10         LMID=client1
                GRPNO=10
group11         LMID=client1
                GRPNO=11
group12         LMID=client1
                GRPNO=12
group13         LMID=client1
                GRPNO=13

#------------------------------------------------------------------
#------------------------------------------------------------------

#------------------------------------------------------------------
*SERVERS
#------------------------------------------------------------------
#
# "--" is application-specific arguments to be passed to server
# "-n" is designed to specify server-id

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n1"
        RQADDR=tpcc_1 SRVID=1

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n2"
        RQADDR=tpcc_2 SRVID=2

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n3"
        RQADDR=tpcc_3 SRVID=3

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n4"
        RQADDR=tpcc_4 SRVID=4

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n5"
        RQADDR=tpcc_5 SRVID=5

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n6"
        RQADDR=tpcc_6 SRVID=6

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n7"
        RQADDR=tpcc_7 SRVID=7

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n8"
        RQADDR=tpcc_8 SRVID=8

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n9"
        RQADDR=tpcc_9 SRVID=9

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n10"
        RQADDR=tpcc_10 SRVID=10

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n11"
        RQADDR=tpcc_11 SRVID=11

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n12"
```

```
        RQADDR=tpcc_12 SRVID=12

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n13"
        RQADDR=tpcc_13 SRVID=13

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n14"
        RQADDR=tpcc_14 SRVID=14

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n15"
        RQADDR=tpcc_15 SRVID=15

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n16"
        RQADDR=tpcc_16 SRVID=16

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n17"
        RQADDR=tpcc_17 SRVID=17

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n18"
        RQADDR=tpcc_18 SRVID=18

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n19"
        RQADDR=tpcc_19 SRVID=19

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n20"
        RQADDR=tpcc_20 SRVID=20

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n21"
        RQADDR=tpcc_21 SRVID=21

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n22"
        RQADDR=tpcc_22 SRVID=22

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n23"
        RQADDR=tpcc_23 SRVID=23

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n24"
        RQADDR=tpcc_24 SRVID=24

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n25"
        RQADDR=tpcc_25 SRVID=25

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n26"
        RQADDR=tpcc_26 SRVID=26

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n27"
        RQADDR=tpcc_27 SRVID=27

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n28"
        RQADDR=tpcc_28 SRVID=28

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n29"
        RQADDR=tpcc_29 SRVID=29

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n30"
        RQADDR=tpcc_30 SRVID=30

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n31"
        RQADDR=tpcc_31 SRVID=31

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n32"
        RQADDR=tpcc_32 SRVID=32

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n33"
        RQADDR=tpcc_33 SRVID=33

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n34"
        RQADDR=tpcc_34 SRVID=34

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n35"
        RQADDR=tpcc_35 SRVID=35

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n36"
        RQADDR=tpcc_36 SRVID=36

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n37"
        RQADDR=tpcc_37 SRVID=37

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n38"
        RQADDR=tpcc_38 SRVID=38

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n39"
        RQADDR=tpcc_39 SRVID=39

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n40"
        RQADDR=tpcc_40 SRVID=40

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n41"
        RQADDR=tpcc_41 SRVID=41

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n42"
        RQADDR=tpcc_42 SRVID=42

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n43"
        RQADDR=tpcc_43 SRVID=43

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n44"
        RQADDR=tpcc_44 SRVID=44

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n45"
        RQADDR=tpcc_45 SRVID=45

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n46"
        RQADDR=tpcc_46 SRVID=46

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n47"
        RQADDR=tpcc_47 SRVID=47

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n48"
        RQADDR=tpcc_48 SRVID=48

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n49"
        RQADDR=tpcc_49 SRVID=49

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n50"
```

```
        RQADDR=tpcc_50 SRVID=50

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n51"
        RQADDR=tpcc_51 SRVID=51

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n52"
        RQADDR=tpcc_52 SRVID=52

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n53"
        RQADDR=tpcc_53 SRVID=53

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n54"
        RQADDR=tpcc_54 SRVID=54

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n55"
        RQADDR=tpcc_55 SRVID=55

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n56"
        RQADDR=tpcc_56 SRVID=56

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n57"
        RQADDR=tpcc_57 SRVID=57

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n58"
        RQADDR=tpcc_58 SRVID=58

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n59"
        RQADDR=tpcc_59 SRVID=59

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n60"
        RQADDR=tpcc_60 SRVID=60

service SRVGRP=group13
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n61"
        RQADDR=tpcc_61 SRVID=61

service SRVGRP=group13
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n62"
        RQADDR=tpcc_62 SRVID=62

service SRVGRP=group13
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n63"
        RQADDR=tpcc_63 SRVID=63

service SRVGRP=group13
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n64"
        RQADDR=tpcc_64 SRVID=64

service SRVGRP=group13
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n65"
        RQADDR=tpcc_65 SRVID=65
#------------------------------------------------------------------
*SERVICES
#------------------------------------------------------------------
*ROUTING
#------------------------------------------------------------------

# This is a UBBconfig for a client1-server configuration.
#
# This UBBconfig requires settings for:
# SERVER_NAME CLIENT_NAME MASTER_NAME SERVER_ADDR CLIENT_ADDR
NODE_NAMES
# TLISTEN_PORT TBRIDGE_PORT
# In addition, it requires setting the things all UBBconfig.gens need:
#               IPCKEY                  some decent IPCKEY, should be different for each
config
#               ROOTDIR
#               TUXCONFIG
#               APPDIR
#               ULOGDIR
#

#------------------------------------------------------------------
*RESOURCES
#------------------------------------------------------------------
                IPCKEY          40001
                PERM            0666
                MASTER          client1

  MAXACCESSERS          6250            # 1024 or more
  MAXGTT    1024
  MAXSERVERS            65
  MAXSERVICES           310   # MAXSERVERS * #-of-services-each-server + 10( for BBL)
  MODEL     SHM
  LDBAL     Y

# During benchmark, don't want to scan too often.  In particular, while
# the client1s are stabilizing in virtual memory, we don't want to sanity
# scan; and if we do sanity scan, we want large timeouts, since the BRIDGE
# the BBL, the DBBL, and the client1s aren't getting much CPU time during that
# period.  Current settings:
#               *  scan servers every 5 minutes (maximum allowed by TUXEDO);
#               *  wait 1 minute for sanity  responses (maximum allowed by TUXEDO);
#               *  scan all the BBLs from DBBL every 30 minutes (want one scan in the
#                  audited results);
#               *  timeout a blocking call after 5 minutes (the maximum).
                SCANUNIT    60
                SANITYSCAN                  5
                DBBLWAIT    1
                BBLQUERY    30
                BLOCKTIME 5
#
#------------------------------------------------------------------
*MACHINES
#------------------------------------------------------------------
DEFAULT:
            TUXCONFIG="/project/iti/confs/TUXconfig.client1"
                                ROOTDIR="/project/iti"
                                APPDIR="/project/tpcc/bin"
                                ULOGPFX="/tmp/TUXEDO_LOG"

# for debugging, put both into the same log on the same machine
#                               ULOGPFX="/home/iti/confs/tpcc/ULOG"
# but for a big run, need some space, and want them local to the
# machine rather than across the net.

# Leave TUXCONFIG alone on the MASTER machine; over-ride for each
# other machine?
client1         LMID=client1
        TUXCONFIG="/project/iti/confs/TUXconfig.client1"
#------------------------------------------------------------------
*GROUPS
#------------------------------------------------------------------
group1          LMID=client1
                GRPNO=1
group2          LMID=client1
                GRPNO=2
group3          LMID=client1
                GRPNO=3
group4          LMID=client1
                GRPNO=4
group5          LMID=client1
                GRPNO=5
group6          LMID=client1
                GRPNO=6
group7          LMID=client1
                GRPNO=7
group8          LMID=client1
                GRPNO=8
group9          LMID=client1
                GRPNO=9
group10           LMID=client1
                GRPNO=10
group11           LMID=client1
                GRPNO=11
group12           LMID=client1
                GRPNO=12

#------------------------------------------------------------------
#------------------------------------------------------------------

#------------------------------------------------------------------
*SERVERS
#------------------------------------------------------------------
#
# "--" is application-specific arguments to be passed to server
# "-n" is designed to specify server-id

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n1"
        RQADDR=tpcc_1 SRVID=1

service SRVGRP=group1
```

```
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n2"
	RQADDR=tpcc_2 SRVID=2

service SRVGRP=group1
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n3"
	RQADDR=tpcc_3 SRVID=3

service SRVGRP=group1
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n4"
	RQADDR=tpcc_4 SRVID=4

service SRVGRP=group1
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n5"
	RQADDR=tpcc_5 SRVID=5

service SRVGRP=group2
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n6"
	RQADDR=tpcc_6 SRVID=6

service SRVGRP=group2
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n7"
	RQADDR=tpcc_7 SRVID=7

service SRVGRP=group2
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n8"
	RQADDR=tpcc_8 SRVID=8

service SRVGRP=group2
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n9"
	RQADDR=tpcc_9 SRVID=9

service SRVGRP=group2
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n10"
	RQADDR=tpcc_10 SRVID=10

service SRVGRP=group3
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n11"
	RQADDR=tpcc_11 SRVID=11

service SRVGRP=group3
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n12"
	RQADDR=tpcc_12 SRVID=12

service SRVGRP=group3
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n13"
	RQADDR=tpcc_13 SRVID=13

service SRVGRP=group3
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n14"
	RQADDR=tpcc_14 SRVID=14

service SRVGRP=group3
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n15"
	RQADDR=tpcc_15 SRVID=15

service SRVGRP=group4
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n16"
	RQADDR=tpcc_16 SRVID=16

service SRVGRP=group4
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n17"
	RQADDR=tpcc_17 SRVID=17

service SRVGRP=group4
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n18"
	RQADDR=tpcc_18 SRVID=18

service SRVGRP=group4
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n19"
	RQADDR=tpcc_19 SRVID=19

service SRVGRP=group4
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n20"
	RQADDR=tpcc_20 SRVID=20

service SRVGRP=group5
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n21"
	RQADDR=tpcc_21 SRVID=21

service SRVGRP=group5
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n22"
	RQADDR=tpcc_22 SRVID=22

service SRVGRP=group5
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n23"
	RQADDR=tpcc_23 SRVID=23

service SRVGRP=group5
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n24"
	RQADDR=tpcc_24 SRVID=24

service SRVGRP=group5
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n25"
	RQADDR=tpcc_25 SRVID=25

service SRVGRP=group6
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n26"
	RQADDR=tpcc_26 SRVID=26

service SRVGRP=group6
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n27"
	RQADDR=tpcc_27 SRVID=27

service SRVGRP=group6
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n28"
	RQADDR=tpcc_28 SRVID=28

service SRVGRP=group6
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n29"
	RQADDR=tpcc_29 SRVID=29

service SRVGRP=group6
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n30"
	RQADDR=tpcc_30 SRVID=30

service SRVGRP=group7
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n31"
	RQADDR=tpcc_31 SRVID=31

service SRVGRP=group7
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n32"
	RQADDR=tpcc_32 SRVID=32

service SRVGRP=group7
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n33"
	RQADDR=tpcc_33 SRVID=33

service SRVGRP=group7
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n34"
	RQADDR=tpcc_34 SRVID=34

service SRVGRP=group7
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n35"
	RQADDR=tpcc_35 SRVID=35

service SRVGRP=group8
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n36"
	RQADDR=tpcc_36 SRVID=36

service SRVGRP=group8
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n37"
	RQADDR=tpcc_37 SRVID=37

service SRVGRP=group8
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n38"
	RQADDR=tpcc_38 SRVID=38

service SRVGRP=group8
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n39"
	RQADDR=tpcc_39 SRVID=39

service SRVGRP=group8
```

```
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n40"
	RQADDR=tpcc_40 SRVID=40

service SRVGRP=group9
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n41"
	RQADDR=tpcc_41 SRVID=41

service SRVGRP=group9
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n42"
	RQADDR=tpcc_42 SRVID=42

service SRVGRP=group9
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n43"
	RQADDR=tpcc_43 SRVID=43

service SRVGRP=group9
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n44"
	RQADDR=tpcc_44 SRVID=44

service SRVGRP=group9
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n45"
	RQADDR=tpcc_45 SRVID=45

service SRVGRP=group10
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n46"
	RQADDR=tpcc_46 SRVID=46

service SRVGRP=group10
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n47"
	RQADDR=tpcc_47 SRVID=47

service SRVGRP=group10
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n48"
	RQADDR=tpcc_48 SRVID=48

service SRVGRP=group10
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n49"
	RQADDR=tpcc_49 SRVID=49

service SRVGRP=group10
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n50"
	RQADDR=tpcc_50 SRVID=50

service SRVGRP=group11
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n51"
	RQADDR=tpcc_51 SRVID=51

service SRVGRP=group11
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n52"
	RQADDR=tpcc_52 SRVID=52

service SRVGRP=group11
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n53"
	RQADDR=tpcc_53 SRVID=53

service SRVGRP=group11
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n54"
	RQADDR=tpcc_54 SRVID=54

service SRVGRP=group11
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n55"
	RQADDR=tpcc_55 SRVID=55

service SRVGRP=group12
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n56"
	RQADDR=tpcc_56 SRVID=56

service SRVGRP=group12
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n57"
	RQADDR=tpcc_57 SRVID=57

service SRVGRP=group12
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n58"
	RQADDR=tpcc_58 SRVID=58

service SRVGRP=group12
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n59"
	RQADDR=tpcc_59 SRVID=59

service SRVGRP=group12
		CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n60"
	RQADDR=tpcc_60 SRVID=60
#------------------------------------------------------------------
*SERVICES
#------------------------------------------------------------------
*ROUTING
#------------------------------------------------------------------

# This is a UBBconfig for a client1-server configuration.
#
# This UBBconfig requires settings for:
# SERVER_NAME CLIENT_NAME MASTER_NAME SERVER_ADDR CLIENT_ADDR
NODE_NAMES
# TLISTEN_PORT TBRIDGE_PORT
# In addition, it requires setting the things all UBBconfig.gens need:
#		IPCKEY			some decent IPCKEY, should be different for each
config
#		ROOTDIR
#		TUXCONFIG
#		APPDIR
#		ULOGDIR
#

#------------------------------------------------------------------
*RESOURCES
#------------------------------------------------------------------
		IPCKEY	40001
		PERM	0666
		MASTER	client2

  MAXACCESSERS		1250		# 1024 or more
  MAXGTT	1024
  MAXSERVERS		40
  MAXSERVICES		185	# MAXSERVERS * #-of-services-each-server + 10( for BBL)
  MODEL	SHM
  LDBAL	Y

# During benchmark, don't want to scan too often.  In particular, while
# the client1s are stabilizing in virtual memory, we don't want to sanity
# scan; and if we do sanity scan, we want large timeouts, since the BRIDGE
# the BBL, the DBBL, and the client1s aren't getting much CPU time during that
# period.  Current settings:
#		*  scan servers every 5 minutes (maximum allowed by TUXEDO);
#		*  wait 1 minute for sanity  responses (maximum allowed by TUXEDO);
#		*  scan all the BBLs from DBBL every 30 minutes (want one scan in the
#		   audited results);
#		*  timeout a blocking call after 5 minutes (the maximum).
		SCANUNIT	60
		SANITYSCAN		5
		DBBLWAIT	1
		BBLQUERY	30
		BLOCKTIME	5
#
#------------------------------------------------------------------
*MACHINES
#------------------------------------------------------------------
DEFAULT:
		TUXCONFIG="/project/iti/confs/TUXconfig.client2"
				ROOTDIR="/project/iti"
				APPDIR="/project/tpcc/bin"
				ULOGPFX="/tmp/TUXEDO_LOG"

# for debugging, put both into the same log on the same machine
#				ULOGPFX="/home/iti/confs/tpcc/ULOG"
# but for a big run, need some space, and want them local to the
# machine rather than across the net.

# Leave TUXCONFIG alone on the MASTER machine; over-ride for each
# other machine?
client2		LMID=client2
	TUXCONFIG="/project/iti/confs/TUXconfig.client2"
#------------------------------------------------------------------
*GROUPS
#------------------------------------------------------------------
group1		LMID=client2
		GRPNO=1
group2		LMID=client2
		GRPNO=2
group3		LMID=client2
		GRPNO=3
group4		LMID=client2
		GRPNO=4
group5		LMID=client2
		GRPNO=5
group6		LMID=client2
		GRPNO=6
group7		LMID=client2
		GRPNO=7
```

```
#------------------------------------------------------------------------
#------------------------------------------------------------------------

#------------------------------------------------------------------------
*SERVERS
#------------------------------------------------------------------------
#
# "--" is application-specific arguments to be passed to server
# "-n" is designed to specify server-id

service SRVGRP=group1
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n1"
    RQADDR=tpcc_1 SRVID=1

service SRVGRP=group1
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n2"
    RQADDR=tpcc_2 SRVID=2

service SRVGRP=group1
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n3"
    RQADDR=tpcc_3 SRVID=3

service SRVGRP=group1
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n4"
    RQADDR=tpcc_4 SRVID=4

service SRVGRP=group1
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n5"
    RQADDR=tpcc_5 SRVID=5

service SRVGRP=group2
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n6"
    RQADDR=tpcc_6 SRVID=6

service SRVGRP=group2
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n7"
    RQADDR=tpcc_7 SRVID=7

service SRVGRP=group2
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n8"
    RQADDR=tpcc_8 SRVID=8

service SRVGRP=group2
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n9"
    RQADDR=tpcc_9 SRVID=9

service SRVGRP=group2
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n10"
    RQADDR=tpcc_10 SRVID=10

service SRVGRP=group3
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n11"
    RQADDR=tpcc_11 SRVID=11

service SRVGRP=group3
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n12"
    RQADDR=tpcc_12 SRVID=12

service SRVGRP=group3
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n13"
    RQADDR=tpcc_13 SRVID=13

service SRVGRP=group3
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n14"
    RQADDR=tpcc_14 SRVID=14

service SRVGRP=group3
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n15"
    RQADDR=tpcc_15 SRVID=15

service SRVGRP=group4
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n16"
    RQADDR=tpcc_16 SRVID=16

service SRVGRP=group4
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n17"
    RQADDR=tpcc_17 SRVID=17

service SRVGRP=group4
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n18"
    RQADDR=tpcc_18 SRVID=18

service SRVGRP=group4
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n19"
    RQADDR=tpcc_19 SRVID=19

service SRVGRP=group4
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n20"
    RQADDR=tpcc_20 SRVID=20

service SRVGRP=group5
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n21"
    RQADDR=tpcc_21 SRVID=21

service SRVGRP=group5
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n22"
    RQADDR=tpcc_22 SRVID=22

service SRVGRP=group5
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n23"
    RQADDR=tpcc_23 SRVID=23

service SRVGRP=group5
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n24"
    RQADDR=tpcc_24 SRVID=24

service SRVGRP=group5
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n25"
    RQADDR=tpcc_25 SRVID=25

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n26"
    RQADDR=tpcc_26 SRVID=26

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n27"
    RQADDR=tpcc_27 SRVID=27

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n28"
    RQADDR=tpcc_28 SRVID=28

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n29"
    RQADDR=tpcc_29 SRVID=29

service SRVGRP=group6
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n30"
    RQADDR=tpcc_30 SRVID=30

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n31"
    RQADDR=tpcc_31 SRVID=31

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n32"
    RQADDR=tpcc_32 SRVID=32

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n33"
    RQADDR=tpcc_33 SRVID=33

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n34"
    RQADDR=tpcc_34 SRVID=34

service SRVGRP=group7
        CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n35"
    RQADDR=tpcc_35 SRVID=35
#------------------------------------------------------------------------
*SERVICES
#------------------------------------------------------------------------
*ROUTING
#------------------------------------------------------------------------
```

Appendix D RTE Configuration

This appendix lists RTE input parameters and code fragments used to generate each transaction input file, to demonstrate the RTE was configured to generate transaction input data as specified in *Clause 2* of the specification.

D.1 Field Value Generation

Source/src/driver/generate.c

```
/*********************************************************************************
 @(#) Version: A.10.10 $Date: 2002/07/18 21:51:41 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*********************************************************************************/

#include <stdio.h>
#include <values.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <fcntl.h>
#include <signal.h>
#include <math.h>

#include "shm_lookup.h"
#include "random.h"

#include <time.h>

int CLAST_CONST_C = 208;
int CID_CONST_C =  37;
int IID_CONST_C = 75;

int trans_type = 0;   /* type of transaction 0 == all */

extern ID warehouse;
extern ID district;

neworder_gen(t)
   neworder_trans *t;
   {
   int i;

   t->W_ID = warehouse;

   t->D_ID = RandomNumber(1, no_dist_pw);
   t->C_ID = NURandomNumber( 1023, 1, no_cust_pd, CID_CONST_C);

   t->O_OL_CNT = RandomNumber(5, 15);

   for (i=0; i<t->O_OL_CNT; i++)
      {
      t->item[i].OL_I_ID = NURandomNumber(8191, 1, no_item, IID_CONST_C);
      t->item[i].OL_SUPPLY_W_ID = RandomWarehouse(warehouse, scale, 1);
      t->item[i].OL_QUANTITY = RandomNumber(1, 10);
      }
   /* Zero out the non-used items as the oracle driver does. */
   for (; i< 15; i++)
      {
      t->item[i].OL_I_ID = 0;
      t->item[i].OL_SUPPLY_W_ID = 0;
      t->item[i].OL_QUANTITY = 0;
      }


   /* 1% of transactions roll back. Give the last order line a bad item */
   if (RandomNumber(1, 100) == 1)
      t->item[t->O_OL_CNT - 1].OL_I_ID = -1;
   }

payment_gen(t)
   payment_trans *t;
   {

   /* home warehouse is fixed */
   t->W_ID = warehouse;

   /* Random district */
   t->D_ID = RandomNumber(1, no_dist_pw);

   /* Customer is from remote warehouse and district 15% of the time */
   t->C_W_ID = RandomWarehouse(warehouse, scale, 15);
   if (t->C_W_ID == t->W_ID)
      t->C_D_ID = t->D_ID;
   else
      t->C_D_ID = RandomNumber(1, no_dist_pw);

   /* by name 60% of the time */
   t->byname = RandomNumber(1, 100) <= 60;
   if (t->byname)
      LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1, CLAST_CONST_C),
                                    t->C_LAST);
   else
      t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);

   /* amount is random from [1.00..5,000.00] */
   t->H_AMOUNT = RandomNumber(100, 500000);

   }

ordstat_gen(t)
   ordstat_trans *t;
   {

   /* home warehouse is fixed */
   t->W_ID = warehouse;

   /* district is randomly selected from warehouse */
   t->D_ID = RandomNumber(1, no_dist_pw);

   /* by name 60% of the time */
   t->byname = RandomNumber(1, 100) <= 60;
   if (t->byname)
      LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1, CLAST_CONST_C),
                                    t->C_LAST);
   else
      t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);
   }

delivery_gen(t)
   delivery_trans *t;
   {
   t->W_ID = warehouse;
   t->O_CARRIER_ID = RandomNumber(1,10);
   }

stocklev_gen(t)
   stocklev_trans *t;
   {
   t->W_ID = warehouse;
   t->D_ID = district;
   t->threshold = RandomNumber(10, 20);
   }

int get_trans_type()
/*********************************************************************************
 * get_trans_type selects a transaction according to the weighted average
 * For TPC-C rev 3.0 and less and TPC-C rev 3.2 this is:
 *       new-order :  ???
 *       payment   : 43.0%
 *       order stat:  4.0%
 *       delivery  :  4.0%
 *       stock     :  4.0%
 *********************************************************************************/
   {
   static double weight[] = { 0.0,  0.0, .4301, .0401, .0401, .0401};
   double drand48();
   int type;
   double r;

   /* choose a random number between 0.0 and 1.0 */
   if (trans_type == 0) {
#ifdef USE_DRAND48
      r = drand48();
#else
      r = randy();
#endif

      /*
       * select one of STOCKLEV, DELIVERY, ORDSTAT and PAYMENT
       * based on weight
       */
      for (type = STOCKLEV;  type > NEWORDER;  type--) {
         r -= weight[type];
         if (r < 0) break;
      }
   } else if (trans_type > 0) {
                  /* user wants only a certain type (say all stocklevel) so do that
                     instead */
                  type = trans_type;
```

```
} else {
                /* Trans type is less than zero, so this means exclude only
                   the selected type */
   for (type = STOCKLEV;  type > NEWORDER;  type--) {
      r -= weight[type];
                   if (-trans_type == type) { continue; }
      if (r < 0) break;
   }
                if (-trans_type == NEWORDER &&
                                type == NEWORDER) { type = PAYMENT; }
}
/* return the value of the selected card, or NEWORDER if none selected */
return type;
}
```

Appendix E Disk Storage

The calculations used to determine the storage requirements for the 8 hours logical log and the 60-day space calculations are contained in this appendix.

The calculations used to determine the storage requirements for the 8 hours logical log and the 60-day space calculations are contained in this appendix.

The calculations for the 8 hours recovery log were based on how often the oracle redo log files were filling up and needed to be switched. The database took a checkpoint, and switched from the "current" log file to the other, "active" log file, every 29.51 minutes. Each log file is 13000MB.
So, to run for 8 hours, we need:

((8 * 60) / 29.51) * 13000 / 1024 = 206.50GB (must be mirrored)

The log disk array has a capacity of 497.22GB.

TSPACE NAME	BLOCKS ALLOCATED	BLOCK SIZE	INITIAL STATIC	INITIAL DYNAMIC	8-HOUR REQUIRED	OVERSIZE (BLOCKS)
CUSTCLUSTER	105697280	2048	93542400	93542400	0	12154880
DISTCLUSTER	140700	2048	73330	73330	0	67370
HIST	17493935	2048	7509561	0	6297822	9984374
ICUST1	454170	16384	336000	336000	0	118170
ICUST2	834940	16384	698880	698880	0	136060
IDIST	28140	2048	14236	14236	0	13904
IITEM	1075	2048	1016	1016	0	59
IORDR2	1178336	16384	483840	483840	0	694496
ISTOK	1049507	16384	927360	927360	0	122147
ITEMCLUSTER	7690	2048	7537	7537	0	153
IWARE	3518	2048	3324	3324	0	194
NORDCLUSTER	3479511	2048	1096049	1096049	0	2383462
ORDRCLUSTER	35757594	16384	12317532	0	10329982	23440062
STOKCLUSTER	168817664	2048	140851200	140851200	0	27966464
SYSTEM	102400	2048	102400	102400	0	0
WARECLUSTER	10500	2048	10500	10500	0	0
Benchmark stats	3427840	2048	0	0	0	3427840

TPC-C 60-Day Space Requirements

TPM 80570.81
Warehouses 6700

SEGMENT	TYPE	TSPACE	BLOCKS	FIVE_PCT	DAILY_GROWTH	BLOCK SIZE	TOTAL in MB
CUSTCLUSTER	CLUSTER	CUST_0	89,088,000	4,454,400	0	2,048	182,700
DISTCLUSTER	CLUSTER	DIST_0	69,838	3,492	0	2,048	143
HIST	TABLE	HIST_0	6,297,822	0	1,211,751	2,048	14,667
ICUST1	INDEX	ICUST1_0	320,000	16,000	0	16,384	5,250
ICUST2	INDEX	ICUST2_0	665,600	33,280	0	16,384	10,920
IDIST	INDEX	IDIST_0	13,558	678	0	2,048	28
IITEM	INDEX	IITEM_0	968	48	0	2,048	2
IORDR2	INDEX	IORD2_0	460,800	23,040	0	16,384	7,560
ISTOK	INDEX	ISTOK_0	883,200	44,160	0	16,384	14,490
ITEMCLUSTER	CLUSTER	ITEM_0	7,178	359	0	2,048	15
IWARE	INDEX	IWARE_0	3,166	158	0	2,048	6
NORDCLUSTER	TABLE	NORD_0	1,043,856	52,193	0	2,048	2,141
ORDRCLUSTER	TABLE	ORDR_0	10,329,982	0	1,987,570	16,384	192,462
STOKCLUSTER	CLUSTER	STOK_0	134,144,000	6,707,200	0	2,048	275,100
SYSTEM	SYS	SYS	102,400	0	0	2,048	200
WARECLUSTER	CLUSTER	WARE_0	10,000	500	0	2,048	21
SYSAUX	SYS	SYSAUX	61,440	0		2,048	120
Benchmark stats	SYS	SYS	18,432	0	0	2,048	36
Undo	SYS	UNDO_TS	1,715,200	0	0	8,192	13,400
Total							719,260

Dynamic space	173,706	Initial MB for (History+Orders)
Static space	512,131	Initial Blocks + 5% - Dynamic
Daily Growth	33,422	Total Dynamic [(calc. as (Initial Blocks)*tpmC/(WHS*62.5)]
Daily Spread	0	Oracle may be configured so that daily spread is 0
60-day space (MB)	2,517,481	Static + 60*(Daily Growth+Daily Spread)
60-day (GB)	2,458.48	Excludes OS, Paging and RDBMS Logs
8-hour log (GB)	206.50	RDBMS Logs
Server swap (GB)	48.00	OS: Paging
Server OS (GB)	16.00	OS: UNIX File System
Total Space Needed	2,728.97	GB

Priced-Sytem Configuration	**Size in MB after RAID 1 redundency**	**Quantity**	**Total (GB)**
VA7100 with 15 18.2 GB disk drives in RAID 1 mode	124,652	12	1,460.77
VA7100 with 15 36.4-GB disk drives in RAID 1 mode	252,811	3	740.66
VA7100 with 15 73-GB disk drives in RAID 1 mode	509,153	1	497.22
Internal 36GB drives	34,733	1	33.92
Total Storage in Priced System (GB)			2,732.56

Appendix F Price Quotes

The following pages contain the price quotes for the hardware included in this FDR.

Andreas Hotea
HP
Cupertino, CA 95014
November 11, 2002

HP Unix Sales Development
19111 Pruneridge Avenue
Cupertino, CA 95014
(408) 447-2320

hp server rx5670

TPC-C Rev 5

Report Date: November 13, 2002

Description	Part Number	Brand	Price Key	US List Price	Qty	Price	3Year Main.Price
Server Hardware							
hp server rx5670 with 1GHZ processor	A6838A		1	26,494	1	26,494	7,052
1GHz Processor with 3MB Cache	A6836A		1	8,250	3	24,750	3,459
4GB PC2100 DDR-SDRAM Memory quad	A6834A		1	8,000	12	96,000	
Memory Carrier Board	A6747A		1	1,981	2	3,962	
36GB 10K Internal Drive	A7048A		1	587	1	587	
DVD ROM	A5557B		1	450	1	450	
Sytstem Console	C1099A		1	550	1	550	
HP-UX Operating Environment	B0916A, Opt AFF		1	520	1	520	
HP-UX 11i, v1.6 w/Performance Pack Per-Processor Lic	B9429AC		1	2,495	4	9,980	
2GB Fibre Channel Adapter	A6795A		1	2,240	4	8,960	
PowerTrust 3.0VA UPS 230VUPS	A1353A		1	1,735	8	13,880	
Surestore VA 7100 w/dual controllers 512MB cache	A6262A		1	44,250	16	708,000	58,784
18GB 15K RPM FC HDD	A6191A, Opt. 0D1		1	914	180	164,520	
36GB 15K RPM FC HDD.	A6193A, Opt 0D1		1	1,349	45	60,705	
73GB 10K RPM FC HDD.	A6194A, Opt 0D1		1	2,019	15	30,285	
2 meter LC/SC Fibre Optic Cable	C7529A		1	215	20	4,300	
HP FC 1GB/2GB Entry Switch 8B, Field Rack	A7346A		1	6,599	4	26,396	
HP9000 Std. Rack System E41	A4902A		1	1,910	2	3,820	
Modular Power Dist.	A5137AZ		1	145	8	1,160	
200-240 Volts Power Option	A5137AZ, Opt AW4		1	94	8	752	
				Subtotal		**1,186,071**	**69,295**
Client Hardware							
HP server rp2470	A6890A		1	1,865	7	13,055	24,815
750Mhz PA-RISC 8700 CPU	A6892A		1	5,100	7	35,700	
18GB 10K HotPlug Ultra 160 SCSI Internal Disk	A6741A		1	732	7	5,124	
2GB Memory Module	A6114A		1	6,000	7	42,000	
1GB Memory Module	A5841A		1	1,999	7	13,993	
HP-UX 11.i Sys Media, CD-ROM	B3920EA, Opt. AAF		1	520	7	3,640	
				Subtotal		**113,512**	**24,815**
Client Software							
HP C/ANSI C Compiler	B3901BA, Option AH0		1	1600	1	1,600	170
				Subtotal		**1,600**	**170**
User Connectivity							
HP ProCurve Switch 4000M	J4121A		1	2379	1	2,379	588
HP ProCurve Switch 100/1000Base-T Mod.	J4115B		1	509	1	509	
				Subtotal		**2,888**	**588**
HP's Large configuration Discount and Support Prepayment*						($387,023)	($10,649)
*All discounts are based on US list prices and for similar quantities and configurations				**Total**		**917,048**	**84,219**

All the components in the price list are currently available. Maintenance support price is for 24 hours, 7 days with 4 hour response time.

From: Herve Lejeune [herve.lejeune@oracle.com]
Sent: Wednesday, November 06, 2002 4:44 PM
To: BOUSHEY,LUCILLE (HP-Cupertino,ex1)
Cc: andreas_hotea@hp.com
Subject: Re: Oracle Quote

The Oracle pricing is below:

Product	Price	Quantity	Extended Price
Oracle10i Database Standard Edition Processor License for 3 years	7,500	4	30,000
Oracle Database Server Support Package for 3 Years	6,000	1	6,000
Oracle Mandory E-Business Discount (license and support)			<1,800>

Oracle pricing contact: Herve Lejeune, herve.lejeune@oracle.com, (650) 506-1894

November 7, 2002

Ms. Lucille Boushey
TPC-C Performance Project Manager
Hewlett Packard
408 447 7364
408 447 5958 FAX

Dear Ms. Boushey:

Per your request I am enclosing the pricing information regarding TUXEDO 8.0 that you requested. This pricing applies to Tuxedo 6.4, 6.5, 7.1, and 8.0. Please note that Tuxedo 8.0 is our most recent version of Tuxedo. Core functionality services pricing is appropriate for your activities. As per the table below HP PA-RISC systems are classified as either a Tier 1, 2, 3, 4 or 5 systems depending on the performance and CPU capacity of the system. This quote is valid for 60 days from the date of this letter.

10.1.1 Tuxedo Core Functionality Services (CFS) Program Product Pricing and Description

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.4, 6.5, 7.1, and 8.0. Prices range from $3,000 for Tier 1 to $250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees. Note that a 5% discount will apply to total list price license purchases less than $100,000 (for instance 6 Tier 1 servers; RP2470 Server 1 cpu configuration– 6 * 3,000 = $18,000 - would be eligible for a 5% license discount). Support is not discountable.

Very Truly Yours,

Rob Gieringer,
Worldwide Pricing Manager

10.1.1.1 BEA Tux/CFS Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 9) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers	**Unlimited**	**$3,000.00**	**$540.00**	**$630.00**
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations with up to 2 CPUs	**Unlimited**	**$12,000.00**	**$2,160.00**	**$2,,520.00**
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	**Unlimited**	**$30,000.00**	**$5,400.00**	**$6,300.00**
Tier 4 - Large (more than 8, less than 32 CPUs)	**Unlimited**	**$100,000.00**	**$18,000.00**	**$21,000.00**
Tier 5 - Massively Parallel Systems, > 32 processors	**Unlimited**	**$250,000.00**	**$45,000.00**	**$52,500.00**

	Tier 1	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Operating System						
HP/UX 9.X;10.X	**Uni-processor Workstation** **B Class - 132/180/2000** **C Class (3000/3600 / 3700)**	**9000/E25** **9000/E35** **9000/E45** **9000/E55** **9000/G30** **9000/G40** **9000/A180** **9000/A180C** **9000 /A400** **RP2470 – 1 CPU** **RP2430**	**9000/G50** **9000/G60** **Multi-Processor Workstations** **J Class (J282/J2240/J5600/J6000/J6700)** **9000/R380,390** **9000/D200,210** **220/30/50/60/80** **D310/20/30** **D350/60/70/80** **9000 /A500** **9000 – L1000** **9000 – R Class** **RP2470 – 2 CPU**	**9000/H20, 30** **9000/H40, 50** **9000/I30, 40** **9000/K1XX** **9000 – L2000/L3000** **9000/I50,60** **9000/H60** **9000/G70** **9000/H70** **9000/I70** **9000/K2XX** **9000/K3XX** **9000/K4XX** **9000/K5XX** **N4xxx Series**	**9000/T500, T520,T600** **1-16 CPUs** **S-Class**	**9000/V series all models** **X-Class** **9000 Series - Superdome**