HP Integrity rx6600 - 4p/8c

using

Red Hat Enterprise Linux AS4

and

Oracle Database 10g Release 2 Enterprise Edition

TPC Benchmark® C
Full Disclosure Report

First Edition

Submitted for Review
October 26, 2006

First Edition - October 26, 2006

Printed in U.S.A., October 26, 2006.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark® C test conducted on the HP Integrity rx6600 - 4p/8c in a client/server configuration, using Oracle Database 10g Release 2 Enterprise Edition and the TUXEDO 8.0 transaction monitor. The operating system used for the benchmark was Red Hat Enterprise Linux AS4. The application was written in C and compiled using HP C++/ANSI C Developer's Bundle /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 Integrity rx6600 - 4p/8c.

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 Integrity rx6600 - 4p/8c	Oracle Database 10g Release 2 Enterprise Edition	Red Hat Enterprise Linux AS4
HP H/W Availability Date - December 1, 2006 S/W Availability Date - December 15, 2006			
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 ($717,016/359440)	
$717,016	**359,440.24 tpmC**	**$1.99 per tpmC**	

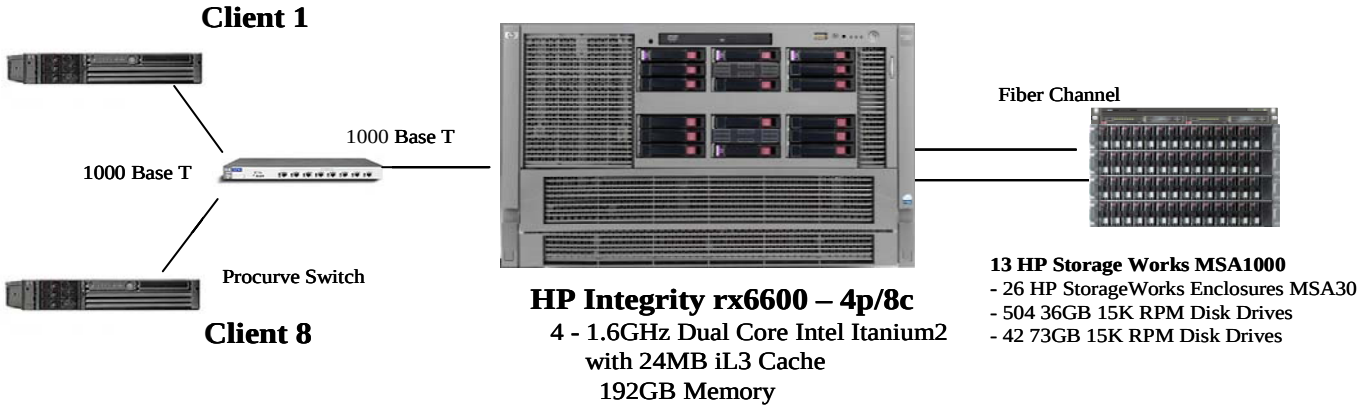
HP Integrity rx6600 - 4p/8c

TPC-C Revision 5.7

Report Date: October 26, 2006

Total System Cost	TPC Throughput	Price/Performance		Availability Date
$717,016	**359,440 tpmC**	**$1.99/tpmC**		**December 15, 2006**
Server Processors/Cores/Threads	Database Manager	Operating System	Other Software	Number of Users
4/8/16 Dual-Core Intel Itanium2 1.6GHz	**Oracle Database 10g Release 2 Enterprise Edition**	**Red Hat Enterprise Linux AS4**	**TUXEDO 8.0**	**283,500**

Server



Clients – 8 hp rx2620 servers

System Components	*Server (HP Integrity rx6600 - 4p/8c)*		*each Client (8 rx2620)*	
	Qty	*Type*	*Qty*	*Type*
Processors/Cores/Threads	4/8/16	1.6GHz Dual-Core Intel Itanium2	2/2/2	1.3GHz Intel Itanium2
Cache Memory	2	24MB iL3 Cache	each	3M iL3 cache
Memory	192	GB	8	GB
Disk Controllers	2	Dual-Port PCI Fibre Channel 4X	1	Ultra2 SCSI LVD
Disk Drives	12	HP StorageWorks Modular Smart Array 1000 with 504 36GB 15K RPM	1	36 GB
	1	HP StorageWorks Modular Smart Array 1000 with 42 73GB 15K RPM		
	4	Internal 72GB/10kpm SAS hard disk drives		
Total Storage		18521.26 GB		
Tape Drives	1	DVD ROM		
Terminals	1	HP Server Thin Client	1	Console Terminal

HP Integrity rx6600 - 4p/8c

TPC-C Rev 5.7

Report Date: October 26, 2006

Description	Part Number	Brand	Price Key	US List Price	Qty	Price	3Year Main.Price
Server Hardware							
HP Integrity rx6600 Base System with 4 1.6GHz/24MB Dual-Core Processor Modules (incl dual-port 1000B-T adapter, 1 power supply)	AD134A, Opt 180		1	43,845	1	43,845	
I/O backplane	AD296A		1	0	1	0	
48-DIMM memory carrier board	AD127A		1	4,495	1	4,495	
72GB/10kpm SAS hard disk drive	AD140A		1	356	4	1,424	
DVD-ROM slimline drive	AD142A		1	230	1	230	
16GB memory quad (4x4GB DIMMS)	AB566A**		1	18,977	12	227,724	
3 Year Svc & Support Price (Hardware and Software)	HA110A3						$7,267
PCI 4GB Fibre Channel Adapter (dual port)	AB379A		1	3,995	2	7,990	
Redundant Power Supply	AD052A		1	749	1	749	
HP Server Thin Client (monitor, keyboard/mouse, cable inc.)	AB300B		1	500	1	500	
Rack Model 5642	358254-B21		1	689	1	689	
				Subtotal		**287,646**	**7,267**
Server Software							
Oracle Database 10g Release 2 Enterprise Edition, Per Processor	Runtime**						
Unlimited Users, 3 years		Oracle	2	20,000	4*	80,000	
Oracle Database Server Support Package for 3 years		Oracle	2	2,000	3		6,000
Red Hat Enterprise Linux AS 4	T2763AA**		1	0	1	0	
RHEL4 Update 4	T2763AA, Opt 004**		1	0	1	0	
Red Hat Linux Subscription License for 3 years	T2763AA, Opt 324**			1,908	1	1,908	
HP Support for Red Hat 7x24	HA107A3-6L4**		1		1		4,839
				Subtotal		**81,908**	**10,839**
Storage							
Rack System/E R3000 XR UPS	AF422A		1	1,366	7	9,562	
HP StorageWorks SAN Switch 2/8 (incl 4 SFP)	288247-B21		1	3,599	3	10,797	
2meter Fibre Optic Cable	221691-B21		1	77	12	924	
5 meter Fibre Optic Cable	221691-B22		1	82	4	328	
HP StorageWorks MSA 1000 (13 + 2 spares)	201723-B22		1	6,995	15	104,925	Included
HP StorageWorks MSA 1000 Controller	218231-B22		1	4,290	1	4,290	
HP StorageWorks Enclosures MSA30 (26 + 3 spares)	302969-B21		1	2,829	29	82,041	
36GB 15K Ultra320 Hard Drive (504 + 51 spares)	286776-B22		1	269	555	149,295	
73GB 15K Ultra320 Hard Drive (42 + 5 spares)	286778-B22		1	399	47	18,753	
10642 (G2) Rack Cabinet	AF001A		1	1,249	4	4,996	
200 - 240 volts North America	A5137AZ AW4		1	94	16	1,504	
				Subtotal		**387,415**	**0**
Client Hardware							
HP server rx2620 w 1.3GHz Intel Itanium 2 Processor	AB333A		1	3,795	8	30,360	
1.3GHz Intel Itanium 2 Processor	AB336A		1	1,650	8	13,200	
3 Year Support Price (Hardware & Software)	HA110A3						31,616
36GB 15K RPM Ultra320 SCSI Internal Disk	AD186A		1	389	8	3,112	
4GB Memory Module (4 x 1GB DIMMS)	AB397A		1	2,550	16	40,800	
HP Server Thin Client (monitor, keyboard/mouse, cable inc.)	AB300B		1	500	1	500	
HP CAT5 Lan Cables (qty 4)	263474-B24		1	29	3	87	
10642 (42U) Rack Cabinet	AF001A		1	1,249	1	1,249	
200-240 Volts Power Option	A5137AZ, Opt AW4		1	94	4	376	
HP-UX Fndn OE DVD Media	B9106AA, Opt. AJR		1	565	1	565	
HP-UX Fndn OE DVD Media, Factory Integrated	B9106AA, Opt. OD1		1	199	8	1,592	
HP-UX Fndn OE Integrity PPL max2CPU w/sys	B9430AC		1	455	16	7,280	
				Subtotal		**99,121**	**31,616**
Client Software							
HP C++/ANSI C Developer's Bundle	B9007AA, Opt. 2AH		1	966	1	966	256
BEA Tuxedo 8.0		Bea Sys.	3	1,200	8	9,600	6,048
				Subtotal		**10,566**	**6,304**
User Connectivity							
HP ProCurve Switch 2724	J4897A		1	1,599	1	1,599	**636**
				Subtotal		**1,599**	**636**
BEA 5% Mandatory Discount			3			(480)	
Oracle Mandatory E-Business Discount			2			(8,600)	
HP's Large configuration Discount and Support Prepayment*			4			($185,714)	($13,107)
				Total		**673,461**	**43,555**

* 2 = 0.50 * 4. Explanation: For the purposes of counting the number of processors which require licensing, an Intel multicore chip with "n" cores shall be determined by multiplying "n" cores by a factor of 0.50.

** These components are not immediately orderable. See the FDR for more information.

*A 24.1% discount was based on the overall value of the specific components from HP (Price Key) in this single quotation. Discounts for similarly sized configurations will be similar to those quoted here, but may vary based on the components in the quotation.

1=HP 2= Oracle (Pricing Contact: MaryBeth Pierantoni (see Appendix F) 3=BEA Systems
4=HP's Large Quantity Pricing

Three Year Cost of Ownership:	**$717,016**
tpmC Rating:	**359,440**
$/tpmC:	**$1.99**

Audited by Lorna Livingtree of Performance Metrics, Inc.

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 Integrity rx6600 - 4p/8c

MQTH, Computed Maximum Qualified Throughput **359,440 tpmC**

Response Times (in seconds)

	90th %-ile	Maximum	Average
New-Order	0.36s	25.51s	0.16s
Payment	0.34s	23.57s	0.16s
Order-Status	0.36s	20.19s	0.16s
Delivery (interactive portion)	0.35s	10.64s	0.09s
Delivery (deferred portion)	0.35s	11.35s	0.14s
Stock-Level	0.35s	23.79s	0.16s
Menu	0.10s	0.09s	0.00s

Transaction Mix, in percent of total transactions

New-Order	44.95%
Payment	43.01%
Order-Status	4.01%
Delivery	4.02%
Stock-Level	4.01%

Keying/Think Times

	Keying Time			Think Time		
	Min	Avg	Max	Min	Avg	Max
New-Order	18.02s	18.03s	18.33s	0.02s	12.14s	202.32s
Payment	3.02s	3.03s	3.29s	0.02s	12.07s	227.67s
Order-Status	2.02s	2.03s	2.27s	0.02s	10.14s	146.95s
Delivery (interactive)	2.02s	2.03s	2.27s	0.02s	5.08s	75.49s
Stock-Level	2.02s	2.03s	2.25s	0.02s	5.07s	83.07s

Test Duration

Ramp up time	80 minutes
Measurement interval	120 minutes
Transactions during measurement interval	95,958,854
Ramp down time	20.336 minutes

Checkpointing

Number of checkpoints in measurement interval	4
Checkpoint Interval	29.33 minutes

TPC Benchmark C Overview

This is the full disclosure report for a benchmark test of the HP Integrity rx6600 - 4p/8c using Oracle Database 10g Release 2 Enterprise Edition . It meets the requirements of the TPC Benchmark® C Standard Specification, Revision 5.7 dated April, 2006.

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 and 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 Developer's Bundle /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 Infrastructure Solutions and Enterprise UNIX 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 Linux operating system parameters values in effect for the configuration used in this benchmark. Also included are all of the Oracle Database 10g Release 2 Enterprise 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 Integrity rx6600 - 4p/8c depicted in Figure 1.1, consisted of:

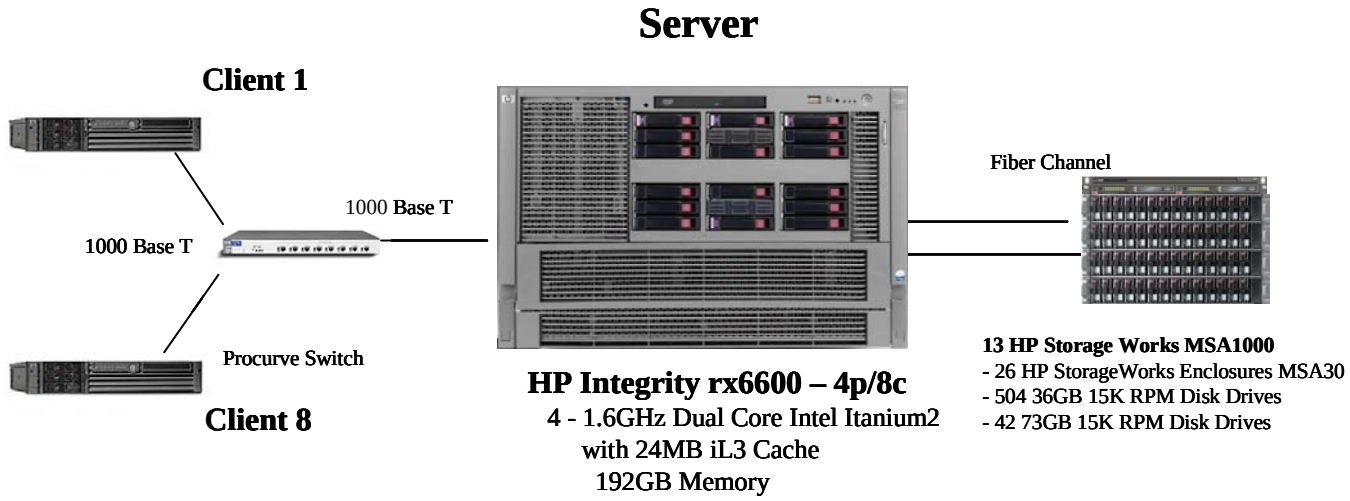
- 4 1.6GHz Dual-Core Intel Itanium2 System Processors

- 192 GB of memory
- 2 Dual-Port PCI Fibre Channel 4X Adapters
- 12 HP StorageWorks Modular Smart Array 1000 with 504 36GB 15K RPM disks
- 1 HP StorageWorks Modular Smart Array 1000 with 42 73GB 15K RPM disks.
- One LAN interface

As indicated in Figure 1.2, this benchmark configuration used Remote Terminal Emulator (RTE) programs that executed on 6 rx2620 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 a 1000 Base-T local area network (LAN) and communicated using TCP/IP. The clients were connected to the SUT via one HP Procurve 2724 switch.

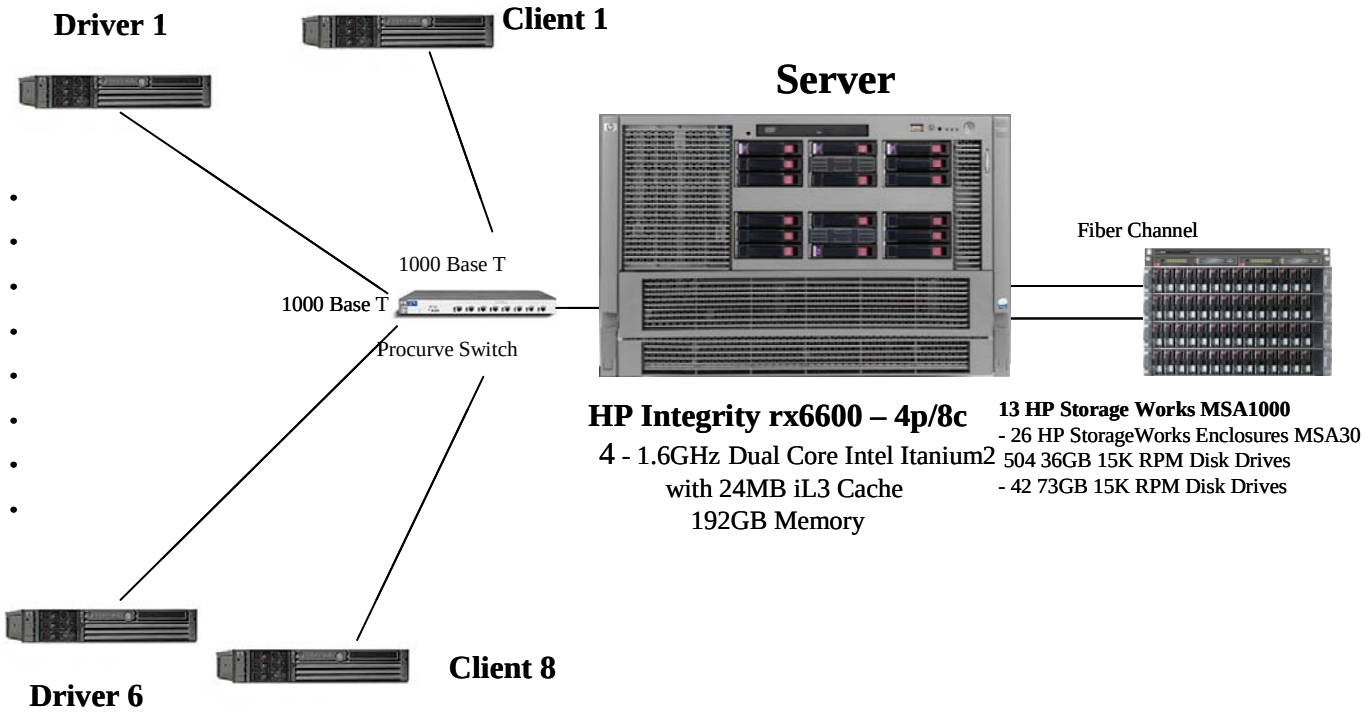
The priced configuration for the HP Integrity rx6600 - 4p/8c is shown in Figure 1.1. In the priced configuration, the RTE shown in the benchmark configuration is replaced by the appropriate number of workstations (emulating ANSI terminals) connected to hubs.

Figure 1.1: HP Integrity rx6600 Priced Configuration



Clients – 8 hp rx2620 servers

Figure 1.2: HP Integrity rx6600 Server Benchmark Configuration



Clients – 8 hp rx2620 servers

Drivers – 6 hp rx2620

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 Oracle Database 10g Release 2 Enterprise 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 Oracle Database 10g Release 2 Enterprise 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 of the database was not used.

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 thin client server 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	99.00%
	Remote warehouse items	1.00%
	Rolled back transactions	1.00%
	Average items per order	10.00
Payment	Home warehouse	85.00%
	Remote warehouse	15.00%
	Non primary key access	59.99%
Order Status	Non primary key access	60.01%
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.95%
	Payment	43.01%
	Order Status	4.01%
	Delivery	4.02%
	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.

These tests were performed on a system configured for 28,350 warehouses. The tests were completed using the same OS (Red Hat Enterprise Linux AS4) and RDBMS (Oracle Database 10G Release 2 Enterprise Edition).

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

These tests were performed on a system configured for 28,350 warehouses. The tests were completed using the same OS (Red Hat Enterprise Linux AS4) and RDBMS (Oracle Database 10G Release 2 Enterprise Edition).

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.

These tests were performed on a system configured for 28,350 warehouses. The tests were completed using the same OS (Red Hat Enterprise Linux AS4) and RDBMS (Oracle Database 10G Release 2 Enterprise Edition).

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.

These tests were performed on a system configured for 28,350 warehouses. The tests were completed using the same OS (Red Hat Enterprise Linux AS4) and RDBMS (Oracle Database 10G Release 2 Enterprise Edition).

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).*

These tests were performed on a system configured for 28,350 warehouses. The tests were completed using the same OS (Red Hat Enterprise Linux AS4) and RDBMS (Oracle Database 10G Release 2 Enterprise Edition).

Both tests were performed under a load of 283,500 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 283,500 users on the full-scale database.

4.5.1 Loss of Log and Data Disks

Because the log 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 data devices are not mirrored and must be restored from backup after failure and subsequent transactions applied from the transaction logs. The tests 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 283,500 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, an individual disk containing recovery log was unplugged from an array.
4. Because of the built-in redundancy in the disk array, the test continued normally.
5. After running again at steady state throughput levels for 5 minutes, an individual disk containing data was unplugged from an array.
6. The test failed with Oracle reporting errors when accessing the files or that array.
7. The disk was replaced.
8. The database files on that array were restored from backup.
9. Oracle was restarted and the transactions in the log were applied to the database.
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 (=1,312,660) matched exactly the number of records for successful New Orders in the RTE "success" file (=1,312,660).
12. Consistency test 3 was run on the database and the results were verified.

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 five minutes at steady state throughout, the server system was 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 (=2,493,208) was 4 more than the number of records for successful New Orders in the RTE "success" file (=2,493,204). *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 32,000 warehouses.

Table	Occurrences
Warehouse	32,000
District	320,000
Customer	960,000,000
History	960,000,000
Orders	288,000,000
New Orders	960,000,000
Order Line	9,600,366,608
Item	100,000
Stock	3,200,000,000

During the measurement all of the warehouses and their associated data were accessed. This was confirmed using D_NEXT_O_ID 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:
==========

The system has 4 internal 72GB SAS disk drives, used as two RAID1 pairs, for OS root, swap and file system.

II) Database files:
==========

We use 12 data arrays and 1 log array. Each of 3 groups of 4 data arrays was connected to the sut through a 2GB 8-port Fibre Channel switch. The log array was connected directly to the host via a 2GB Fibre Channel link.
Each of the 12 data arrays contains 42 33.90GB disk drives for a total capacity of 1423.81GB . The log array contains 42 68.36GB disk drives for a total capacity of 1435.54GB after mirroring.

device path	use
/dev/disk/msa/c5a2dx	data
/dev/disk/msa/c5a3dx	data
/dev/disk/msa/c5a4dx	data
/dev/disk/msa/c6a1dx	data
/dev/disk/msa/c6a2dx	data
/dev/disk/msa/c6a3dx	data
/dev/disk/msa/c6a4dx	data
/dev/disk/msa/c7a1dx	data
/dev/disk/msa/c8a1dx	data
/dev/disk/msa/c8a2dx	data
/dev/disk/msa/c8a3dx	data
/dev/disk/msa/c8a4dx	data
/dev/disk/msa/c7a4d0_p1	log_1_1
/dev/disk/msa/c7a4d1_p1	log_1_2

(the 'x' suffix in the device path denotes a wildcard lun/partition)

The two Oracle logs are on 60GB raw partitions on the log array, leaving 1315.54GB of available capacity. The log array has enough space for 8 hours of redo logs.

Most of the space on the arrays in the tested system was unused during the performance tests, but would be available to satisfy the 8-hour log and 60-day storage requirements.

Each data array was configured identically with nine raid0 scsi luns, each having up to 15 software partitions.
For any given data array, software partition N on lun K was either dedicated to a single database file,
or was part of a 256kb chunk-size software raid0 device, consisting of the same partition N, lun K on
each of the 12 data arrays. A single 60G partition was created on each of the two log array luns.

data array scsi lun formatted capacities

LU 0 120730.55 MB
LU 1 80930.63 MB
LU 2 2458.36 MB
LU 3 19077.19 MB
LU 4 80619.84 MB
LU 5 43640.86 MB
LU 6 9948.98 MB
LU 7 9347.34 MB

LU 8 6143.91 MB

explicit depiction of the assignment of database tables to media:

note to the reader: to find the filesystem name for a database table on a raw partition, you must append _p<pnum> to the path in the dsf column. eg, table stok_0_0 is on dsf /dev/disk/msa/c5a2d0_p1.

psize_mb	size	layout	dsf	lun	pnum	start_mb	end_mb	psize_mb
stok_0_0	8041MB	raw	/dev/disk/msa/c5a2d0	0	1	0.25	8041.25	8041.00MB
stok_0_1	8041MB	raw	/dev/disk/msa/c5a3d0	0	1	0.25	8041.25	8041.00MB
stok_0_2	8041MB	raw	/dev/disk/msa/c5a4d0	0	1	0.25	8041.25	8041.00MB
stok_0_3	8041MB	raw	/dev/disk/msa/c6a1d0	0	1	0.25	8041.25	8041.00MB
stok_0_4	8041MB	raw	/dev/disk/msa/c6a2d0	0	1	0.25	8041.25	8041.00MB
stok_0_5	8041MB	raw	/dev/disk/msa/c6a3d0	0	1	0.25	8041.25	8041.00MB
stok_0_6	8041MB	raw	/dev/disk/msa/c6a4d0	0	1	0.25	8041.25	8041.00MB
stok_0_7	8041MB	raw	/dev/disk/msa/c7a1d0	0	1	0.25	8041.25	8041.00MB
stok_0_8	8041MB	raw	/dev/disk/msa/c8a1d0	0	1	0.25	8041.25	8041.00MB
stok_0_9	8041MB	raw	/dev/disk/msa/c8a2d0	0	1	0.25	8041.25	8041.00MB
stok_0_10	8041MB	raw	/dev/disk/msa/c8a3d0	0	1	0.25	8041.25	8041.00MB
stok_0_11	8041MB	raw	/dev/disk/msa/c8a4d0	0	1	0.25	8041.25	8041.00MB
stok_0_12	8041MB	raw	/dev/disk/msa/c5a2d0	0	2	8041.25	16082.25	8041.00MB
stok_0_13	8041MB	raw	/dev/disk/msa/c5a3d0	0	2	8041.25	16082.25	8041.00MB
stok_0_14	8041MB	raw	/dev/disk/msa/c5a4d0	0	2	8041.25	16082.25	8041.00MB
stok_0_15	8041MB	raw	/dev/disk/msa/c6a1d0	0	2	8041.25	16082.25	8041.00MB
stok_0_16	8041MB	raw	/dev/disk/msa/c6a2d0	0	2	8041.25	16082.25	8041.00MB
stok_0_17	8041MB	raw	/dev/disk/msa/c6a3d0	0	2	8041.25	16082.25	8041.00MB
stok_0_18	8041MB	raw	/dev/disk/msa/c6a4d0	0	2	8041.25	16082.25	8041.00MB
stok_0_19	8041MB	raw	/dev/disk/msa/c7a1d0	0	2	8041.25	16082.25	8041.00MB
stok_0_20	8041MB	raw	/dev/disk/msa/c8a1d0	0	2	8041.25	16082.25	8041.00MB
stok_0_21	8041MB	raw	/dev/disk/msa/c8a2d0	0	2	8041.25	16082.25	8041.00MB
stok_0_22	8041MB	raw	/dev/disk/msa/c8a3d0	0	2	8041.25	16082.25	8041.00MB
stok_0_23	8041MB	raw	/dev/disk/msa/c8a4d0	0	2	8041.25	16082.25	8041.00MB
stok_0_24	8041MB	raw	/dev/disk/msa/c5a2d0	0	3	16082.25	24123.25	8041.00MB
stok_0_25	8041MB	raw	/dev/disk/msa/c5a3d0	0	3	16082.25	24123.25	8041.00MB
stok_0_26	8041MB	raw	/dev/disk/msa/c5a4d0	0	3	16082.25	24123.25	8041.00MB
stok_0_27	8041MB	raw	/dev/disk/msa/c6a1d0	0	3	16082.25	24123.25	8041.00MB
stok_0_28	8041MB	raw	/dev/disk/msa/c6a2d0	0	3	16082.25	24123.25	8041.00MB
stok_0_29	8041MB	raw	/dev/disk/msa/c6a3d0	0	3	16082.25	24123.25	8041.00MB
stok_0_30	8041MB	raw	/dev/disk/msa/c6a4d0	0	3	16082.25	24123.25	8041.00MB
stok_0_31	8041MB	raw	/dev/disk/msa/c7a1d0	0	3	16082.25	24123.25	8041.00MB
stok_0_32	8041MB	raw	/dev/disk/msa/c8a1d0	0	3	16082.25	24123.25	8041.00MB
stok_0_33	8041MB	raw	/dev/disk/msa/c8a2d0	0	3	16082.25	24123.25	8041.00MB
stok_0_34	8041MB	raw	/dev/disk/msa/c8a3d0	0	3	16082.25	24123.25	8041.00MB
stok_0_35	8041MB	raw	/dev/disk/msa/c8a4d0	0	3	16082.25	24123.25	8041.00MB
stok_0_36	8041MB	raw	/dev/disk/msa/c5a2d0	0	4	24123.25	32164.25	8041.00MB
stok_0_37	8041MB	raw	/dev/disk/msa/c5a3d0	0	4	24123.25	32164.25	8041.00MB
stok_0_38	8041MB	raw	/dev/disk/msa/c5a4d0	0	4	24123.25	32164.25	8041.00MB
stok_0_39	8041MB	raw	/dev/disk/msa/c6a1d0	0	4	24123.25	32164.25	8041.00MB
stok_0_40	8041MB	raw	/dev/disk/msa/c6a2d0	0	4	24123.25	32164.25	8041.00MB

stok_0_41	8041MB	raw	/dev/disk/msa/c6a3d0	0	4	24123.25	32164.25	8041.00MB
stok_0_42	8041MB	raw	/dev/disk/msa/c6a4d0	0	4	24123.25	32164.25	8041.00MB
stok_0_43	8041MB	raw	/dev/disk/msa/c7a1d0	0	4	24123.25	32164.25	8041.00MB
stok_0_44	8041MB	raw	/dev/disk/msa/c8a1d0	0	4	24123.25	32164.25	8041.00MB
stok_0_45	8041MB	raw	/dev/disk/msa/c8a2d0	0	4	24123.25	32164.25	8041.00MB
stok_0_46	8041MB	raw	/dev/disk/msa/c8a3d0	0	4	24123.25	32164.25	8041.00MB
stok_0_47	8041MB	raw	/dev/disk/msa/c8a4d0	0	4	24123.25	32164.25	8041.00MB
stok_0_48	8041MB	raw	/dev/disk/msa/c5a2d0	0	5	32164.25	40205.25	8041.00MB
stok_0_49	8041MB	raw	/dev/disk/msa/c5a3d0	0	5	32164.25	40205.25	8041.00MB
stok_0_50	8041MB	raw	/dev/disk/msa/c5a4d0	0	5	32164.25	40205.25	8041.00MB
stok_0_51	8041MB	raw	/dev/disk/msa/c6a1d0	0	5	32164.25	40205.25	8041.00MB
stok_0_52	8041MB	raw	/dev/disk/msa/c6a2d0	0	5	32164.25	40205.25	8041.00MB
stok_0_53	8041MB	raw	/dev/disk/msa/c6a3d0	0	5	32164.25	40205.25	8041.00MB
stok_0_54	8041MB	raw	/dev/disk/msa/c6a4d0	0	5	32164.25	40205.25	8041.00MB
stok_0_55	8041MB	raw	/dev/disk/msa/c7a1d0	0	5	32164.25	40205.25	8041.00MB
stok_0_56	8041MB	raw	/dev/disk/msa/c8a1d0	0	5	32164.25	40205.25	8041.00MB
stok_0_57	8041MB	raw	/dev/disk/msa/c8a2d0	0	5	32164.25	40205.25	8041.00MB
stok_0_58	8041MB	raw	/dev/disk/msa/c8a3d0	0	5	32164.25	40205.25	8041.00MB
stok_0_59	8041MB	raw	/dev/disk/msa/c8a4d0	0	5	32164.25	40205.25	8041.00MB
stok_0_60	8041MB	raw	/dev/disk/msa/c5a2d0	0	6	40205.25	48246.25	8041.00MB
stok_0_61	8041MB	raw	/dev/disk/msa/c5a3d0	0	6	40205.25	48246.25	8041.00MB
stok_0_62	8041MB	raw	/dev/disk/msa/c5a4d0	0	6	40205.25	48246.25	8041.00MB
stok_0_63	8041MB	raw	/dev/disk/msa/c6a1d0	0	6	40205.25	48246.25	8041.00MB
stok_0_64	8041MB	raw	/dev/disk/msa/c6a2d0	0	6	40205.25	48246.25	8041.00MB
stok_0_65	8041MB	raw	/dev/disk/msa/c6a3d0	0	6	40205.25	48246.25	8041.00MB
stok_0_66	8041MB	raw	/dev/disk/msa/c6a4d0	0	6	40205.25	48246.25	8041.00MB
stok_0_67	8041MB	raw	/dev/disk/msa/c7a1d0	0	6	40205.25	48246.25	8041.00MB
stok_0_68	8041MB	raw	/dev/disk/msa/c8a1d0	0	6	40205.25	48246.25	8041.00MB
stok_0_69	8041MB	raw	/dev/disk/msa/c8a2d0	0	6	40205.25	48246.25	8041.00MB
stok_0_70	8041MB	raw	/dev/disk/msa/c8a3d0	0	6	40205.25	48246.25	8041.00MB
stok_0_71	8041MB	raw	/dev/disk/msa/c8a4d0	0	6	40205.25	48246.25	8041.00MB
stok_0_72	8041MB	raw	/dev/disk/msa/c5a2d0	0	7	48246.25	56287.25	8041.00MB
stok_0_73	8041MB	raw	/dev/disk/msa/c5a3d0	0	7	48246.25	56287.25	8041.00MB
stok_0_74	8041MB	raw	/dev/disk/msa/c5a4d0	0	7	48246.25	56287.25	8041.00MB
stok_0_75	8041MB	raw	/dev/disk/msa/c6a1d0	0	7	48246.25	56287.25	8041.00MB
stok_0_76	8041MB	raw	/dev/disk/msa/c6a2d0	0	7	48246.25	56287.25	8041.00MB
stok_0_77	8041MB	raw	/dev/disk/msa/c6a3d0	0	7	48246.25	56287.25	8041.00MB
stok_0_78	8041MB	raw	/dev/disk/msa/c6a4d0	0	7	48246.25	56287.25	8041.00MB
stok_0_79	8041MB	raw	/dev/disk/msa/c7a1d0	0	7	48246.25	56287.25	8041.00MB
stok_0_80	8041MB	raw	/dev/disk/msa/c8a1d0	0	7	48246.25	56287.25	8041.00MB
stok_0_81	8041MB	raw	/dev/disk/msa/c8a2d0	0	7	48246.25	56287.25	8041.00MB
stok_0_82	8041MB	raw	/dev/disk/msa/c8a3d0	0	7	48246.25	56287.25	8041.00MB
stok_0_83	8041MB	raw	/dev/disk/msa/c8a4d0	0	7	48246.25	56287.25	8041.00MB
stok_0_84	8041MB	raw	/dev/disk/msa/c5a2d0	0	8	56287.25	64328.25	8041.00MB
stok_0_85	8041MB	raw	/dev/disk/msa/c5a3d0	0	8	56287.25	64328.25	8041.00MB
stok_0_86	8041MB	raw	/dev/disk/msa/c5a4d0	0	8	56287.25	64328.25	8041.00MB
stok_0_87	8041MB	raw	/dev/disk/msa/c6a1d0	0	8	56287.25	64328.25	8041.00MB
stok_0_88	8041MB	raw	/dev/disk/msa/c6a2d0	0	8	56287.25	64328.25	8041.00MB
stok_0_89	8041MB	raw	/dev/disk/msa/c6a3d0	0	8	56287.25	64328.25	8041.00MB
stok_0_90	8041MB	raw	/dev/disk/msa/c6a4d0	0	8	56287.25	64328.25	8041.00MB

stok_0_91	8041MB	raw	/dev/disk/msa/c7a1d0	0	8	56287.25	64328.25	8041.00MB
stok_0_92	8041MB	raw	/dev/disk/msa/c8a1d0	0	8	56287.25	64328.25	8041.00MB
stok_0_93	8041MB	raw	/dev/disk/msa/c8a2d0	0	8	56287.25	64328.25	8041.00MB
stok_0_94	8041MB	raw	/dev/disk/msa/c8a3d0	0	8	56287.25	64328.25	8041.00MB
stok_0_95	8041MB	raw	/dev/disk/msa/c8a4d0	0	8	56287.25	64328.25	8041.00MB
stok_0_96	8041MB	raw	/dev/disk/msa/c5a2d0	0	9	64328.25	72369.25	8041.00MB
stok_0_97	8041MB	raw	/dev/disk/msa/c5a3d0	0	9	64328.25	72369.25	8041.00MB
stok_0_98	8041MB	raw	/dev/disk/msa/c5a4d0	0	9	64328.25	72369.25	8041.00MB
stok_0_99	8041MB	raw	/dev/disk/msa/c6a1d0	0	9	64328.25	72369.25	8041.00MB
stok_0_100	8041MB	raw	/dev/disk/msa/c6a2d0	0	9	64328.25	72369.25	8041.00MB
stok_0_101	8041MB	raw	/dev/disk/msa/c6a3d0	0	9	64328.25	72369.25	8041.00MB
stok_0_102	8041MB	raw	/dev/disk/msa/c6a4d0	0	9	64328.25	72369.25	8041.00MB
stok_0_103	8041MB	raw	/dev/disk/msa/c7a1d0	0	9	64328.25	72369.25	8041.00MB
stok_0_104	8041MB	raw	/dev/disk/msa/c8a1d0	0	9	64328.25	72369.25	8041.00MB
stok_0_105	8041MB	raw	/dev/disk/msa/c8a2d0	0	9	64328.25	72369.25	8041.00MB
stok_0_106	8041MB	raw	/dev/disk/msa/c8a3d0	0	9	64328.25	72369.25	8041.00MB
stok_0_107	8041MB	raw	/dev/disk/msa/c8a4d0	0	9	64328.25	72369.25	8041.00MB
stok_0_108	8041MB	raw	/dev/disk/msa/c5a2d0	0	10	72369.25	80410.25	8041.00MB
stok_0_109	8041MB	raw	/dev/disk/msa/c5a3d0	0	10	72369.25	80410.25	8041.00MB
stok_0_110	8041MB	raw	/dev/disk/msa/c5a4d0	0	10	72369.25	80410.25	8041.00MB
stok_0_111	8041MB	raw	/dev/disk/msa/c6a1d0	0	10	72369.25	80410.25	8041.00MB
stok_0_112	8041MB	raw	/dev/disk/msa/c6a2d0	0	10	72369.25	80410.25	8041.00MB
stok_0_113	8041MB	raw	/dev/disk/msa/c6a3d0	0	10	72369.25	80410.25	8041.00MB
stok_0_114	8041MB	raw	/dev/disk/msa/c6a4d0	0	10	72369.25	80410.25	8041.00MB
stok_0_115	8041MB	raw	/dev/disk/msa/c7a1d0	0	10	72369.25	80410.25	8041.00MB
stok_0_116	8041MB	raw	/dev/disk/msa/c8a1d0	0	10	72369.25	80410.25	8041.00MB
stok_0_117	8041MB	raw	/dev/disk/msa/c8a2d0	0	10	72369.25	80410.25	8041.00MB
stok_0_118	8041MB	raw	/dev/disk/msa/c8a3d0	0	10	72369.25	80410.25	8041.00MB
stok_0_119	8041MB	raw	/dev/disk/msa/c8a4d0	0	10	72369.25	80410.25	8041.00MB
stok_0_120	8041MB	raw	/dev/disk/msa/c5a2d0	0	11	80410.25	88451.25	8041.00MB
stok_0_121	8041MB	raw	/dev/disk/msa/c5a3d0	0	11	80410.25	88451.25	8041.00MB
stok_0_122	8041MB	raw	/dev/disk/msa/c5a4d0	0	11	80410.25	88451.25	8041.00MB
stok_0_123	8041MB	raw	/dev/disk/msa/c6a1d0	0	11	80410.25	88451.25	8041.00MB
stok_0_124	8041MB	raw	/dev/disk/msa/c6a2d0	0	11	80410.25	88451.25	8041.00MB
stok_0_125	8041MB	raw	/dev/disk/msa/c6a3d0	0	11	80410.25	88451.25	8041.00MB
stok_0_126	8041MB	raw	/dev/disk/msa/c6a4d0	0	11	80410.25	88451.25	8041.00MB
stok_0_127	8041MB	raw	/dev/disk/msa/c7a1d0	0	11	80410.25	88451.25	8041.00MB
stok_0_128	8041MB	raw	/dev/disk/msa/c8a1d0	0	11	80410.25	88451.25	8041.00MB
stok_0_129	8041MB	raw	/dev/disk/msa/c8a2d0	0	11	80410.25	88451.25	8041.00MB
stok_0_130	8041MB	raw	/dev/disk/msa/c8a3d0	0	11	80410.25	88451.25	8041.00MB
stok_0_131	8041MB	raw	/dev/disk/msa/c8a4d0	0	11	80410.25	88451.25	8041.00MB
stok_0_132	8041MB	raw	/dev/disk/msa/c5a2d0	0	12	88451.25	96492.25	8041.00MB
stok_0_133	8041MB	raw	/dev/disk/msa/c5a3d0	0	12	88451.25	96492.25	8041.00MB
stok_0_134	8041MB	raw	/dev/disk/msa/c5a4d0	0	12	88451.25	96492.25	8041.00MB
stok_0_135	8041MB	raw	/dev/disk/msa/c6a1d0	0	12	88451.25	96492.25	8041.00MB
stok_0_136	8041MB	raw	/dev/disk/msa/c6a2d0	0	12	88451.25	96492.25	8041.00MB
stok_0_137	8041MB	raw	/dev/disk/msa/c6a3d0	0	12	88451.25	96492.25	8041.00MB
stok_0_138	8041MB	raw	/dev/disk/msa/c6a4d0	0	12	88451.25	96492.25	8041.00MB
stok_0_139	8041MB	raw	/dev/disk/msa/c7a1d0	0	12	88451.25	96492.25	8041.00MB
stok_0_140	8041MB	raw	/dev/disk/msa/c8a1d0	0	12	88451.25	96492.25	8041.00MB

stok_0_141	8041MB	raw	/dev/disk/msa/c8a2d0	0	12	88451.25	96492.25	8041.00MB
stok_0_142	8041MB	raw	/dev/disk/msa/c8a3d0	0	12	88451.25	96492.25	8041.00MB
stok_0_143	8041MB	raw	/dev/disk/msa/c8a4d0	0	12	88451.25	96492.25	8041.00MB
stok_0_144	8041MB	raw	/dev/disk/msa/c5a2d0	0	13	96492.25	104533.3	8041.00MB
stok_0_145	8041MB	raw	/dev/disk/msa/c5a3d0	0	13	96492.25	104533.3	8041.00MB
stok_0_146	8041MB	raw	/dev/disk/msa/c5a4d0	0	13	96492.25	104533.3	8041.00MB
stok_0_147	8041MB	raw	/dev/disk/msa/c6a1d0	0	13	96492.25	104533.3	8041.00MB
stok_0_148	8041MB	raw	/dev/disk/msa/c6a2d0	0	13	96492.25	104533.3	8041.00MB
stok_0_149	8041MB	raw	/dev/disk/msa/c6a3d0	0	13	96492.25	104533.3	8041.00MB
stok_0_150	8041MB	raw	/dev/disk/msa/c6a4d0	0	13	96492.25	104533.3	8041.00MB
stok_0_151	8041MB	raw	/dev/disk/msa/c7a1d0	0	13	96492.25	104533.3	8041.00MB
stok_0_152	8041MB	raw	/dev/disk/msa/c8a1d0	0	13	96492.25	104533.3	8041.00MB
stok_0_153	8041MB	raw	/dev/disk/msa/c8a2d0	0	13	96492.25	104533.3	8041.00MB
stok_0_154	8041MB	raw	/dev/disk/msa/c8a3d0	0	13	96492.25	104533.3	8041.00MB
stok_0_155	8041MB	raw	/dev/disk/msa/c8a4d0	0	13	96492.25	104533.3	8041.00MB
stok_0_156	8041MB	raw	/dev/disk/msa/c5a2d0	0	14	104533.3	112574.3	8041.00MB
stok_0_157	8041MB	raw	/dev/disk/msa/c5a3d0	0	14	104533.3	112574.3	8041.00MB
stok_0_158	8041MB	raw	/dev/disk/msa/c5a4d0	0	14	104533.3	112574.3	8041.00MB
stok_0_159	8041MB	raw	/dev/disk/msa/c6a1d0	0	14	104533.3	112574.3	8041.00MB
stok_0_160	8041MB	raw	/dev/disk/msa/c6a2d0	0	14	104533.3	112574.3	8041.00MB
stok_0_161	8041MB	raw	/dev/disk/msa/c6a3d0	0	14	104533.3	112574.3	8041.00MB
stok_0_162	8041MB	raw	/dev/disk/msa/c6a4d0	0	14	104533.3	112574.3	8041.00MB
stok_0_163	8041MB	raw	/dev/disk/msa/c7a1d0	0	14	104533.3	112574.3	8041.00MB
stok_0_164	8041MB	raw	/dev/disk/msa/c8a1d0	0	14	104533.3	112574.3	8041.00MB
stok_0_165	8041MB	raw	/dev/disk/msa/c8a2d0	0	14	104533.3	112574.3	8041.00MB
stok_0_166	8041MB	raw	/dev/disk/msa/c8a3d0	0	14	104533.3	112574.3	8041.00MB
cust_0_0	8091MB	raw	/dev/disk/msa/c8a4d0	0	14	104533.3	112624.3	8091.00MB
cust_0_1	8091MB	raw	/dev/disk/msa/c5a2d0	0	15	112574.3	120665.3	8091.00MB
cust_0_2	8091MB	raw	/dev/disk/msa/c5a3d0	0	15	112574.3	120665.3	8091.00MB
cust_0_3	8091MB	raw	/dev/disk/msa/c5a4d0	0	15	112574.3	120665.3	8091.00MB
cust_0_4	8091MB	raw	/dev/disk/msa/c6a1d0	0	15	112574.3	120665.3	8091.00MB
cust_0_5	8091MB	raw	/dev/disk/msa/c6a2d0	0	15	112574.3	120665.3	8091.00MB
cust_0_6	8091MB	raw	/dev/disk/msa/c6a3d0	0	15	112574.3	120665.3	8091.00MB
cust_0_7	8091MB	raw	/dev/disk/msa/c6a4d0	0	15	112574.3	120665.3	8091.00MB
cust_0_8	8091MB	raw	/dev/disk/msa/c7a1d0	0	15	112574.3	120665.3	8091.00MB
cust_0_9	8091MB	raw	/dev/disk/msa/c8a1d0	0	15	112574.3	120665.3	8091.00MB
cust_0_10	8091MB	raw	/dev/disk/msa/c8a2d0	0	15	112574.3	120665.3	8091.00MB
cust_0_11	8091MB	raw	/dev/disk/msa/c8a3d0	0	15	112574.3	120665.3	8091.00MB
cust_0_12	8091MB	raw	/dev/disk/msa/c8a4d0	0	15	112624.3	120715.3	8091.00MB
cust_0_13	8091MB	raw	/dev/disk/msa/c5a2d1	1	1	0.25	8091.25	8091.00MB
cust_0_14	8091MB	raw	/dev/disk/msa/c5a3d1	1	1	0.25	8091.25	8091.00MB
cust_0_15	8091MB	raw	/dev/disk/msa/c5a4d1	1	1	0.25	8091.25	8091.00MB
cust_0_16	8091MB	raw	/dev/disk/msa/c6a1d1	1	1	0.25	8091.25	8091.00MB
cust_0_17	8091MB	raw	/dev/disk/msa/c6a2d1	1	1	0.25	8091.25	8091.00MB
cust_0_18	8091MB	raw	/dev/disk/msa/c6a3d1	1	1	0.25	8091.25	8091.00MB
cust_0_19	8091MB	raw	/dev/disk/msa/c6a4d1	1	1	0.25	8091.25	8091.00MB
cust_0_20	8091MB	raw	/dev/disk/msa/c7a1d1	1	1	0.25	8091.25	8091.00MB
cust_0_21	8091MB	raw	/dev/disk/msa/c8a1d1	1	1	0.25	8091.25	8091.00MB
cust_0_22	8091MB	raw	/dev/disk/msa/c8a2d1	1	1	0.25	8091.25	8091.00MB
cust_0_23	8091MB	raw	/dev/disk/msa/c8a3d1	1	1	0.25	8091.25	8091.00MB

cust_0_24	8091MB	raw	/dev/disk/msa/c8a4d1	1	1	0.25	8091.25	8091.00MB
cust_0_25	8091MB	raw	/dev/disk/msa/c5a2d1	1	2	8091.25	16182.25	8091.00MB
cust_0_26	8091MB	raw	/dev/disk/msa/c5a3d1	1	2	8091.25	16182.25	8091.00MB
cust_0_27	8091MB	raw	/dev/disk/msa/c5a4d1	1	2	8091.25	16182.25	8091.00MB
cust_0_28	8091MB	raw	/dev/disk/msa/c6a1d1	1	2	8091.25	16182.25	8091.00MB
cust_0_29	8091MB	raw	/dev/disk/msa/c6a2d1	1	2	8091.25	16182.25	8091.00MB
cust_0_30	8091MB	raw	/dev/disk/msa/c6a3d1	1	2	8091.25	16182.25	8091.00MB
cust_0_31	8091MB	raw	/dev/disk/msa/c6a4d1	1	2	8091.25	16182.25	8091.00MB
cust_0_32	8091MB	raw	/dev/disk/msa/c7a1d1	1	2	8091.25	16182.25	8091.00MB
cust_0_33	8091MB	raw	/dev/disk/msa/c8a1d1	1	2	8091.25	16182.25	8091.00MB
cust_0_34	8091MB	raw	/dev/disk/msa/c8a2d1	1	2	8091.25	16182.25	8091.00MB
cust_0_35	8091MB	raw	/dev/disk/msa/c8a3d1	1	2	8091.25	16182.25	8091.00MB
cust_0_36	8091MB	raw	/dev/disk/msa/c8a4d1	1	2	8091.25	16182.25	8091.00MB
cust_0_37	8091MB	raw	/dev/disk/msa/c5a2d1	1	3	16182.25	24273.25	8091.00MB
cust_0_38	8091MB	raw	/dev/disk/msa/c5a3d1	1	3	16182.25	24273.25	8091.00MB
cust_0_39	8091MB	raw	/dev/disk/msa/c5a4d1	1	3	16182.25	24273.25	8091.00MB
cust_0_40	8091MB	raw	/dev/disk/msa/c6a1d1	1	3	16182.25	24273.25	8091.00MB
cust_0_41	8091MB	raw	/dev/disk/msa/c6a2d1	1	3	16182.25	24273.25	8091.00MB
cust_0_42	8091MB	raw	/dev/disk/msa/c6a3d1	1	3	16182.25	24273.25	8091.00MB
cust_0_43	8091MB	raw	/dev/disk/msa/c6a4d1	1	3	16182.25	24273.25	8091.00MB
cust_0_44	8091MB	raw	/dev/disk/msa/c7a1d1	1	3	16182.25	24273.25	8091.00MB
cust_0_45	8091MB	raw	/dev/disk/msa/c8a1d1	1	3	16182.25	24273.25	8091.00MB
cust_0_46	8091MB	raw	/dev/disk/msa/c8a2d1	1	3	16182.25	24273.25	8091.00MB
cust_0_47	8091MB	raw	/dev/disk/msa/c8a3d1	1	3	16182.25	24273.25	8091.00MB
cust_0_48	8091MB	raw	/dev/disk/msa/c8a4d1	1	3	16182.25	24273.25	8091.00MB
cust_0_49	8091MB	raw	/dev/disk/msa/c5a2d1	1	4	24273.25	32364.25	8091.00MB
cust_0_50	8091MB	raw	/dev/disk/msa/c5a3d1	1	4	24273.25	32364.25	8091.00MB
cust_0_51	8091MB	raw	/dev/disk/msa/c5a4d1	1	4	24273.25	32364.25	8091.00MB
cust_0_52	8091MB	raw	/dev/disk/msa/c6a1d1	1	4	24273.25	32364.25	8091.00MB
cust_0_53	8091MB	raw	/dev/disk/msa/c6a2d1	1	4	24273.25	32364.25	8091.00MB
cust_0_54	8091MB	raw	/dev/disk/msa/c6a3d1	1	4	24273.25	32364.25	8091.00MB
cust_0_55	8091MB	raw	/dev/disk/msa/c6a4d1	1	4	24273.25	32364.25	8091.00MB
cust_0_56	8091MB	raw	/dev/disk/msa/c7a1d1	1	4	24273.25	32364.25	8091.00MB
cust_0_57	8091MB	raw	/dev/disk/msa/c8a1d1	1	4	24273.25	32364.25	8091.00MB
cust_0_58	8091MB	raw	/dev/disk/msa/c8a2d1	1	4	24273.25	32364.25	8091.00MB
cust_0_59	8091MB	raw	/dev/disk/msa/c8a3d1	1	4	24273.25	32364.25	8091.00MB
cust_0_60	8091MB	raw	/dev/disk/msa/c8a4d1	1	4	24273.25	32364.25	8091.00MB
cust_0_61	8091MB	raw	/dev/disk/msa/c5a2d1	1	5	32364.25	40455.25	8091.00MB
cust_0_62	8091MB	raw	/dev/disk/msa/c5a3d1	1	5	32364.25	40455.25	8091.00MB
cust_0_63	8091MB	raw	/dev/disk/msa/c5a4d1	1	5	32364.25	40455.25	8091.00MB
cust_0_64	8091MB	raw	/dev/disk/msa/c6a1d1	1	5	32364.25	40455.25	8091.00MB
cust_0_65	8091MB	raw	/dev/disk/msa/c6a2d1	1	5	32364.25	40455.25	8091.00MB
cust_0_66	8091MB	raw	/dev/disk/msa/c6a3d1	1	5	32364.25	40455.25	8091.00MB
cust_0_67	8091MB	raw	/dev/disk/msa/c6a4d1	1	5	32364.25	40455.25	8091.00MB
cust_0_68	8091MB	raw	/dev/disk/msa/c7a1d1	1	5	32364.25	40455.25	8091.00MB
cust_0_69	8091MB	raw	/dev/disk/msa/c8a1d1	1	5	32364.25	40455.25	8091.00MB
cust_0_70	8091MB	raw	/dev/disk/msa/c8a2d1	1	5	32364.25	40455.25	8091.00MB
cust_0_71	8091MB	raw	/dev/disk/msa/c8a3d1	1	5	32364.25	40455.25	8091.00MB
cust_0_72	8091MB	raw	/dev/disk/msa/c8a4d1	1	5	32364.25	40455.25	8091.00MB
cust_0_73	8091MB	raw	/dev/disk/msa/c5a2d1	1	6	40455.25	48546.25	8091.00MB

cust_0_74	8091MB	raw	/dev/disk/msa/c5a3d1	1	6	40455.25	48546.25	8091.00MB
cust_0_75	8091MB	raw	/dev/disk/msa/c5a4d1	1	6	40455.25	48546.25	8091.00MB
cust_0_76	8091MB	raw	/dev/disk/msa/c6a1d1	1	6	40455.25	48546.25	8091.00MB
cust_0_77	8091MB	raw	/dev/disk/msa/c6a2d1	1	6	40455.25	48546.25	8091.00MB
cust_0_78	8091MB	raw	/dev/disk/msa/c6a3d1	1	6	40455.25	48546.25	8091.00MB
cust_0_79	8091MB	raw	/dev/disk/msa/c6a4d1	1	6	40455.25	48546.25	8091.00MB
cust_0_80	8091MB	raw	/dev/disk/msa/c7a1d1	1	6	40455.25	48546.25	8091.00MB
cust_0_81	8091MB	raw	/dev/disk/msa/c8a1d1	1	6	40455.25	48546.25	8091.00MB
cust_0_82	8091MB	raw	/dev/disk/msa/c8a2d1	1	6	40455.25	48546.25	8091.00MB
cust_0_83	8091MB	raw	/dev/disk/msa/c8a3d1	1	6	40455.25	48546.25	8091.00MB
cust_0_84	8091MB	raw	/dev/disk/msa/c8a4d1	1	6	40455.25	48546.25	8091.00MB
cust_0_85	8091MB	raw	/dev/disk/msa/c5a2d1	1	7	48546.25	56637.25	8091.00MB
cust_0_86	8091MB	raw	/dev/disk/msa/c5a3d1	1	7	48546.25	56637.25	8091.00MB
cust_0_87	8091MB	raw	/dev/disk/msa/c5a4d1	1	7	48546.25	56637.25	8091.00MB
cust_0_88	8091MB	raw	/dev/disk/msa/c6a1d1	1	7	48546.25	56637.25	8091.00MB
cust_0_89	8091MB	raw	/dev/disk/msa/c6a2d1	1	7	48546.25	56637.25	8091.00MB
cust_0_90	8091MB	raw	/dev/disk/msa/c6a3d1	1	7	48546.25	56637.25	8091.00MB
cust_0_91	8091MB	raw	/dev/disk/msa/c6a4d1	1	7	48546.25	56637.25	8091.00MB
cust_0_92	8091MB	raw	/dev/disk/msa/c7a1d1	1	7	48546.25	56637.25	8091.00MB
cust_0_93	8091MB	raw	/dev/disk/msa/c8a1d1	1	7	48546.25	56637.25	8091.00MB
cust_0_94	8091MB	raw	/dev/disk/msa/c8a2d1	1	7	48546.25	56637.25	8091.00MB
cust_0_95	8091MB	raw	/dev/disk/msa/c8a3d1	1	7	48546.25	56637.25	8091.00MB
cust_0_96	8091MB	raw	/dev/disk/msa/c8a4d1	1	7	48546.25	56637.25	8091.00MB
cust_0_97	8091MB	raw	/dev/disk/msa/c5a2d1	1	8	56637.25	64728.25	8091.00MB
cust_0_98	8091MB	raw	/dev/disk/msa/c5a3d1	1	8	56637.25	64728.25	8091.00MB
cust_0_99	8091MB	raw	/dev/disk/msa/c5a4d1	1	8	56637.25	64728.25	8091.00MB
cust_0_100	8091MB	raw	/dev/disk/msa/c6a1d1	1	8	56637.25	64728.25	8091.00MB
cust_0_101	8091MB	raw	/dev/disk/msa/c6a2d1	1	8	56637.25	64728.25	8091.00MB
cust_0_102	8091MB	raw	/dev/disk/msa/c6a3d1	1	8	56637.25	64728.25	8091.00MB
cust_0_103	8091MB	raw	/dev/disk/msa/c6a4d1	1	8	56637.25	64728.25	8091.00MB
cust_0_104	8091MB	raw	/dev/disk/msa/c7a1d1	1	8	56637.25	64728.25	8091.00MB
cust_0_105	8091MB	raw	/dev/disk/msa/c8a1d1	1	8	56637.25	64728.25	8091.00MB
cust_0_106	8091MB	raw	/dev/disk/msa/c8a2d1	1	8	56637.25	64728.25	8091.00MB
cust_0_107	8091MB	raw	/dev/disk/msa/c8a3d1	1	8	56637.25	64728.25	8091.00MB
cust_0_108	8091MB	raw	/dev/disk/msa/c8a4d1	1	8	56637.25	64728.25	8091.00MB
cust_0_109	8091MB	raw	/dev/disk/msa/c5a2d1	1	9	64728.25	72819.25	8091.00MB
cust_0_110	8091MB	raw	/dev/disk/msa/c5a3d1	1	9	64728.25	72819.25	8091.00MB
cust_0_111	8091MB	raw	/dev/disk/msa/c5a4d1	1	9	64728.25	72819.25	8091.00MB
cust_0_112	8091MB	raw	/dev/disk/msa/c6a1d1	1	9	64728.25	72819.25	8091.00MB
cust_0_113	8091MB	raw	/dev/disk/msa/c6a2d1	1	9	64728.25	72819.25	8091.00MB
cust_0_114	8091MB	raw	/dev/disk/msa/c6a3d1	1	9	64728.25	72819.25	8091.00MB
cust_0_115	8091MB	raw	/dev/disk/msa/c6a4d1	1	9	64728.25	72819.25	8091.00MB
cust_0_116	8091MB	raw	/dev/disk/msa/c7a1d1	1	9	64728.25	72819.25	8091.00MB
cust_0_117	8091MB	raw	/dev/disk/msa/c8a1d1	1	9	64728.25	72819.25	8091.00MB
cust_0_118	8091MB	raw	/dev/disk/msa/c8a2d1	1	9	64728.25	72819.25	8091.00MB
cust_0_119	8091MB	raw	/dev/disk/msa/c8a3d1	1	9	64728.25	72819.25	8091.00MB
cust_0_120	8091MB	raw	/dev/disk/msa/c8a4d1	1	9	64728.25	72819.25	8091.00MB
cust_0_121	8091MB	raw	/dev/disk/msa/c5a2d1	1	10	72819.25	80910.25	8091.00MB
cust_0_122	8091MB	raw	/dev/disk/msa/c5a3d1	1	10	72819.25	80910.25	8091.00MB
cust_0_123	8091MB	raw	/dev/disk/msa/c5a4d1	1	10	72819.25	80910.25	8091.00MB

cust_0_124	8091MB	raw	/dev/disk/msa/c6a1d1	1	10	72819.25	80910.25	8091.00MB
cust_0_125	8091MB	raw	/dev/disk/msa/c6a2d1	1	10	72819.25	80910.25	8091.00MB
cust_0_126	8091MB	raw	/dev/disk/msa/c6a3d1	1	10	72819.25	80910.25	8091.00MB
cust_0_127	8091MB	raw	/dev/disk/msa/c6a4d1	1	10	72819.25	80910.25	8091.00MB
cust_0_128	8091MB	raw	/dev/disk/msa/c7a1d1	1	10	72819.25	80910.25	8091.00MB
cust_0_129	8091MB	raw	/dev/disk/msa/c8a1d1	1	10	72819.25	80910.25	8091.00MB
cust_0_130	8091MB	raw	/dev/disk/msa/c8a2d1	1	10	72819.25	80910.25	8091.00MB
cust_0_131	8091MB	raw	/dev/disk/msa/c8a3d1	1	10	72819.25	80910.25	8091.00MB
cust_0_132	8091MB	raw	/dev/disk/msa/c8a4d1	1	10	72819.25	80910.25	8091.00MB
nord_0_0	5591MB	striped	/dev/md0	2	1	0.25	466.5	466.25MB
nord_0_1	5591MB	striped	/dev/md1	2	2	466.5	932.75	466.25MB
item_0_0	21MB	striped	/dev/md2	2	3	932.75	934.75	2.00MB
iware_0_0	51MB	striped	/dev/md3	2	4	934.75	939.25	4.50MB
idist_0_0	161MB	striped	/dev/md4	2	5	939.25	953	13.75MB
iitem_0_0	21MB	striped	/dev/md5	2	6	953	955	2.00MB
ware_0_0	71MB	striped	/dev/md6	2	7	955	961.25	6.25MB
dist_0_0	681MB	striped	/dev/md7	2	8	961.25	1018.25	57.00MB
system_1	401MB	striped	/dev/md8	2	9	1018.25	1052	33.75MB
sp_0	2049MB	striped	/dev/md9	2	10	1052	1223	171.00MB
tpccaux	121MB	striped	/dev/md10	2	11	1223	1233.5	10.50MB
control_001	26MB	striped	/dev/md11	2	12	1233.5	1236	2.50MB
control_002	26MB	striped	/dev/md12	2	13	1236	1238.5	2.50MB
iordr2_0_0	7141MB	striped	/dev/md13	2	14	1238.5	1834	595.50MB
iordr2_0_1	7141MB	striped	/dev/md14	2	15	1834	2429.5	595.50MB
iordr2_0_2	7141MB	striped	/dev/md15	3	1	0.25	595.75	595.50MB
iordr2_0_3	7141MB	striped	/dev/md16	3	2	595.75	1191.25	595.50MB
iordr2_0_4	7141MB	striped	/dev/md17	3	3	1191.25	1786.75	595.50MB
iordr2_0_5	7141MB	striped	/dev/md18	3	4	1786.75	2382.25	595.50MB
iordr2_0_6	7141MB	striped	/dev/md19	3	5	2382.25	2977.75	595.50MB
iordr2_0_7	7141MB	striped	/dev/md20	3	6	2977.75	3573.25	595.50MB
icust2_0_0	8111MB	striped	/dev/md21	3	7	3573.25	4249.5	676.25MB
icust2_0_1	8111MB	striped	/dev/md22	3	8	4249.5	4925.75	676.25MB
icust2_0_2	8111MB	striped	/dev/md23	3	9	4925.75	5602	676.25MB
icust2_0_3	8111MB	striped	/dev/md24	3	10	5602	6278.25	676.25MB
icust2_0_4	8111MB	striped	/dev/md25	3	11	6278.25	6954.5	676.25MB
icust2_0_5	8111MB	striped	/dev/md26	3	12	6954.5	7630.75	676.25MB
icust2_0_6	8111MB	striped	/dev/md27	3	13	7630.75	8307	676.25MB
ordr_0_0	64461MB	striped	/dev/md28	3	14	8307	13679	5372.00MB
ordr_0_1	64461MB	striped	/dev/md29	3	15	13679	19051	5372.00MB
ordr_0_2	64461MB	striped	/dev/md30	4	1	0.25	5372.25	5372.00MB
ordr_0_3	64461MB	striped	/dev/md31	4	2	5372.25	10744.25	5372.00MB
ordr_0_4	64461MB	striped	/dev/md32	4	3	10744.25	16116.25	5372.00MB
ordr_0_5	64461MB	striped	/dev/md33	4	4	16116.25	21488.25	5372.00MB
ordr_0_6	64461MB	striped	/dev/md34	4	5	21488.25	26860.25	5372.00MB
ordr_0_7	64461MB	striped	/dev/md35	4	6	26860.25	32232.25	5372.00MB
ordr_0_8	64461MB	striped	/dev/md36	4	7	32232.25	37604.25	5372.00MB
ordr_0_9	64461MB	striped	/dev/md37	4	8	37604.25	42976.25	5372.00MB
ordr_0_10	64461MB	striped	/dev/md38	4	9	42976.25	48348.25	5372.00MB
ordr_0_11	64461MB	striped	/dev/md39	4	10	48348.25	53720.25	5372.00MB
ordr_0_12	64461MB	striped	/dev/md40	4	11	53720.25	59092.25	5372.00MB

ordr_0_13	64461MB	striped	/dev/md41	4	12	59092.25	64464.25	5372.00MB
ordr_0_14	64461MB	striped	/dev/md42	4	13	64464.25	69836.25	5372.00MB
ordr_0_15	64461MB	striped	/dev/md43	4	14	69836.25	75208.25	5372.00MB
ordr_0_16	64461MB	striped	/dev/md44	4	15	75208.25	80580.25	5372.00MB
ordr_0_17	64461MB	striped	/dev/md45	5	1	0.25	5372.25	5372.00MB
ordr_0_18	64461MB	striped	/dev/md46	5	2	5372.25	10744.25	5372.00MB
ordr_0_19	64461MB	striped	/dev/md47	5	3	10744.25	16116.25	5372.00MB
ordr_0_20	64461MB	striped	/dev/md48	5	4	16116.25	21488.25	5372.00MB
ordr_0_21	64461MB	striped	/dev/md49	5	5	21488.25	26860.25	5372.00MB
ordr_0_22	64461MB	striped	/dev/md50	5	6	26860.25	32232.25	5372.00MB
istok_0_0	33401MB	striped	/dev/md51	5	7	32232.25	35016	2783.75MB
istok_0_1	33401MB	striped	/dev/md52	5	8	35016	37799.75	2783.75MB
icust1_0_0	22571MB	striped	/dev/md53	5	9	37799.75	39681	1881.25MB
hist_0_0	7861MB	striped	/dev/md54	5	10	39681	40336.5	655.50MB
hist_0_1	7861MB	striped	/dev/md55	5	11	40336.5	40992	655.50MB
hist_0_2	7861MB	striped	/dev/md56	5	12	40992	41647.5	655.50MB
hist_0_3	7861MB	striped	/dev/md57	5	13	41647.5	42303	655.50MB
hist_0_4	7861MB	striped	/dev/md58	5	14	42303	42958.5	655.50MB
hist_0_5	7861MB	striped	/dev/md59	5	15	42958.5	43614	655.50MB
hist_0_6	7861MB	striped	/dev/md60	6	1	0.25	655.75	655.50MB
hist_0_7	7861MB	striped	/dev/md61	6	2	655.75	1311.25	655.50MB
hist_0_8	7861MB	striped	/dev/md62	6	3	1311.25	1966.75	655.50MB
hist_0_9	7861MB	striped	/dev/md63	6	4	1966.75	2622.25	655.50MB
hist_0_10	7861MB	striped	/dev/md64	6	5	2622.25	3277.75	655.50MB
hist_0_11	7861MB	striped	/dev/md65	6	6	3277.75	3933.25	655.50MB
hist_0_12	7861MB	striped	/dev/md66	6	7	3933.25	4588.75	655.50MB
roll1	8097MB	striped	/dev/md67	6	8	4588.75	5263.75	675.00MB
temp_0_0	7981MB	striped	/dev/md68	6	9	5263.75	5929.25	665.50MB
temp_0_1	7981MB	striped	/dev/md69	6	10	5929.25	6594.75	665.50MB
temp_0_2	7981MB	striped	/dev/md70	6	11	6594.75	7260.25	665.50MB
temp_0_3	7981MB	striped	/dev/md71	6	12	7260.25	7925.75	665.50MB
temp_0_4	7981MB	striped	/dev/md72	6	13	7925.75	8591.25	665.50MB
temp_0_5	7981MB	striped	/dev/md73	6	14	8591.25	9256.75	665.50MB
temp_0_6	7981MB	striped	/dev/md74	6	15	9256.75	9922.25	665.50MB
temp_0_7	7981MB	striped	/dev/md75	7	1	0.25	665.75	665.50MB
temp_0_8	7981MB	striped	/dev/md76	7	2	665.75	1331.25	665.50MB
temp_0_9	7981MB	striped	/dev/md77	7	3	1331.25	1996.75	665.50MB
temp_0_10	7981MB	striped	/dev/md78	7	4	1996.75	2662.25	665.50MB
temp_0_11	7981MB	striped	/dev/md79	7	5	2662.25	3327.75	665.50MB
temp_0_12	7981MB	striped	/dev/md80	7	6	3327.75	3993.25	665.50MB
temp_0_13	7981MB	striped	/dev/md81	7	7	3993.25	4658.75	665.50MB
temp_0_14	7981MB	striped	/dev/md82	7	8	4658.75	5324.25	665.50MB
temp_0_15	7981MB	striped	/dev/md83	7	9	5324.25	5989.75	665.50MB
temp_0_16	7981MB	striped	/dev/md84	7	10	5989.75	6655.25	665.50MB
temp_0_17	7981MB	striped	/dev/md85	7	11	6655.25	7320.75	665.50MB
temp_0_18	7981MB	striped	/dev/md86	7	12	7320.75	7986.25	665.50MB
temp_0_19	7981MB	striped	/dev/md87	7	13	7986.25	8651.75	665.50MB
temp_0_20	7981MB	striped	/dev/md88	7	14	8651.75	9317.25	665.50MB
istok_0_2	33401MB	striped	/dev/md89	8	1	0.25	2784	2783.75MB
dist_0_1	681MB	striped	/dev/md90	8	2	2784	2841.25	57.00MB

icust1_0_1	22571MB	striped	/dev/md91	8	3	2841.25	4722.75	1881.25MB
idist_0_1	161MB	striped	/dev/md92	8	4	4722.75	4736.75	13.75MB
ware_0_1	71MB	striped	/dev/md93	8	5	4736.75	4743.25	6.25MB

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.*

Oracle Database 10g Release 2 Enterprise 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.

Partitioning was not 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®	359,440.24

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	0.16s	0.36s	25.51s
Payment	0.16s	0.34s	23.57s
Order-Status	0.16s	0.36s	20.19s
Delivery (interactive portion)	0.09s	0.35s	10.64s
Delivery (deferred portion)	0.14s	0.35s	11.35s
Stock-Level	0.16s	0.35s	23.79s
Menu	0.000s	0.10s	0.09s

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

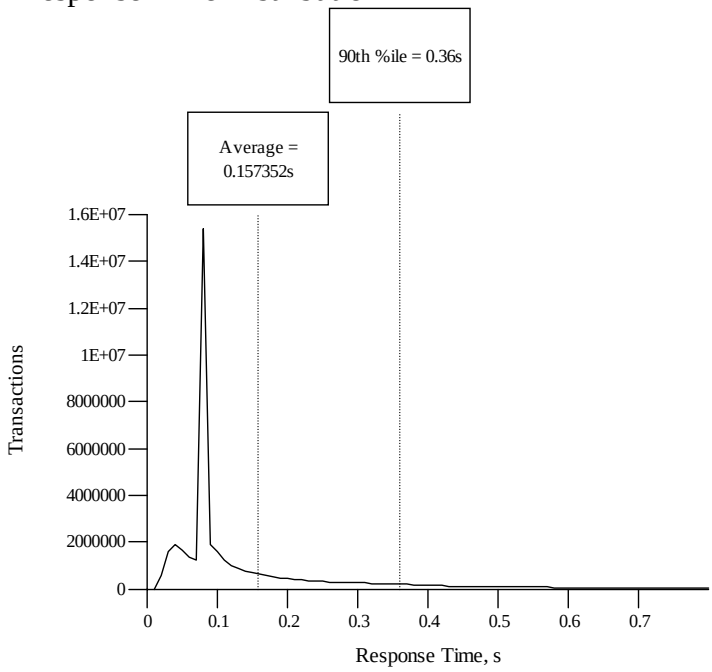
Keying Times	Minimum	Average	Maximum
New Order	18.02s	18.03s	18.33s
Payment	3.02s	3.03s	3.29s
Order Status	2.02s	2.03s	2.27s
Interactive Delivery	2.02s	2.03s	2.27s
Stock Level	2.02s	2.03s	2.25s

Table 6.4: Think Times

Think Times	Minimum	Average	Maximum
New Order	0.02s	12.14s	202.32s
Payment	0.02s	12.07s	227.67s
Order Status	0.02s	10.14s	146.95s
Interactive Delivery	0.02s	5.08s	75.49s
Stock Level	0.02s	5.07s	83.07s

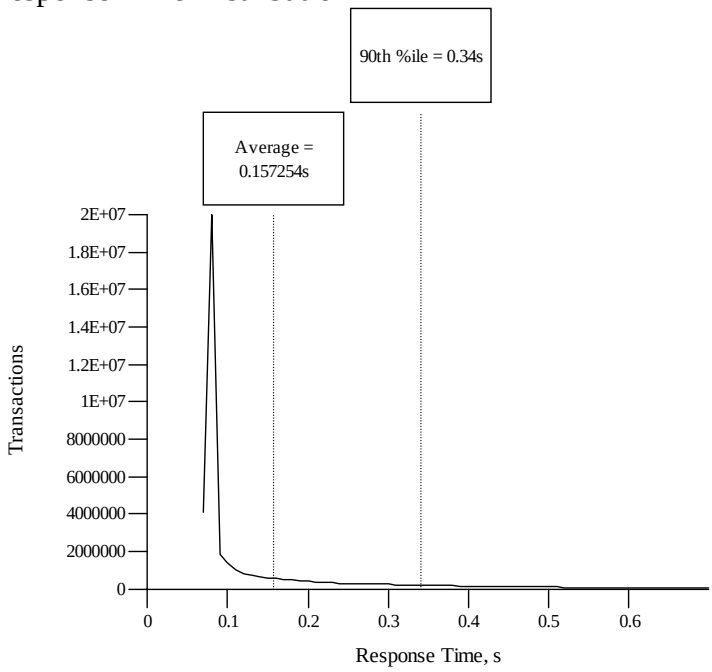
6.4 Response Time Frequency Distribution Curves and Other Graphs

Figure 6.1: New Order Response Time Distribution



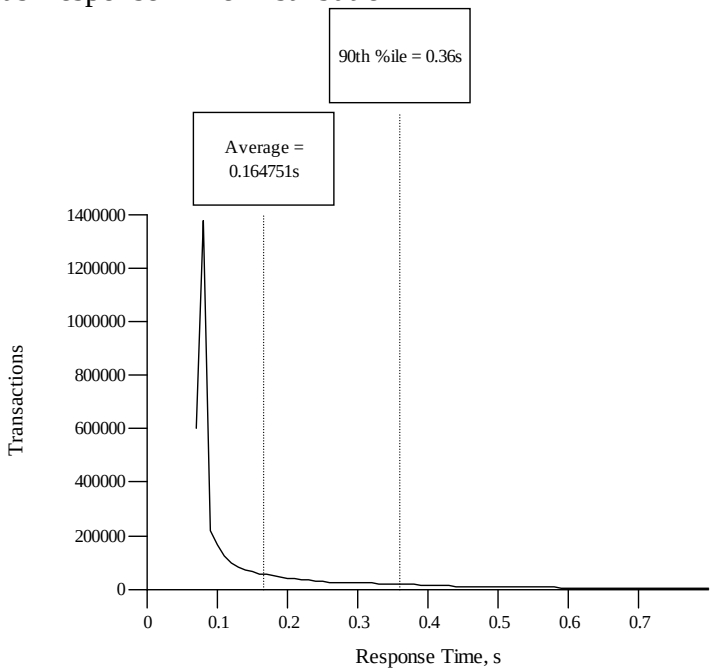
Response time frequency distribution for New Order transaction

Figure 6.2: Payment Response Time Distribution



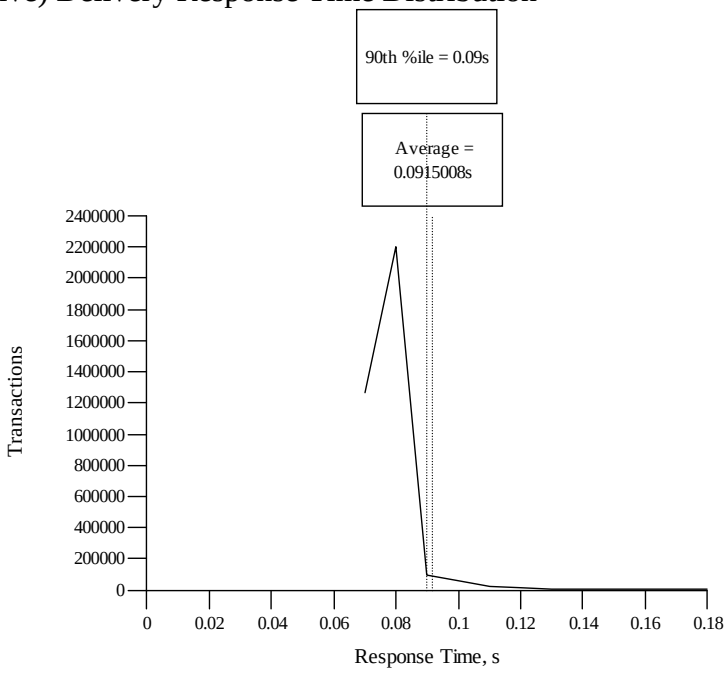
Response time frequency distribution for Payment transaction

Figure 6.3: Order Status Response Time Distribution



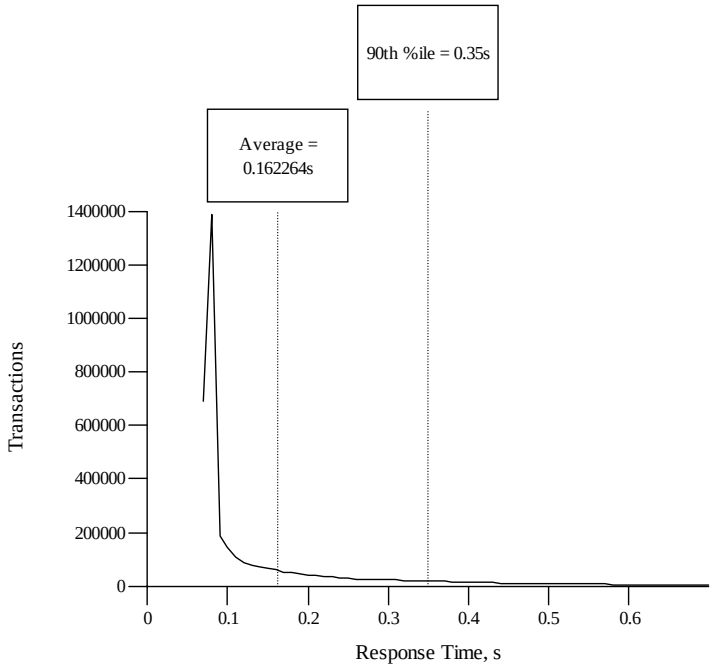
Response time frequency distribution for Order Status transaction

Figure 6.4: (Interactive) Delivery Response Time Distribution



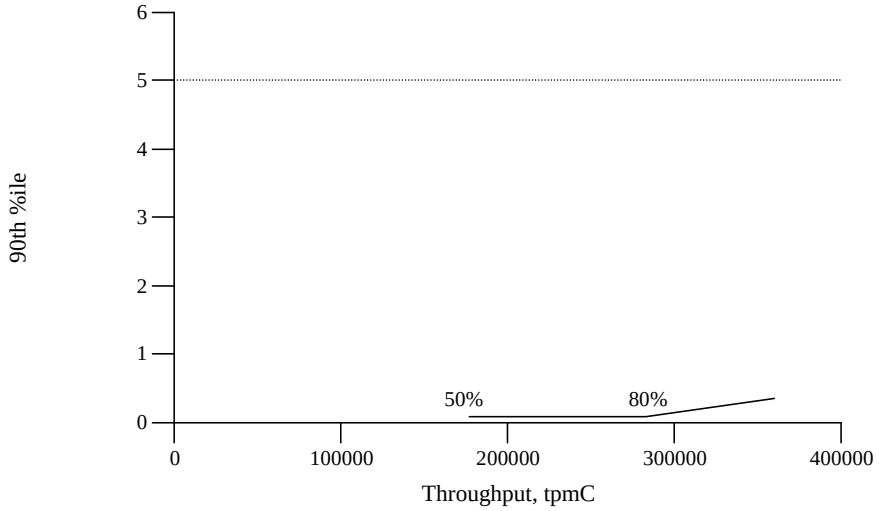
Response time frequency distribution for Delivery transaction

Figure 6.5: Stock Level Response Time Distribution



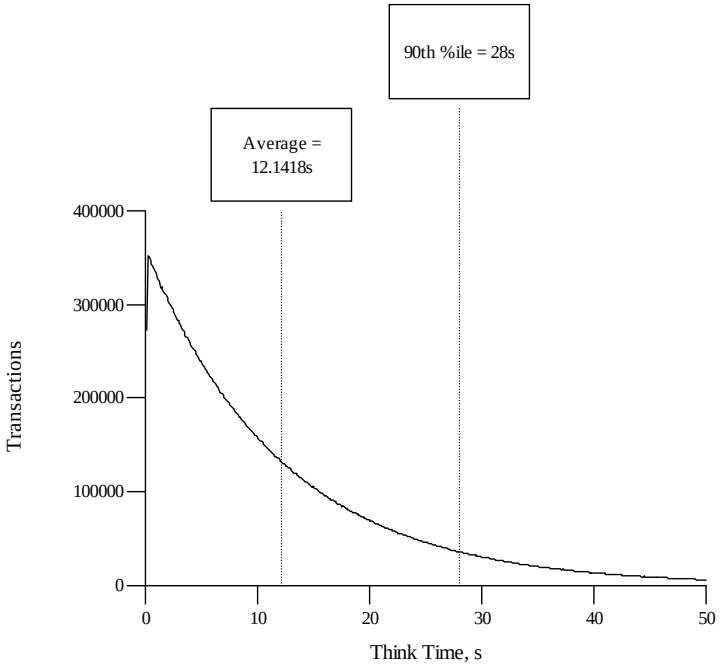
Response time frequency distribution for Stock Level transaction

Figure 6.6: Response Time Versus Throughput



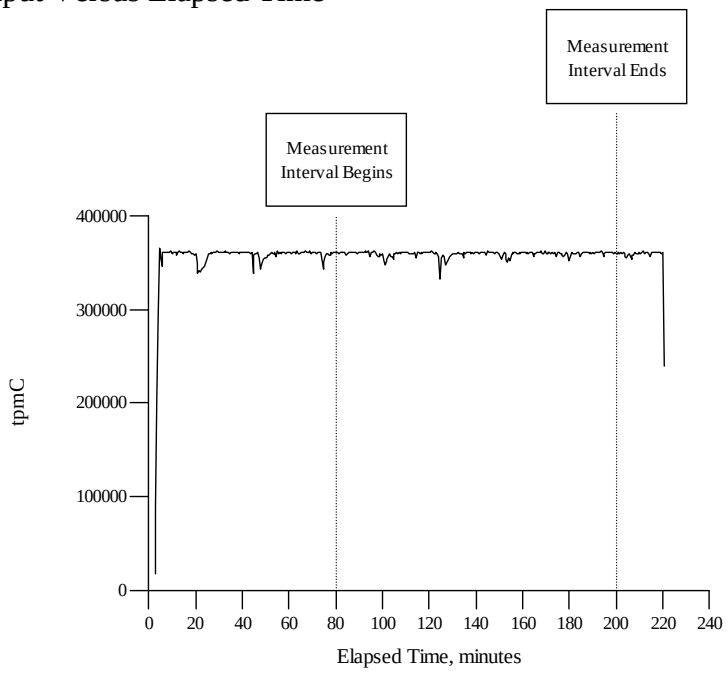
New Order response time versus Throughput

Figure 6.7: New Order Think Time Distribution



Think time frequency distribution for New Order transaction

Figure 6.8: Throughput Versus Elapsed 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 Oracle Database 10g Release 2 Enterprise 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

Oracle Database 10g Release 2 Enterprise 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 29.33 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	44.95%
Payment	43.01%
Order Status	4.01%
Delivery	4.02%
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

Three checkpoints were completed in the measurement run before the measurement interval began. Four checkpoints were initiated within the measurement interval. The first (#3) started 2:24 into the measurement interval. Within the MI, the average checkpoint interval is 29:20, and the average checkpoint duration is 26:24.

The run started at 17:54:39. The Measurement Interval was 19:14:39 to 21:14:39

Event	Start	Stop	duration	interval
run start	17:54:39	-		
ckpt #0	17:24:22	18:16:20	CD: 00:51:58	CI: 00:52:04
ckpt #1	18:17:57	18:44:27	CD: 00:26:30	CI: 00:53:35
ckpt #2	18:47:22	19:14:09	CD: 00:26:47	CI: 00:29:25
MI start	19:14:39	-		
ckpt #3	19:17:03	19:43:31	CD: 00:26:28	CI: 00:29:41
ckpt #4	19:46:22	20:12:34	CD: 00:26:12	CI: 00:29:19
ckpt #5	20:15:26	20:41:49	CD: 00:26:23	CI: 00:29:04
ckpt #6	20:44:42	21:11:15	CD: 00:26:33	CI: 00:29:16
ckpt #7	21:14:09	-		
MI end	-	21:14:39		

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, 6 drivers and 8 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.

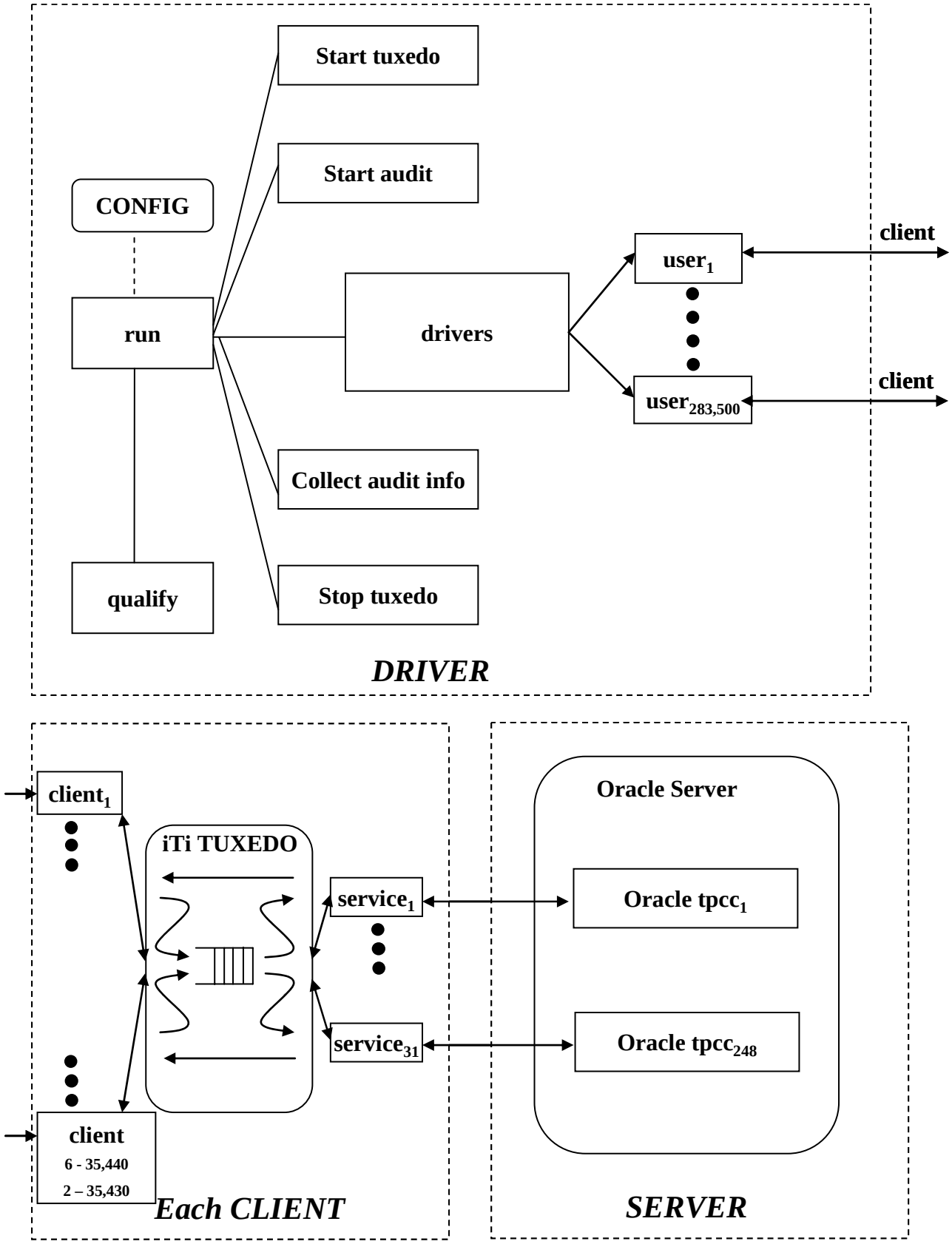


Figure 7.1: Benchmark Software

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 283,500 simulated workstations connected to the clients over 1 1000BT lan through one hp procurve switch. In the priced configuration, the 283,500 worksations would connect to the clients via a combination of hubs and switches which eventually mutipexed down to 1 1000BT lan.

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 1000 Base-T to one 100BT/1000BT-Ethernet switch which in turn are connected via 1000BT-Ethernet to the SUT.

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

The 1000 Base-T local area networks (LAN) used, had 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.

Description	Part Number	Order ability Date	Availability Date
16GB memory quad (4x4GB DIMMS)	AB566A**	December 1, 2006	December 1, 2006
Red Hat Enterprise Linux AS4 Factory Integrated		November 1, 2006	November 1, 2006
Red Hat Enterprise Linux AS4 Server License for 3 years		November 1, 2006	November 1, 2006
Oracle Database 10g Enterprise Edition, Per Processor Unlimited Users, 3 years	Runtime**	December 15, 2006	December 15, 2006

For HP pricing verification, please contact HP Unix Sales Development at 408-447-2320
For Oracle pricing verification, please contact MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com, 916-315-5081

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 Integrity rx6600 - 4p/8c was audited by Lorna Livingtree for Performance Metrics, Inc..

Lorna Livingtree
Performance Metrics, Inc.
P.O. Box 984
Klamath, CA 95548
U.S.A.
Phone: (707) 482-0523
Fax:(707) 482-1352

The attestation letter is shown on the following pages.

PERFORMANCE METRICS INC.
TPC Certified Auditors

October 23, 2006

Curt G. Thiem
HP-UX Performance Manager
Hewlett-Packard Company
Mailstop 4220
19447 Pruneridge Ave.
Cupertino, CA 95014-0683

I have verified the TPC Benchmark™ C client/server for the following configuration:

Platform: **HP Integrity rx6600**
Database Manager: **Oracle Database 10g Release 2 Enterprise Edition**
Operating System: **Red Hat Enterprise Linux AS4.0**
Transaction Manager: **BEA Tuxedo 8.0**

Server: HP Integrity rx6600				
CPUs	**Memory**	**Disks**	**90% Response**	**tpmC**
4 dual-core processors @ 1.6GHz	**Main: 192 GB**	**504 @ 36GB 46 @ 73GB**	**0.36**	**359,440**

Clients: 8 HP server rx2620		
CPUs	**Memory**	**Disks**
2 1.3GHz Intel Itanium™ 2 Processors @ 1.3GHz	**Main: 8GB**	**1 @ 36GB**

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 was properly sized and populated.**
- **The database was properly scaled with 32,000 warehouses, 28,350 of which were used. I verified that d_next_o_id and w_ytd contained initial values for the unused warehouses**
- **The ACID properties were met. The disk-loss tests used the full database and the performance exceeded 10% of the measured tpmC.**
- **Input data was generated according to the specified percentages.**
- **Eight hours of mirrored log space was configured on the measured system.**
- **Eight hours of dynamic table growth space was configured on the measured system.**
- **The 60-day space calculation was verified; the measured system had sufficient storage.**
- **There were 283,500 user contexts present on the system.**
- **Each user started with a different random number seed.**
- **The NURand constants used for database load and at run time were 1 and 86.**
- **The steady state portion of the test was 2 hours.**
- **The system pricing was checked for major components and maintenance.**

Auditor Notes:

Additional disks were attached to the server and used for backup. I verified that these disks were unused during the performance run.

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 Tpc Administrator
P.O. Box 29920
San Francisco, CA 94129-0920

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: 2005/04/11 10:10:29 $

 (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 <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/tcp.h>
#include <netinet/in.h>
#include <netdb.h>
#include <fcntl.h>
#include <errno.h>
#include <pthread.h>
#include <sched.h>
#include <math.h>

#include "key_chars.h"
#include "tpcc.h"
#include "tux_transaction.h"
#include "iobuf.h"

/* 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]

#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;

/*
 * 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

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

/* Maximum number of threads per server */
#ifdef __hpux
#define MAX_USERS_PER_PROCESS   1000
int number_of_servers          = 15; /* default number of servers to spawn */
#else
#define MAX_USERS_PER_PROCESS   240  /* make it less than threads-per-process on
Linux:250 */
int number_of_servers          = 64;
#endif

/* 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*/
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 int  next_field(int current, int key, int max);
static int  neworder(iobuf *in_buf, iobuf *out_buf, iobuf *neworder_form, void **data_ptr,
neworder_trans *t, ID warehouse);
static int  neworder_read(iobuf *in_buf, iobuf *out_buf, neworder_trans *t, ID warehouse);
static void neworder_write(iobuf *out_buf, neworder_trans *t);
static void neworder_setup(iobuf *neworder_form, ID warehouse);
static int  payment(iobuf *in_buf, iobuf *out_buf, iobuf *payment_form, void **data_ptr,
payment_trans *t, ID warehouse);
static void payment_setup(iobuf *payment_form, ID warehouse);
static int  payment_read(iobuf *in_buf, iobuf *out_buf, payment_trans *t, ID warehouse);
static void payment_write(iobuf *out_buf, payment_trans *t);
static int  ordstat(iobuf *in_buf, iobuf *out_buf, iobuf *ordstat_form, void **data_ptr,
ordstat_trans *t, ID warehouse);
static void ordstat_setup(iobuf *ordstat_form, ID warehouse);
static int  ordstat_read(iobuf *in_buf, iobuf *out_buf, ordstat_trans *t, ID warehouse);
static void ordstat_write(iobuf *out_buf, ordstat_trans *t);
static int  delivery(iobuf *in_buf, iobuf *out_buf, iobuf *delivery_form, void *data_ptr,
delivery_trans *t, ID warehouse);
static void delivery_setup(iobuf *delivery_form, ID warehouse);
static int  delivery_read(iobuf *in_buf, iobuf *out_buf, delivery_trans *t, ID warehouse);
static void delivery_write(iobuf *out_buf, delivery_trans *t);
static int  stocklev(iobuf *in_buf, iobuf *out_buf, iobuf *stocklev_form, void **data_ptr,
stocklev_trans *t, ID warehouse, ID district);
static void stocklev_setup(iobuf *stocklev_form, ID warehouse, ID district);
static int  stocklev_read(iobuf *in_buf, iobuf *out_buf, stocklev_trans *t, ID warehouse, ID
district);
static void stocklev_write(iobuf *out_buf, 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,
                                                iobuf *in_buf, iobuf *out_buf);
static int  read_text(int row, int col, char *s, int width, iobuf *in_buf,
                                                iobuf *out_buf);
static int  read_money(int row, int col, double *m, int width, iobuf *in_buf,
                                                iobuf *out_buf);
static int  read_number(int row, int col, int *n, int width, iobuf *in_buf,
                                                iobuf *out_buf);
static void clear_screen(iobuf *out_buf);
static void position(iobuf *out_buf, int row, int col);
static void trigger(iobuf *out_buf);
static void trigger2(iobuf *out_buf);
static void status(iobuf *out_buf, int row, int col, int status);
static void blanks(iobuf *out_buf, int row, int col, int len);
static void empty(iobuf *out_buf, int row, int col, int len);
static void zip(iobuf *out_buf, int row, int col, char *str);
static void phone(iobuf *out_buf, int row, int col,char *str);
static void text(iobuf *out_buf, int row, int col, char str[]);
static void long_text(iobuf *out_buf, int row, int col, char *str, int width);
static void money(iobuf *out_buf, int row, int col, double x, int width);
static void date_only(iobuf *out_buf, int row, int col, char *date_str);
static void date(iobuf *out_buf, int row, int col, char *date_str);
static void real(iobuf *out_buf, int row, int col, double x, int width, int dec);
static void number(iobuf *out_buf, int row, int col, int  n, int width);
static void cleanup(iobuf *out_buf);
static void msgline(iobuf *out_buf, char *str);
static int  menu_read(iobuf *in_buf, iobuf *out_buf);
static void menu_setup(iobuf *out_buf);
static int  login(iobuf *in_buf, iobuf *out_buf, ID *warehouse, ID *district);

void *
client_main(void *arg)
{
  /*
   * variables set up during initialization
   */
  iobuf *in_buf;
  iobuf *out_buf;
  void  *data_ptr; /* where data gets copied to Tuxedo */
  ID    warehouse;
  ID    district;
  int   key;
  int   user;

  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);

   /* a generic transaction variable. */
   generic_trans generic_transaction;

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

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

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

   /* Initialize input/output */
   init_iobuf(output_stuff, OUTPUT_BUF_SIZE, td->fd, td->fd);
   init_iobuf(input_stuff,  INPUT_BUF_SIZE, td->fd, td->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(in_buf, out_buf, &warehouse, &district);
   user      = warehouse*DIST_PER_WARE + district + 1;

   /* set up the forms */
   menu_setup(out_buf);
   neworder_setup(neworder_form, warehouse);
   payment_setup(payment_form, warehouse);
   ordstat_setup(ordstat_form, warehouse);
   delivery_setup(delivery_form, warehouse);
   stocklev_setup(stocklev_form, warehouse, district);

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

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

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

      /* process according to the choice */
      switch(key) {
         case '1':
                                        key = neworder(in_buf, out_buf,
neworder_form,
                                                        &data_ptr,
&trans->neworder, warehouse);
                                                break;
         case '2':
                                        key = payment(in_buf, out_buf,
payment_form,
                                                        &data_ptr,
&trans->payment, warehouse);
                                                break;
         case '3':
                                        key = ordstat(in_buf, out_buf, ordstat_form,
                                                        &data_ptr,
&trans->ordstat, warehouse);
                                                break;
         case '4':
                                        /* Note delivery does not modify data_ptr, thus
the
                                           address is not taken */
                                        key = delivery(in_buf, out_buf, delivery_form,
                                                        data_ptr, &trans-
>delivery, warehouse);
                                                break;
         case '5':
                                        key = stocklev(in_buf, out_buf, stocklev_form,
                                                        &data_ptr,
&trans->stocklev, warehouse, district);
                                        break;
         case EOF:
                                        break;
         case '9':
                                        break;
         default:
                                        msgline(out_buf, "Please enter a valid menu
choice");
                                        break;
      }
   }

   /* done */
   cleanup(out_buf);

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

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


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

Neworder form processing

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


static int
neworder(iobuf *in_buf, iobuf *out_buf, iobuf *neworder_form,
                                void **data_ptr, neworder_trans *t, ID warehouse)
{
   int key;
   display(neworder_form);
   key = neworder_read(in_buf, out_buf, t, warehouse);
   if (key != ENTER) return key;
   neworder_transaction(data_ptr, t);
   neworder_write(out_buf, t);
   return key;
}


static int
neworder_read(iobuf *in_buf, iobuf *out_buf, neworder_trans *t, ID warehouse)
{
   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, in_buf, out_buf);
                 break;

         case 2: key = read_number(5, 12, &t->C_ID, 4, in_buf, out_buf);
                 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, in_buf,
out_buf);
                 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, in_buf, out_buf);
            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, in_buf, out_buf);
            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(out_buf, "Please specify district"); goto retry;}
   if (t->C_ID == EMPTY_NUM)
      {field=2; msgline(out_buf, "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(out_buf, "Please enter
supply warehouse");
                                                      goto retry;
         }
                  if (t->item[i].OL_I_ID == EMPTY_NUM) {
                                                      field=i*3+4;
                                                      msgline(out_buf, "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(out_buf, "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(out_buf, "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(out_buf, 6, 42, t->O_OL_CNT, 2);

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


static void
neworder_write(iobuf *out_buf, 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.
    */

   if (t->status == E_INVALID_ITEM) {
                /* CASE: invalid item, display only these values */
      text(out_buf, 5, 25, t->C_LAST);
      text(out_buf, 5,52,  t->C_CREDIT);
      number(out_buf, 6, 15, t->O_ID, 8);
   } else if (t->status == OK) {
                /* CASE: everything OK, display everything */
      text(out_buf, 5, 25, t->C_LAST);
      text(out_buf, 5,52,  t->C_CREDIT);
      number(out_buf, 6, 15, t->O_ID, 8);
      date(out_buf, 4, 61, t->O_ENTRY_D);
      real(out_buf, 5, 64, t->C_DISCOUNT * 100, 5, 2);
      real(out_buf, 6, 59, t->W_TAX*100, 5, 2);
      real(out_buf, 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(out_buf, 9+i, 2, t->item[i].OL_SUPPLY_W_ID,6);
         number(out_buf, 9+i,10, t->item[i].OL_I_ID, 6);
         text(out_buf, 9+i, 19, t->item[i].I_NAME);
         number(out_buf, 9+i,45, t->item[i].OL_QUANTITY, 2);
         number(out_buf, 9+i, 51, t->item[i].S_QUANTITY, 3);
         position(out_buf, 9+i, 58);   pushc(out_buf,t->item[i].brand_generic);
         money(out_buf, 9+i, 62, t->item[i].I_PRICE, 7);
         money(out_buf, 9+i, 71, amount, 8);
      }

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

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

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

static void
neworder_setup(iobuf *neworder_form, ID warehouse)
{
   int item;
   iobuf *old;

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

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

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

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

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

Payment form processing

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



static int
payment(iobuf *in_buf, iobuf *out_buf, iobuf *payment_form,
                              void **data_ptr, payment_trans *t, ID warehouse)
{
   int key;
   display(payment_form);
   key = payment_read(in_buf, out_buf, t, warehouse);
   if (key != ENTER)  return key;
   payment_transaction(data_ptr, t);
   payment_write(out_buf, t);
```

```
    return key;
}


static void
payment_setup(iobuf *payment_form, ID warehouse)
{
    /* start with an empty form */
    reset(payment_form);

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

    /* set up all the field labels */
    text(payment_form, 3, 38, "Payment");
    text(payment_form, 4, 1, "Date:");
    text(payment_form, 6, 1, "Warehouse:");
    number(payment_form, 6, 12, warehouse, 6);
    text(payment_form, 6, 42, "District:");
    empty(payment_form, 6, 52, 2);
    text(payment_form, 11, 1, "Customer:");
    empty(payment_form, 11, 11, 4);
    text(payment_form, 11, 17, "Cust-Warehouse:");
    empty(payment_form, 11, 33, 6);
    text(payment_form, 11, 40, "Cust-District:");
    empty(payment_form, 11, 54, 2);
    text(payment_form, 12, 1, "Name:");
    empty(payment_form, 12, 29, 16);
    text(payment_form, 12, 50, "Since:");
    text(payment_form, 13, 50, "Credit:");
    text(payment_form, 14, 50, "%Disc:");
    text(payment_form, 15, 50, "Phone:");
    text(payment_form, 17, 1, "Amount Paid:");
    empty(payment_form, 17, 23, 8);
    text(payment_form, 17, 37, "New Cust-Balance:");
    text(payment_form, 18, 1, "Credit Limit:");
    text(payment_form, 20, 1, "Cust-Data:");
}


static int
payment_read(iobuf *in_buf, iobuf *out_buf, payment_trans *t, ID warehouse)
{
    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, in_buf, out_buf);
                    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, in_buf, out_buf);

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

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

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

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

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

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

            case 6: key = read_money(17, 23, &t->H_AMOUNT, 8, in_buf, out_buf);
                    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(out_buf, "Please enter district id"); goto retry;}
    if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
        {field=2; msgline(out_buf, "C_ID or C_LAST must be entered"); goto retry;}
    if (t->C_W_ID == EMPTY_NUM)
        {field=3; msgline(out_buf, "Please enter customer's warehouse"); goto retry;}
    if (t->C_D_ID == EMPTY_NUM)
        {field=4; msgline(out_buf, "please enter customer's district"); goto retry;}
    if (t->H_AMOUNT == EMPTY_FLT)
        {field=6; msgline(out_buf, "Please enter payment amount"); goto retry;}
    if (t->H_AMOUNT <= 0)
                {field=6; msgline(out_buf, "Please enter a positive payment"); goto retry;}


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


static void
payment_write(iobuf *out_buf, payment_trans *t)
{

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

    /* display the screen */
    date(out_buf, 4, 7, t->H_DATE);
    text(out_buf, 7, 1, t->W_STREET_1);
    text(out_buf, 7, 42, t->D_STREET_1);
    text(out_buf, 8, 1, t->W_STREET_2);
    text(out_buf, 8, 42, t->D_STREET_2);
    text(out_buf, 9, 1, t->W_CITY);
    text(out_buf, 9, 22, t->W_STATE);
    zip(out_buf, 9, 25, t->W_ZIP);
    text(out_buf, 9, 42, t->D_CITY);
    text(out_buf, 9, 63, t->D_STATE);
    zip(out_buf, 9, 66, t->D_ZIP);
    number(out_buf, 11, 11, t->C_ID, 4);
    text(out_buf, 12, 9, t->C_FIRST);
    text(out_buf, 12, 26, t->C_MIDDLE);
    text(out_buf, 12, 29, t->C_LAST);
    date_only(out_buf, 12, 58, t->C_SINCE);
    text(out_buf, 13, 9, t->C_STREET_1);
    text(out_buf, 13, 58, t->C_CREDIT);
    text(out_buf, 14, 9, t->C_STREET_2);
    real(out_buf, 14, 58, t->C_DISCOUNT*100, 5, 2);  /* percentage or fraction? */
    text(out_buf, 15, 9, t->C_CITY);
    text(out_buf, 15, 30, t->C_STATE);
    zip(out_buf, 15, 33, t->C_ZIP);
    phone(out_buf,15, 58, t->C_PHONE);
    money(out_buf, 17, 17, t->H_AMOUNT,14);
    money(out_buf, 17, 55, t->C_BALANCE, 15);
    money(out_buf, 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(out_buf, 20, 12, t->C_DATA, 50);
    }

    trigger2(out_buf);
}

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

```
/* ORDSTAT form processing                                    */
/*************************************************************/

static int
ordstat(iobuf *in_buf, iobuf *out_buf, iobuf *ordstat_form,
                                    void **data_ptr, ordstat_trans *t, ID warehouse)
{
   int key;
   display(ordstat_form);
   key = ordstat_read(in_buf, out_buf, t, warehouse);
   if (key != ENTER)  return key;
   ordstat_transaction(data_ptr, t);
   ordstat_write(out_buf, t);
   return key;
}


static void
ordstat_setup(iobuf *ordstat_form, ID warehouse)
{
   /* start with an empty form */
   reset(ordstat_form);

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

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

static int
ordstat_read(iobuf *in_buf, iobuf *out_buf, ordstat_trans *t, ID warehouse)
{
   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, in_buf, out_buf);
               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, in_buf, out_buf);

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

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

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

            /* refresh the C_ID underlines, if possibly needed */
            else if (t->C_ID == EMPTY_NUM)
               empty(out_buf, 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(out_buf, "Please enter district id"); goto retry;}
   if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
      {field=2;  msgline(out_buf, "C_ID or C_LAST must be entered"); goto retry;}


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

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

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

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

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

/*************************************************************/
/* delivery form processing                                  */
/*************************************************************/

static int
delivery(iobuf *in_buf, iobuf *out_buf, iobuf *delivery_form,
                              void *data_ptr, delivery_trans *t, ID warehouse)
{
   int key;
   display(delivery_form);
   key = delivery_read(in_buf, out_buf, t, warehouse);
   if (key != ENTER)  return key;
   delivery_enque(data_ptr, t);
   delivery_write(out_buf, t);
   return key;
}

static void
delivery_setup(iobuf *delivery_form, ID warehouse)
{
   /* start with an empty form */
   reset(delivery_form);

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

   /* set up all the field labels */
   text(delivery_form, 3, 38, "Delivery");
```

```
    text(delivery_form, 4, 1, "Warehouse:");
    number(delivery_form, 4, 12, warehouse, 6);
    text(delivery_form, 6, 1, "Carrier Number:");
    empty(delivery_form, 6, 17, 2);
}

static int
delivery_read(iobuf *in_buf, iobuf *out_buf, delivery_trans *t, ID warehouse)
{
    int i;
    int field;
    int key;

    /* Our warehouse number is fixed */
    t->del.W_ID = warehouse;
    t->del.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->del.O_CARRIER_ID, 2, in_buf, out_buf);
                    break;
            }
                    }

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

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

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

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



/*****************************************************************/
/* stocklev form processing                                      */
/*****************************************************************/

static int
stocklev(iobuf *in_buf, iobuf *out_buf, iobuf *stocklev_form,
                                    void **data_ptr, stocklev_trans *t, ID warehouse, ID district)
{
    int key;
    display(stocklev_form);
    key = stocklev_read(in_buf, out_buf, t, warehouse, district);
    if (key != ENTER)  return key;
    stocklev_transaction(data_ptr, t);
    stocklev_write(out_buf, t);
    return key;
}


static void
stocklev_setup(iobuf *stocklev_form, ID warehouse, ID district)
{
    /* start with an empty form */
    reset(stocklev_form);

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

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

static int
stocklev_read(iobuf *in_buf, iobuf *out_buf, stocklev_trans *t,
                    ID warehouse, ID district)
{
    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, in_buf, out_buf);
                    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(out_buf, "Please enter a threshold within 10 and 20"); goto retry; }

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

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



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

login form processing

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

static int
login(iobuf *in_buf, iobuf *out_buf, ID *warehouse, ID *district)
{
    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(out_buf, 1,1);  clear_screen(out_buf);
    text(out_buf, 3, 30, "Please login.");
    text(out_buf, 5,5,"Warehouse:");
    number(out_buf, 5, 16, w_id, 6);
    text(out_buf, 5, 24, "District:");
    number(out_buf, 5, 34, d_id, 2);
    text(out_buf, 15, 5, "Audit String:");
    text(out_buf, 15, 19, CLIENT_AUDIT_STRING);
    empty(out_buf, 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, in_buf, out_buf);
                break;

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

            case 3:
```

```
            key = read_text(16, 19, auditstr, 20, in_buf, out_buf);
            break;
        }
                }

    if (key != ENTER) {
        return EOF;
                }

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

    if (d_id == EMPTY_NUM && *district == EMPTY_NUM) {
        msgline(out_buf, "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 */
    return key;
}

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

menu form processing

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

static void
menu_setup(iobuf *out_buf)
{

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

static int
menu_read(iobuf *in_buf, iobuf *out_buf)
{
    position(out_buf, 1, 1);
    trigger(out_buf);
    return getkey(in_buf, out_buf);
}



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(iobuf *out_buf, char *str)
{
    position(out_buf, 24, 1);
    clear_screen(out_buf);
    string(out_buf, str);
}

static void
cleanup(iobuf *out_buf)
{
    /* detach from the delivery queue */
    delivery_done();

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

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

Screen Output Routines

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


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



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

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

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

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


static void
long_text(iobuf *out_buf, 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(out_buf, row, col);
            pos = 0;
            row++;
            }

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

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


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

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


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

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

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

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


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

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

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

ASCII terminal control

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


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

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

static void
position(iobuf *out_buf, 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(iobuf *out_buf)
/*********************************************************************
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, iobuf *in_buf, iobuf *out_buf)
/*********************************************************************
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, in_buf, out_buf);
                if (funny(key)) return key;

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

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

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


static int
read_money(int row, int col, double *m, int width, iobuf *in_buf, iobuf *out_buf)
{
    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, in_buf, out_buf);
                if (funny(key)) return key;

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

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

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

static int
read_text(int row, int col, char *s, int width, iobuf *in_buf, iobuf *out_buf)
{
    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, in_buf, out_buf);
    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(out_buf,row, col, temp);

    return key;
}

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

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

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

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

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

        /* 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 prepare_socket(int fd)
{
   int yes = 1;
   int level;

#ifdef __hpux
                level = SOL_SOCKET;
#else
                level = IPPROTO_TCP;
#endif
   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, level, TCP_NODELAY, &yes, sizeof(yes)) < 0)
                return -1;
   return 0;
}

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;
   int retval;

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

   /* 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, (struct sockaddr *)&address, sizeof(struct sockaddr)) < 0) {
      syserror("Can't bind the socket to address\n");
                }

   if (listen(fd, SOMAXCONN) < 0) {
      syserror("Can't listen\n");
                }

   return fd;
```

```
}

static void
GetArgs(int argc, char **argv)
{
   extern char *optarg;
   extern int optind;
   char ch;
   int total_users=0;
   int nr_client=1;  /* minimum nr_client */
   int req_servers=0;

   while((ch = getopt(argc, argv, "u:c:p:")) != EOF)  {
                switch (ch) {
                   case 'u':
                                total_users = atoi(optarg);
                                break;
        case 'c':
                                nr_client= ((nr_client = atoi(optarg))>0 ? nr_client: 1);
                                break;
        case 'p':
           port_number = atoi(optarg);
                                break;
        default:
           printf("Usage: %s -u total_users -c nr_client  -p port_number]\n", argv[0]);
                                exit(1);
     }
    }
    req_servers=(int)ceil((double)total_users/nr_client/MAX_USERS_PER_PROCESS);
    number_of_servers = (req_servers > number_of_servers) ?
                                                req_servers : number_of_servers;
}


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

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

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

#ifdef __hpux
                policy             = SCHED_NOAGE;
   param.sched_priority = PRI_HPUX_TO_POSIX(180);
#else
                policy             = SCHED_OTHER;
   param.sched_priority = 0;
#endif

   if ( (sched_setscheduler(0, policy, &param)) < 0 ) {
        perror("Server can't run sched_noage");
   }

   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;
}
```

client/tux_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;

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

/* void  */
int
transaction_begin(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(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 = (MONEY) 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(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 = (float) 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(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(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->del.W_ID;
  ora_str.delin.o_carrier_id = str->del.O_CARRIER_ID;
  retries     = 0;

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

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

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

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

void
stocklev_transaction(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/Makefile

```
#************************************************************************
*****
#@(#) Version: A.10.10 $Date: 2003/06/26 16:03:06 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#************************************************************************
*****

#
# Common makefile definitions that are used on all platforms.
#

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

include ./database.mk

SH_OPT     =
OPT        = -Wl,-E
LDOPTS     = -E -ldld

TUX_INCLUDE= -I${TUXDIR}/include

COMMON_FLAGS = $(OPTIMIZE) $(DD_DS_FLAGS) $(CINCLUDES)
SH_CFLAGS    = $(COMMON_FLAGS) ${SH_OPT} ${TUX_INCLUDE}
CFLAGS             = $(COMMON_FLAGS) ${OPT} ${TUX_INCLUDE}


PROGRAMS     = client client_batch client_batch_dumb service_real service_dumb raw

all: ${PROGRAMS}

${S}/dummy/transaction.o: ${S}/dummy/transaction.c
                $(CC) ${CFLAGS} -o ${S}/dummy/transaction.o -c
${S}/dummy/transaction.c

raw: raw.o $(D)/socket.o
                cc ${CFLAGS} raw.o $(D)/socket.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

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

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

client: client.o  tux_transaction.o
                ${TUXDIR}/bin/buildclient -v -o client \
        -f "$(COMMON_FLAGS) -D_REENTRANT $(OPT) $(VMPSZ) \
           client.o tux_transaction.o $(L)/tpc_lib.a" -l "-lnsl -lm -lc"

service_dumb: service.o post_dvry.o ${S}/dummy/transaction.o $(L)/tpc_lib.a
                ${TUXDIR}/bin/buildserver -v -b shm -s NEWO_SVC -s PMT_SVC \
                -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -o service_dumb \
                -f "$(SVC_VMPSZ) service.o post_dvry.o \
                ${S}/dummy/transaction.o $L/tpc_lib.a $L/load_lib.a" -l "-lnsl -lm"

service_real: service.o post_dvry.o $(DB_OBJS) $(L)/tpc_lib.a
                ${TUXDIR}/bin/buildserver -v -b shm -s NEWO_SVC -s PMT_SVC \
                -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -o service_real \
        -f "$(SVC_VMPSZ) service.o post_dvry.o \
                $(DB_OBJS) $L/tpc_lib.a ${DB_LDFLAGS}" -l "-lnsl"

client_batch: $(D)/driver.o $(D)/keystroke_batch.o $(D)/screen.o \
                        $(D)/generate.o $(D)/shm.o $(D)/tpcc.o \
                        post_dvry.o $(DB_OBJS) dummy_que.o $(L)/tpc_lib.a
                $(CC) $(D)/driver.o $(D)/keystroke_batch.o $(D)/screen.o \
                $(D)/generate.o $(D)/shm.o $(D)/tpcc.o post_dvry.o \
                $(DB_OBJS) dummy_que.o $(L)/tpc_lib.a \
                ${DB_LDFLAGS} -o client_batch;

client_batch_dumb: $(D)/driver.o $(D)/keystroke_batch.o $(D)/screen.o \
                        $(D)/generate.o $(D)/shm.o $(D)/tpcc.o \
                        post_dvry.o ${S}/dummy/transaction.o dummy_que.o \
            $(L)/tpc_lib.a $(L)/load_lib.a
                $(CC) $(CFLAGS) $(D)/driver.o $(D)/keystroke_batch.o $(D)/screen.o \
        $(D)/generate.o $(D)/shm.o $(D)/tpcc.o post_dvry.o \
        ${S}/dummy/transaction.o dummy_que.o $(L)/tpc_lib.a $(L)/load_lib.a \
                -lnsl -lm -lc -lpthread \
        -o client_batch_dumb;

#-lnsl -lm -lc -lcl

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

clobber: clean
                rm -f ${PROGRAMS}
                rm -f ${WORK_DIR}/bin/service

install: ${PROGRAMS}
                cp ${PROGRAMS} ${WORK_DIR}/bin
                ln -s  ./service_real ${WORK_DIR}/bin/service
```

A.2 Tpc_lib Source

lib/tpcc.h

```
/*************************************************************************
*****
 @(#) Version: A.10.10 $Date: 2005/04/11 09:59:56 $

 (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>
#include <sys/time.h>
#include <sys/resource.h>
#include <errno.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"

#define BATCH_MODE ( tpcc_mode == 1 || tpcc_mode == 5 )

#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 long long BIN_TYPE;
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 */
/* tpcc_start_time is declared in "delay.c" */
#include <time.h>
extern struct timeval tpcc_start_time;
#define elapsed_time(t) ( ((t)->tv_sec - tpcc_start_time.tv_sec) + \
                  ((t)->tv_usec - tpcc_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
/*****************************************************************/
/* 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

typedef struct
   {
      int tv_sec;  /* use 4-byte int, 8-byte timeval across 32/64b machines */
      int tv_usec;
   } timeval_32b;

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_trans1;

/*
 * Delivery structure needs to be padded to a multiple of 512-bytes
 * for things to work properly with a sigle delivery file.
 */
typedef struct {
   delivery_trans1 del;
   char pad_data[512-sizeof(delivery_trans1)]; /* pad out to 512 bytes */
} delivery_trans;

typedef struct {
   int status;
   ID  W_ID;
   ID  O_CARRIER_ID;
   struct {
      ID O_ID;
                  int status;
   } order[10];
   timeval_32b enque[1];
```

```
    timeval_32b deque[1];
    timeval_32b complete[1];
    ACID_STUFF;
    } delivery_rec; /* use delivery_rec to write/read to/from result files */

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
*******************************************************/
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 clientnum;
    int userid;
    } success_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: 2005/04/11 10:00:24 $

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

#define MSG_BUF_SIZE 3*1024

static int msgfile_fd = -1;

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

        ltimestamp = strftime(writebuf, 64, "%m/%d %T ", localtime(&tepoch));
        strncpy(writebuf+ltimestamp, buf, size);
        write(msgfile_fd, writebuf, ltimestamp + size);
}

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

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

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

void
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);
}

void
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);
}


void
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);
}

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

```
                exit(1);
}

/*
 * Configure the file descriptor for the message subsystem.
 * This is not multithreaded, so it must be done by the master
 * process before threads are spawned.
 */
void
configure_error_log(void)
{
                char  *fname;

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

                if (msgfile_fd == -1)           {
                                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");
                                }
                }
}
```

lib/fmt.c

```
/*****************************************************************************
*****
 @(#) Version: A.10.10 $Date: 2005/04/11 10:00:28 $

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

/* formatting routines.  */

/* Note: Currently use integer routines to format and convert.  Need to
   modify the code for cases when integers don't work.  */
void fmt_flt(char *str, double x, int width, int dec);
void cvt_text(char *s, char *text);
void fmtint(char *field, int value, int size, char fill);

void
fmt_money(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(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);
   }


void
fmt_num(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);
   }



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



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



void
fmt_text(char *s, char *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';
   }



void
cvt_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';
   }



void
fmtint(char *field, int value, int size, char 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 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(char *str)
/*********************************************************************
getint extracts an integer value from the given character field
(ex: turns the string "123" into the integer 123)
*********************************************************************/
   {
   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;
   }
```

```
void
fmt_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';
   }


void
fmt_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

```
#ifndef __TPCC_IOBUF__
#define __TPCC_IOBUF__

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

/*
 * Input/Output Buffer management
 */
typedef struct {
   int  ifd;      /* input file descriptor */
   int  ofd;      /* output file descriptor */
   char *beg;     /* Start of the buffer */
   char *end;     /* for output buffers */
   char *max;     /* Last address of the buffer */
   char *cur;     /* for input buffers   */
   char *data;    /* data for the buffer */
} iobuf;

#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

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

extern void string(iobuf *out_buf, char *str);
extern void display(iobuf *scr);
extern void input(iobuf *scr);
extern int getkey(iobuf *in_buf, iobuf *out_buf);

#endif
```

lib/iobuf.c

```
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include "iobuf.h"
#include "errlog.h"

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

void
display(iobuf *scr)
{
   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;
                }
}

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

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

int
getkey(iobuf *in_buf, iobuf *out_buf)
{
                if (in_buf->cur == in_buf->end) {
                        flush(out_buf);
                        reset(in_buf);
                        input(in_buf);
                }

                return popc(in_buf);
}
```

lib/random.c

```
/*****************************************************************************
*****
 @(#) Version: A.10.10 $Date: 2005/11/14 15:34:50 $

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

/*
 * The are all write-once, read-many variables.
 * In a multithreaded environment, they need to
 * be setup in the main process before creating
 * threads.  This is done by calling InitRandomStrings().
```

```
 */
char lastNames[1000][16];

#define THREAD_LOCAL

#ifdef _REENTRANT
#   ifdef HPUX
#undef THREAD_LOCAL
#define THREAD_LOCAL __thread
#   endif
#endif

/* The seed value needs to be per-thread given each call to
 * the "randy" function modifies the seed.  If it were not per
 * thread, then corruption could occur as multiple threads
 * access the randy function simultaneously.
 */
static THREAD_LOCAL int RandySeedIter;

static double exponential(double mean);

void
GenerateLastNames(void)
{
   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]);
   }
}

ID
RandomWarehouse(ID local, ID scale, int percent)
{
   ID w_id;

   /* For the given percent of the time, pick the local warehouse */
   if (RandomNumber(1, 100) > percent || scale == 1) {
      w_id = local;
   } else {
      /* Otherwise, pick a non-local warehouse */
      w_id = RandomNumber(2, scale);
      if (w_id == local) {
         w_id = 1;
                  }
   }
   return w_id;
}

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

   secs = exponential(mean);

   delay(secs+adjust);
}

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

   x = -log(1.0-RandomValue()) * mean;
   return x;
}

void
SetRandomSeed(int val)
{
#ifdef USE_DRAND48
   srand48(val);
#else
   /*
    * Seed must be between 1 and 2147483646 inclusive
    * In particular, it can't be 0 or 2147483647
    */
   if (val < 1) {
      val = 1;
   } else if (val > 2147483646) {
      val = 2147483646;
   }
   RandySeedIter = val;
   randy();
#endif
}

/* 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(void)
{
 int    hi, lo, test;

 /*
  * Make sure the seed is not zero.  It could be zero if someone calls this
  * function without first initializing the seed.
  */
 if (RandySeedIter == 0) RandySeedIter = 1;

 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  */

/*********************************************************************
*****/
/* double RandomValue(void)                                          */
/*                                                                   */
/* return a random value in the range [0.0 .. 1.0)                   */
/*                                                                   */
/*********************************************************************
*****/
double
RandomValue(void)
{
#ifdef USE_DRAND48
   return drand48();
#else
   return randy();
#endif
}
```

lib/Makefile

```
#*********************************************************************
*****
#@(#) Version: A.10.10 $Date: 2002/12/10 14:23:24 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*********************************************************************
*****

CFLAGS= -DHOTWARE -D_REENTRANT $(BUILDFLAGS)

utils=delay.o errlog.o fmt.o random.o tas.o date.o spinlock.o iobuf.o
load_files=random_load.o

all: tpc_lib.a load_lib.a

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

load_lib.a: ${load_files}
                rm -f load_lib.a
                ar -r load_lib.a ${load_files}
clean:
                rm -f *.o
                rm -f *.a

clobber: clean

.s.o:
```

```
		cc $(BUILDFLAGS) -c $*.s

.c.o:
		cc $(BUILDFLAGS) -c $*.c
```

A.3 Transaction Source

client/service.c

```
/*************************************************************************
*****
 @(#) Version: A.10.10 $Date: 2005/04/11 10:10:29 $

 (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 <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/tcp.h>
#include <netinet/in.h>
#include <netdb.h>
#include <fcntl.h>
#include <errno.h>
#include <pthread.h>
#include <sched.h>
#include <math.h>

#include "key_chars.h"
#include "tpcc.h"
#include "tux_transaction.h"
#include "iobuf.h"

/* 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]

#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;

/*
 * 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

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

/* Maximum number of threads per server */
#ifdef __hpux
#define MAX_USERS_PER_PROCESS   1000
int number_of_servers          = 15; /* default number of servers to spawn */
#else
#define MAX_USERS_PER_PROCESS   240  /* make it less than threads-per-process on
Linux:250 */
int number_of_servers          = 64;
#endif

/* 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*/
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 int  next_field(int current, int key, int max);
static int  neworder(iobuf *in_buf, iobuf *out_buf, iobuf *neworder_form, void **data_ptr,
neworder_trans *t, ID warehouse);
static int  neworder_read(iobuf *in_buf, iobuf *out_buf, neworder_trans *t, ID warehouse);
static void neworder_write(iobuf *out_buf, neworder_trans *t);
static void neworder_setup(iobuf *neworder_form, ID warehouse);
static int  payment(iobuf *in_buf, iobuf *out_buf, iobuf *payment_form, void **data_ptr,
payment_trans *t, ID warehouse);
static void payment_setup(iobuf *payment_form, ID warehouse);
static int  payment_read(iobuf *in_buf, iobuf *out_buf, payment_trans *t, ID warehouse);
static void payment_write(iobuf *out_buf, payment_trans *t);
static int  ordstat(iobuf *in_buf, iobuf *out_buf, iobuf *ordstat_form, void **data_ptr,
ordstat_trans *t, ID warehouse);
static void ordstat_setup(iobuf *ordstat_form, ID warehouse);
static int  ordstat_read(iobuf *in_buf, iobuf *out_buf, ordstat_trans *t, ID warehouse);
static void ordstat_write(iobuf *out_buf, ordstat_trans *t);
static int  delivery(iobuf *in_buf, iobuf *out_buf, iobuf *delivery_form, void *data_ptr,
delivery_trans *t, ID warehouse);
static void delivery_setup(iobuf *delivery_form, ID warehouse);
static int  delivery_read(iobuf *in_buf, iobuf *out_buf, delivery_trans *t, ID warehouse);
static void delivery_write(iobuf *out_buf, delivery_trans *t);
static int  stocklev(iobuf *in_buf, iobuf *out_buf, iobuf *stocklev_form, void **data_ptr,
stocklev_trans *t, ID warehouse, ID district);
static void stocklev_setup(iobuf *stocklev_form, ID warehouse, ID district);
static int  stocklev_read(iobuf *in_buf, iobuf *out_buf, stocklev_trans *t, ID warehouse, ID
district);
static void stocklev_write(iobuf *out_buf, 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,
                                                iobuf *in_buf, iobuf *out_buf);
static int  read_text(int row, int col, char *s, int width, iobuf *in_buf,
                                                iobuf *out_buf);
static int  read_money(int row, int col, double *m, int width, iobuf *in_buf,
                                                iobuf *out_buf);
static int  read_number(int row, int col, int *n, int width, iobuf *in_buf,
                                                iobuf *out_buf);
static void clear_screen(iobuf *out_buf);
static void position(iobuf *out_buf, int row, int col);
static void trigger(iobuf *out_buf);
static void trigger2(iobuf *out_buf);
static void status(iobuf *out_buf, int row, int col, int status);
static void blanks(iobuf *out_buf, int row, int col, int len);
static void empty(iobuf *out_buf, int row, int col, int len);
static void zip(iobuf *out_buf, int row, int col, char *str);
static void phone(iobuf *out_buf, int row, int col,char *str);
static void text(iobuf *out_buf, int row, int col, char str[]);
static void long_text(iobuf *out_buf, int row, int col, char *str, int width);
static void money(iobuf *out_buf, int row, int col, double x, int width);
static void date_only(iobuf *out_buf, int row, int col, char *date_str);
static void date(iobuf *out_buf, int row, int col, char *date_str);
static void real(iobuf *out_buf, int row, int col, double x, int width, int dec);
static void number(iobuf *out_buf, int row, int col, int  n, int width);
static void cleanup(iobuf *out_buf);
static void msgline(iobuf *out_buf, char *str);
static int  menu_read(iobuf *in_buf, iobuf *out_buf);
static void menu_setup(iobuf *out_buf);
static int  login(iobuf *in_buf, iobuf *out_buf, ID *warehouse, ID *district);

void *
client_main(void *arg)
{
  /*
   * variables set up during initialization
   */
   iobuf *in_buf;
   iobuf *out_buf;
   void  *data_ptr; /* where data gets copied to Tuxedo */
   ID    warehouse;
   ID    district;
   int   key;
   int   user;

   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);

   /* a generic transaction variable. */
   generic_trans generic_transaction;
```

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

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

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

    /* Initialize input/output */
    init_iobuf(output_stuff, OUTPUT_BUF_SIZE, td->fd, td->fd);
    init_iobuf(input_stuff,  INPUT_BUF_SIZE, td->fd, td->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(in_buf, out_buf, &warehouse, &district);
    user      = warehouse*DIST_PER_WARE + district + 1;

    /* set up the forms */
    menu_setup(out_buf);
    neworder_setup(neworder_form, warehouse);
    payment_setup(payment_form, warehouse);
    ordstat_setup(ordstat_form, warehouse);
    delivery_setup(delivery_form, warehouse);
    stocklev_setup(stocklev_form, warehouse, district);

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

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

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

        /* process according to the choice */
        switch(key) {
            case '1':
                                                key = neworder(in_buf, out_buf,
neworder_form,
                                                        &data_ptr,
&trans->neworder, warehouse);
                                                break;
            case '2':
                                                key = payment(in_buf, out_buf,
payment_form,
                                                        &data_ptr,
&trans->payment, warehouse);
                                                break;
            case '3':
                                                key = ordstat(in_buf, out_buf, ordstat_form,
                                                        &data_ptr,
&trans->ordstat, warehouse);
                                                break;
            case '4':
                                                /* Note delivery does not modify data_ptr, thus
the
                                                   address is not taken */
                                                key = delivery(in_buf, out_buf, delivery_form,
                                                        data_ptr, &trans-
>delivery, warehouse);
                                                break;
            case '5':
                                                key = stocklev(in_buf, out_buf, stocklev_form,
                                                        &data_ptr,
&trans->stocklev, warehouse, district);
                                                break;
            case EOF:
                                                break;
            case '9':
                                                break;
            default:
                                                msgline(out_buf, "Please enter a valid menu
choice");
                                                break;
            }
        }

    /* done */
    cleanup(out_buf);

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

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



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

Neworder form processing

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


static int
neworder(iobuf *in_buf, iobuf *out_buf, iobuf *neworder_form,
                                        void **data_ptr, neworder_trans *t, ID warehouse)
{
    int key;
    display(neworder_form);
    key = neworder_read(in_buf, out_buf, t, warehouse);
    if (key != ENTER) return key;
    neworder_transaction(data_ptr, t);
    neworder_write(out_buf, t);
    return key;
}


static int
neworder_read(iobuf *in_buf, iobuf *out_buf, neworder_trans *t, ID warehouse)
{
    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, in_buf, out_buf);
                    break;

            case 2: key = read_number(5, 12, &t->C_ID, 4, in_buf, out_buf);
                    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, in_buf,
out_buf);
                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, in_buf, out_buf);
                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, in_buf, out_buf);
                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(out_buf, "Please specify district"); goto retry;}
    if (t->C_ID == EMPTY_NUM)
        {field=2; msgline(out_buf, "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(out_buf, "Please enter
supply warehouse");
                                                goto retry;
            }
                        if (t->item[i].OL_I_ID == EMPTY_NUM) {
                                                field=i*3+4;
                                                msgline(out_buf, "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(out_buf, "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(out_buf, "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(out_buf, 6, 42, t->O_OL_CNT, 2);

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


static void
neworder_write(iobuf *out_buf, 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.
     */

    if (t->status == E_INVALID_ITEM) {
                /* CASE: invalid item, display only these values */
        text(out_buf, 5, 25, t->C_LAST);
        text(out_buf, 5,52,  t->C_CREDIT);
        number(out_buf, 6, 15, t->O_ID, 8);
    } else if (t->status == OK) {
                /* CASE: everything OK, display everything */
        text(out_buf, 5, 25, t->C_LAST);
        text(out_buf, 5,52,  t->C_CREDIT);
        number(out_buf, 6, 15, t->O_ID, 8);
        date(out_buf, 4, 61, t->O_ENTRY_D);
        real(out_buf, 5, 64, t->C_DISCOUNT * 100, 5, 2);
        real(out_buf, 6, 59, t->W_TAX*100, 5, 2);
        real(out_buf, 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(out_buf, 9+i, 2, t->item[i].OL_SUPPLY_W_ID,6);
            number(out_buf, 9+i,10, t->item[i].OL_I_ID, 6);
            text(out_buf, 9+i, 19, t->item[i].I_NAME);
            number(out_buf, 9+i,45, t->item[i].OL_QUANTITY, 2);
            number(out_buf, 9+i, 51, t->item[i].S_QUANTITY, 3);
            position(out_buf, 9+i, 58);   pushc(out_buf,t->item[i].brand_generic);
            money(out_buf, 9+i, 62, t->item[i].I_PRICE, 7);
            money(out_buf, 9+i, 71, amount, 8);
        }

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

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

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

static void
neworder_setup(iobuf *neworder_form, ID warehouse)
{
    int item;
    iobuf *old;

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

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

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

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

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

Payment form processing

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



static int
payment(iobuf *in_buf, iobuf *out_buf, iobuf *payment_form,
                                void **data_ptr, payment_trans *t, ID warehouse)
{
    int key;
    display(payment_form);
    key = payment_read(in_buf, out_buf, t, warehouse);
    if (key != ENTER)  return key;
    payment_transaction(data_ptr, t);
    payment_write(out_buf, t);
    return key;
}


static void
```

```
payment_setup(iobuf *payment_form, ID warehouse)
{
   /* start with an empty form */
   reset(payment_form);

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

   /* set up all the field labels */
   text(payment_form, 3, 38, "Payment");
   text(payment_form, 4, 1, "Date:");
   text(payment_form, 6, 1, "Warehouse:");
   number(payment_form, 6, 12, warehouse, 6);
   text(payment_form, 6, 42, "District:");
   empty(payment_form, 6, 52, 2);
   text(payment_form, 11, 1, "Customer:");
   empty(payment_form, 11, 11, 4);
   text(payment_form, 11, 17, "Cust-Warehouse:");
   empty(payment_form, 11, 33, 6);
   text(payment_form, 11, 40, "Cust-District:");
   empty(payment_form, 11, 54, 2);
   text(payment_form, 12, 1, "Name:");
   empty(payment_form, 12, 29, 16);
   text(payment_form, 12, 50, "Since:");
   text(payment_form, 13, 50, "Credit:");
   text(payment_form, 14, 50, "%Disc:");
   text(payment_form, 15, 50, "Phone:");
   text(payment_form, 17, 1, "Amount Paid:");
   empty(payment_form, 17, 23, 8);
   text(payment_form, 17, 37, "New Cust-Balance:");
   text(payment_form, 18, 1, "Credit Limit:");
   text(payment_form, 20, 1, "Cust-Data:");
}


static int
payment_read(iobuf *in_buf, iobuf *out_buf, payment_trans *t, ID warehouse)
{
   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, in_buf, out_buf);
               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, in_buf, out_buf);

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

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

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

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

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

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

         case 6: key = read_money(17, 23, &t->H_AMOUNT, 8, in_buf, out_buf);
               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(out_buf, "Please enter district id"); goto retry;}
   if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
      {field=2; msgline(out_buf, "C_ID or C_LAST must be entered"); goto retry;}
   if (t->C_W_ID == EMPTY_NUM)
      {field=3; msgline(out_buf, "Please enter customer's warehouse"); goto retry;}
   if (t->C_D_ID == EMPTY_NUM)
      {field=4; msgline(out_buf, "please enter customer's district"); goto retry;}
   if (t->H_AMOUNT == EMPTY_FLT)
      {field=6; msgline(out_buf, "Please enter payment amount"); goto retry;}
   if (t->H_AMOUNT <= 0)
               {field=6; msgline(out_buf, "Please enter a positive payment"); goto retry;}


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


static void
payment_write(iobuf *out_buf, payment_trans *t)
{

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

   /* display the screen */
   date(out_buf, 4, 7, t->H_DATE);
   text(out_buf, 7, 1, t->W_STREET_1);
   text(out_buf, 7, 42, t->D_STREET_1);
   text(out_buf, 8, 1, t->W_STREET_2);
   text(out_buf, 8, 42, t->D_STREET_2);
   text(out_buf, 9, 1, t->W_CITY);
   text(out_buf, 9, 22, t->W_STATE);
   zip(out_buf, 9, 25, t->W_ZIP);
   text(out_buf, 9, 42, t->D_CITY);
   text(out_buf, 9, 63, t->D_STATE);
   zip(out_buf, 9, 66, t->D_ZIP);
   number(out_buf, 11, 11, t->C_ID, 4);
   text(out_buf, 12, 9, t->C_FIRST);
   text(out_buf, 12, 26, t->C_MIDDLE);
   text(out_buf, 12, 29, t->C_LAST);
   date_only(out_buf, 12, 58, t->C_SINCE);
   text(out_buf, 13, 9, t->C_STREET_1);
   text(out_buf, 13, 58, t->C_CREDIT);
   text(out_buf, 14, 9, t->C_STREET_2);
   real(out_buf, 14, 58, t->C_DISCOUNT*100, 5, 2);  /* percentage or fraction? */
   text(out_buf, 15, 9, t->C_CITY);
   text(out_buf, 15, 30, t->C_STATE);
   zip(out_buf, 15, 33, t->C_ZIP);
   phone(out_buf,15, 58, t->C_PHONE);
   money(out_buf, 17, 17, t->H_AMOUNT,14);
   money(out_buf, 17, 55, t->C_BALANCE, 15);
   money(out_buf, 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(out_buf, 20, 12, t->C_DATA, 50);
   }

   trigger2(out_buf);
}

/*****************************************************************/
/* ORDSTAT form processing                                       */
/*****************************************************************/

static int
ordstat(iobuf *in_buf, iobuf *out_buf, iobuf *ordstat_form,
```

```
                                void **data_ptr, ordstat_trans *t, ID warehouse)
{
   int key;
   display(ordstat_form);
   key = ordstat_read(in_buf, out_buf, t, warehouse);
   if (key != ENTER)  return key;
   ordstat_transaction(data_ptr, t);
   ordstat_write(out_buf, t);
   return key;
}


static void
ordstat_setup(iobuf *ordstat_form, ID warehouse)
{
   /* start with an empty form */
   reset(ordstat_form);

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

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

static int
ordstat_read(iobuf *in_buf, iobuf *out_buf, ordstat_trans *t, ID warehouse)
{
   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, in_buf, out_buf);
                 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, in_buf, out_buf);

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

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

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

            /* refresh the C_ID underlines, if possibly needed */
            else if (t->C_ID == EMPTY_NUM)
               empty(out_buf, 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(out_buf, "Please enter district id"); goto retry;}
   if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
      {field=2;  msgline(out_buf, "C_ID or C_LAST must be entered"); goto retry;}


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

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

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

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

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

/*****************************************************************/
/* delivery form processing                                      */
/*****************************************************************/

static int
delivery(iobuf *in_buf, iobuf *out_buf, iobuf *delivery_form,
                                void *data_ptr, delivery_trans *t, ID warehouse)
{
   int key;
   display(delivery_form);
   key = delivery_read(in_buf, out_buf, t, warehouse);
   if (key != ENTER)  return key;
   delivery_enque(data_ptr, t);
   delivery_write(out_buf, t);
   return key;
}

static void
delivery_setup(iobuf *delivery_form, ID warehouse)
{
   /* start with an empty form */
   reset(delivery_form);

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

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

```
static int
delivery_read(iobuf *in_buf, iobuf *out_buf, delivery_trans *t, ID warehouse)
{
   int i;
   int field;
   int key;

   /* Our warehouse number is fixed */
   t->del.W_ID = warehouse;
   t->del.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->del.O_CARRIER_ID, 2, in_buf, out_buf);
               break;
      }
            }

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

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

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

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



/*********************************************************************/
/* stocklev form processing                                          */
/*********************************************************************/

static int
stocklev(iobuf *in_buf, iobuf *out_buf, iobuf *stocklev_form,
                        void **data_ptr, stocklev_trans *t, ID warehouse, ID district)
{
   int key;
   display(stocklev_form);
   key = stocklev_read(in_buf, out_buf, t, warehouse, district);
   if (key != ENTER)  return key;
   stocklev_transaction(data_ptr, t);
   stocklev_write(out_buf, t);
   return key;
}


static void
stocklev_setup(iobuf *stocklev_form, ID warehouse, ID district)
{
   /* start with an empty form */
   reset(stocklev_form);

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

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

static int
stocklev_read(iobuf *in_buf, iobuf *out_buf, stocklev_trans *t,
            ID warehouse, ID district)
{
   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, in_buf, out_buf);
               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(out_buf, "Please enter a threshold within 10 and 20"); goto retry; }

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

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



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

login form processing

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

static int
login(iobuf *in_buf, iobuf *out_buf, ID *warehouse, ID *district)
{
   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(out_buf, 1,1);  clear_screen(out_buf);
   text(out_buf, 3, 30, "Please login.");
   text(out_buf, 5,5,"Warehouse:");
   number(out_buf, 5, 16, w_id, 6);
   text(out_buf, 5, 24, "District:");
   number(out_buf, 5, 34, d_id, 2);
   text(out_buf, 15, 5, "Audit String:");
   text(out_buf, 15, 19, CLIENT_AUDIT_STRING);
   empty(out_buf, 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, in_buf, out_buf);
            break;

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

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

```
    if (key != ENTER) {
        return EOF;
                }

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

    if (d_id == EMPTY_NUM && *district == EMPTY_NUM) {
        msgline(out_buf, "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 */
    return key;
}

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

menu form processing

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

static void
menu_setup(iobuf *out_buf)
{

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

static int
menu_read(iobuf *in_buf, iobuf *out_buf)
{
    position(out_buf, 1, 1);
    trigger(out_buf);
    return getkey(in_buf, out_buf);
}



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(iobuf *out_buf, char *str)
{
    position(out_buf, 24, 1);
    clear_screen(out_buf);
    string(out_buf, str);
}

static void
cleanup(iobuf *out_buf)
{
    /* detach from the delivery queue */
    delivery_done();

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


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

Screen Output Routines

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


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



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

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

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

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


static void
long_text(iobuf *out_buf, 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(out_buf, row, col);
            pos = 0;
            row++;
            }

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

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


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

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


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

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

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

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


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

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

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

ASCII terminal control

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


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

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

static void
position(iobuf *out_buf, 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(iobuf *out_buf)
/*************************************************************************
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, iobuf *in_buf, iobuf *out_buf)
/*************************************************************************
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, in_buf, out_buf);
                  if (funny(key)) return key;

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

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

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


static int
read_money(int row, int col, double *m, int width, iobuf *in_buf, iobuf *out_buf)
{
   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, in_buf, out_buf);
                  if (funny(key)) return key;

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

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

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

static int
read_text(int row, int col, char *s, int width, iobuf *in_buf, iobuf *out_buf)
{
   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, in_buf, out_buf);
   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(out_buf,row, col, temp);

    return key;
}

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

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

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

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

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

        /* 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 prepare_socket(int fd)
{
    int yes = 1;
    int level;

#ifdef __hpux
                    level = SOL_SOCKET;
#else
                    level = IPPROTO_TCP;
#endif
    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, level, TCP_NODELAY, &yes, sizeof(yes)) < 0)
                    return -1;
    return 0;
}

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;
    int retval;

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

    /* 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, (struct sockaddr *)&address, sizeof(struct sockaddr)) < 0) {
        syserror("Can't bind the socket to address\n");
                    }

    if (listen(fd, SOMAXCONN) < 0) {
        syserror("Can't listen\n");
                    }

    return fd;
}

static void
GetArgs(int argc, char **argv)
{
```

```
    extern char *optarg;
    extern int optind;
    char ch;
    int total_users=0;
    int nr_client=1;  /* minimum nr_client */
    int req_servers=0;

    while((ch = getopt(argc, argv, "u:c:p:")) != EOF)  {
                    switch (ch) {
                        case 'u':
                                        total_users = atoi(optarg);
                                        break;
            case 'c':
                                        nr_client= ((nr_client = atoi(optarg))>0 ? nr_client: 1);
                                        break;
            case 'p':
                port_number = atoi(optarg);
                                        break;
            default:
                printf("Usage: %s -u total_users -c nr_client  -p port_number]\n", argv[0]);
                                        exit(1);
        }
      }
      req_servers=(int)ceil((double)total_users/nr_client/MAX_USERS_PER_PROCESS);
      number_of_servers = (req_servers > number_of_servers) ?
                                                    req_servers : number_of_servers;
}


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

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

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

#ifdef __hpux
                    policy              = SCHED_NOAGE;
    param.sched_priority = PRI_HPUX_TO_POSIX(180);
#else
                    policy              = SCHED_OTHER;
    param.sched_priority = 0;
#endif

    if ( (sched_setscheduler(0, policy, &param)) < 0 ) {
          perror("Server can't run sched_noage");
    }

    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;
}
```

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;

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

/* void  */
int
transaction_begin(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(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 = (MONEY) 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(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 = (float) 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(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(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->del.W_ID;
  ora_str.delin.o_carrier_id = str->del.O_CARRIER_ID;
  retries     = 0;

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

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

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

```
 * Stock Level Query

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

void
stocklev_transaction(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,v 1.4 2003/07/01 15:42:13 mliu Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */


/*================================================================
==+
|      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
|                OPEN SYSTEMS PERFORMANCE GROUP                 |
|                   All Rights Reserved                         |

+================================================================
==+
| FILENAME
|   tpccpl.c
| DESCRIPTION
|   TPC-C transactions in PL/SQL.

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

#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 *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(stderr,fmtp,va);
  va_end(va);
}
#endif

/* vmm313 void ocierror(fname, lineno, errhp, status)  */
int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
  text errbuf[512];
  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);
    /* sprintf(realfile,"%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);
}



TPCinit (id, uid, pwd)

int id;
char *uid;
char *pwd;

{

  char filename[40];
  text stmbuf[100];

  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);
  }

  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;
  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;

  temp = str->newout.i_price[i];
  str->newout.i_price[i] = str->newout.i_price[j];
  str->newout.i_price[j] = temp;

  temp = str->newout.ol_amount[i];
  str->newout.ol_amount[i] = str->newout.ol_amount[j];
```

```
  str->newout.ol_amount[j] = temp;

  temp = str->newout.s_quantity[i];
  str->newout.s_quantity[i] = str->newout.s_quantity[j];
  str->newout.s_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);


  tmpch = str->newout.brand_generic[i];
  str->newout.brand_generic[i] = str->newout.brand_generic[j];
  str->newout.brand_generic[j] = tmpch;


}

#endif
```

client/oracle/plnew.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: plnew.c,v 1.4 2003/08/05 14:10:58 root 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


#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[32*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));
#if defined(ISO)
  sqlfile("../blocks/tkvcpnew_iso.sql",stmbuf);
#else
#if defined(ISO7)
  sqlfile("../blocks/tkvcpnew_iso7.sql",stmbuf);
#else
  sqlfile("../blocks/tkvcpnew.sql",stmbuf);
#endif
#endif

  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

  total_amount = 0;
  for (i = 0; i < o_ol_cnt; i++) total_amount += nol_amount[i];
  total_amount *= ((float)(1.0 - c_discount)) * (float)(1.0 + ((float)(d_tax)) + ((float)
(w_tax)));
  total_amount = total_amount/100;

  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,v 1.3 2003/07/01 15:44:03 mliu Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*================================================================
==+
|       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
|                OPEN SYSTEMS PERFORMANCE GROUP                  |
|                    All Rights Reserved                         |

+================================================================
==+
| FILENAME
|    plpay.c
| DESCRIPTION
|    OCI version (using PL/SQL stored procedure) of
|    PAYMENT transaction in TPC-C benchmark.

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

#include "ora_tpcc.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"

/*---------------------------------------------------------------------------
                     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;

#if defined(ISO9)
  int secondread = 0;
  char sdate[30];
  ub4  datelen;
  sysdate(sdate);
  printf("Order Status started at: %s\n", sdate);
#endif

  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 {
     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  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))
    {
     errcode=OCIERROR(errhp, execstatus);
                if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
     {
      DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
      retries++;
      goto retry;
     } else {
      return -1;
     }
    }
    if (octx->rcount+1 < 2*10 )
     octx->cust_idx=(octx->rcount+1)/2 ;
    else          /* */
    {
     cbctx.reexec = TRUE;
     cbctx.count = (octx->rcount+1)/2 ;
     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 ("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 {
          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 {
      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
    {
      return -1;
    }
   }
#ifdef ISO9
    sysdate (sdate);
if (!secondread)
    printf ("---------- FIRST READ RESULT (out) %s ----------\n", sdate);
else
    printf ("---------- SECOND READ RESULT (out) %s ----------\n", sdate);

    printf ("c_id = %d\n", c_id);
    printf ("c_last = %s\n", c_last);
    printf ("c_first = %s\n", c_first);
    printf ("c_middle = %s\n", c_middle);
    printf ("c_balance = %7.2f\n", (float)c_balance/100);
    printf ("o_id = %d\n", o_id);
    datelen = sizeof(o_entry_d);

OCIERROR(errhp,OCIDateToText(errhp,&o_entry_d_base,(text*)FULLDATE,SIZ(FULL
DATE),(text*
)0,0,&datelen,o_entry_d));
    printf ("o_entry_d = %s\n", o_entry_d);
    printf ("o_carrier_id = %d\n", o_carrier_id);
    printf ("o_ol_cnt = %d\n", o_ol_cnt);
    printf ("-------------------------------------------------\n\n", sdate);

if (!secondread) {
    printf ("Sleep before re-read order at: %s\n", sdate);
    sleep (30);
    sysdate (sdate);
    printf ("Wake up and reread at: %s\n", sdate);
    secondread = 1;
    goto retry;
}
#endif /* ISO9 */
   }
  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
   {
    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,v 1.3 2003/07/01 15:40:19 mliu Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*============================================================
==+
|       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
|                  OPEN SYSTEMS PERFORMANCE GROUP                 |
|                      All Rights Reserved                        |

+============================================================
==+
| FILENAME
|   plsto.c
| DESCRIPTION
|   OCI version of STOCK LEVEL transaction in TPC-C benchmark.

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

#include "ora_tpcc.h"

#ifdef PLSQLSTO
#define SQLTXT "BEGIN stocklevel.getstocklevel (:w_id, :d_id, :threshold, \
  :low_stock); END;"
#else
#define SQLTXT "SELECT  /*+ nocache(stok) */ 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"
#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,v 1.3 2003/07/01 15:33:25 mliu Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*=================================================================
==+
|       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
|                  OPEN SYSTEMS PERFORMANCE GROUP                 |
|                       All Rights Reserved                       |

+=================================================================
==+
| FILENAME
|    pldel.c
| DESCRIPTION
|    OCI version of DELIVERY transaction in TPC-C benchmark.

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

#include "ora_tpcc.h"
#ifdef TUX
#include <userlog.h>
#endif

/*
extern int userlog();
*/

#define DMLRETDEL

#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];

   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];

   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];

   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];
  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;

  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;
  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);
#if defined(ISO5) || defined(ISO6) || defined(ISO8)
 #if defined(ISO5)
   sqlfile("../blocks/tkvcpdel_iso5.sql",stmbuf);
 #endif
 #if defined(ISO6)
   sqlfile("../blocks/tkvcpdel_iso6.sql",stmbuf);
 #endif
 #if defined(ISO8)
   sqlfile("../blocks/tkvcpdel_iso8.sql",stmbuf);
 #endif
#else
   sqlfile("../blocks/tkvcpdel.sql",stmbuf);
#endif
   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",
         ADR(pldctx->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));

   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 (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:

    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);
      }
    }

    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

    /* array update of customer table */
    execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,
               NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
               OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);

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

   /* 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)
    {
      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: ora_tpcc.h,v 1.3 2003/07/01 15:31:08 mliu Exp $ Copyr (c) 1993 Oracle
 */
/*=============================================================
==+
|        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
|                OPEN SYSTEMS PERFORMANCE GROUP                   |
|                    All Rights Reserved                          |

+=============================================================
==+
| FILENAME
|    tpcc.h
| DESCRIPTION
|    Include file for TPC-C benchmark programs.

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

#ifndef TPCC_H
#define TPCC_H

#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;
   float 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

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 N
    WHERE no_d_id = inittpcc.dist(d)
     AND no_w_id = :w_id
     AND no_o_id = (select min (no_o_id)
                    from nord
                    where no_d_id = N.no_d_id
                        and no_w_id = N.no_w_id)
    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;
```

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;
```

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;
```

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
```

countuser

```
#!/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`"
```

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;
```

tpcc.c

```
#include "tpcc.h"

/* Error message strings */
const char *e_mesg[]={"Transaction complete.","Error","Invalid item number.",
                      "Not enough orders.", "Database ERROR !!!!"};

/* the name of each transaction */
const char *transaction_name[] =
  {"", "New_Order", "Payment", "Order-Status",
   "Delivery", "Stock-Level", "Deferred-Delivery"};
```

delay.c

```
/*****************************************************************************
*****
 @(#) Version: A.10.10 $Date: 2005/04/11 10:00:23 $

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

void
delay(double sec)
/*****************************************************************************
delay sleeps for the specified number of seconds.  (to closest 1/100'th second)
*****************************************************************************
***/
   {
   struct timespec delay;

   /* 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;
   delay.tv_nsec = (sec - delay.tv_sec) * 1000000000;

   if (nanosleep(&delay,NULL) == -1) {
                if (errno != EINTR) {
                                if (errno == ENOSYS) {
                                    struct timeval delay_select;
                                    /* Use select instead */
                                    delay_select.tv_sec  = sec;
                                    delay_select.tv_usec = (sec -
delay_select.tv_sec)*1000000;
                                if (select(0, NULL, NULL, NULL, &delay_select) < 0) {
                                                perror("select");
                                    }
                                } else {
                                                perror("nanosleep");
                                }
                }
   }
}

struct timeval tpcc_start_time;

void
initclock(void)
   {
   gettimeofday(&tpcc_start_time, NULL);
   }


TIME getclock(void)
/*****************************************************************************
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: 2005/04/11 09:59:50 $

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

/*
 * Linux does not have efficient thread local data like HP-UX, so
 * just use the built in drand48 library which will provide a
 * thread-safe implementation probably by a mutex.
 */
#ifdef __linux
#define USE_DRAND48
#endif

#ifndef USE_DRAND48
double randy(void);
#endif

extern void   GenerateLastNames(void);
extern ID     RandomWarehouse(ID local, ID scale, int percent);
extern void   RandomDelay(double mean, double adjust);
extern void   Randomize(void);
extern void   SetRandomSeed(int val);

/* Return a random value in the range [0.0..1.0) */
extern double RandomValue(void);

extern char lastNames[1000][16];

/*****************************************************************************
*****/
/* 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)

/*************************************************************************
*****/
/* LastName selects a random TPC-C style last name.                      */
/*************************************************************************
*****/
#define LastName(num, name) strcpy(name, lastNames[(num)])

#endif
```

set_affinity.sh

```
#!/usr/bin/ksh
#
# core named by index: 0-7
#
# irq 60: log array
# irq 59: 4 data arrays
# irq 61: 4 data arrays
# irq 62: 4 data arrays
# irq 58: private lan interrupts (eth1)
#
# core   irqs   mask    mask
#  0     60     01      0001    lgw
#  2     59     04      0010    dbw2
#  4     61     10      0100    dbw1
#  6     62     40      1000    dbw0
#  7     58     80      4000    eth1
#
#
echo "00000000,00000001" >/proc/irq/60/smp_affinity
echo "00000000,00000010" >/proc/irq/59/smp_affinity
echo "00000000,00000100" >/proc/irq/61/smp_affinity
echo "00000000,00001000" >/proc/irq/62/smp_affinity
echo "00000000,00004000" >/proc/irq/58/smp_affinity
LGWR_P=0
DBW2_P=4
DBW1_P=8
DBW0_P=12

#
# affinity: cpus named by index (0-7) or mask
#
ps -ef | grep -v grep | egrep 'ora_dbw|ora_lgwr|tnslsnr' | awk '{print $2, $8;}' |
  while read pid pname; do
    pname=${pname##*/}   # delete leading path component
    echo $pid $pname
    case $pname in
                *dbw0*)
                   taskset -pc $DBW0_P $pid ;;
                *dbw1*)
                   taskset -pc $DBW1_P $pid ;;
                *dbw2*)
                   taskset -pc $DBW2_P $pid ;;
                *lgwr*)
                   taskset -pc $LGWR_P $pid ;;
                *lsnr*)
                   taskset -p 0xffffffff $pid ;;
    esac
  done
```

set_priority.sh

```
sleep $1
ps -ef | grep -v grep | egrep 'ora_|tnslsnr' | awk '{print $2, $8;}' |
  while read pid pname; do
    pname=${pname##*/}   # delete leading path component
    case $pname in
       *dbw*)
          chrt -fp 49 $pid ;;
       *lgwr*)
          chrt -fp 50 $pid ;;
       *lsnr*|*ora_*)
          chrt -rp 48 $pid ;;
    esac
  done
echo "After $0:"
ps -eo psr,pid,args,bsdtime,%cpu,cls,pri| sort -nk1,2 | grep -v grep | egrep 'ora_|tnslsnr'
```

every_boot.sh

```
#!/usr/bin/ksh
# 192 GB
echo 0x3000000000 > /proc/sys/kernel/shmmax            # shmmax: max shm seg size in
bytes: 192GB
echo 0x0000c00000 > /proc/sys/kernel/shmall            # shmall: ceiling on max # of shm
pages (16K) in system.
echo 750  > /proc/sys/vm/nr_hugepages                  # 97.65% of physical memory, thank-
you.

echo 1048576  > /proc/sys/fs/aio-max-nr                # system-wide  ceiling on aio requests
echo 5 > /proc/sys/kernel/printk                       # shhh
sysctl -w kernel.wake_balance=0                        # less is more
```

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
```

serverinfo

```
#!/bin/ksh

print "Interrupts";       cat /proc/interrupts
print "Shared memory";    ipcs -a
print "Scheduler";        ps -eo psr,pid,args,bsdtime,%cpu,cls,pri| sort -nk1,2
```

audit_table.sh

```
#!/usr/bin/sh -f
#
#============================================================
==+
#          Copyright (c) 1995  Oracle Corp, Belmont, CA         |
#               OPEN SYSTEMS PERFORMANCE GROUP                  |
#                    All Rights Reserved                        |
#============================================================
==+
# FILENAME
#    upd_date.sh
# DESCRIPTION
#    Insert start time into tpcc_audit_tab.
# USAGE
#    upd_date.sh
#

sqlplus -s tpcc/tpcc <<!
  set echo on;
  delete from tpcc_audit_tab;
  insert into tpcc_audit_tab (starttime)
    select sysdate from dual;
  commit;
  select to_char(starttime, 'MM/DD/YY HH24:MI:SS') RUN_START_DATE
       from tpcc_audit_tab;
  quit;
!
```

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;
!
```

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`"

sqlplus 'sys/change_on_install as sysdba' << !
set linesize 100;
column member format a40
select group#, sequence#, status, bytes/1073741824 GB from v\$log;
select group#, member from v\$logfile;
exit;
!
```

Appendix B Database Design

The source code for the process to define, create and populate the **Oracle Database 10g Release 2 Enterprise Edition** TPC-C database is included in this appendix.

B.1 Scripts

addfile.sh

```
#!/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

```
#!/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) or (d) for dictionary
# 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
    createsql="create tablespace $1 datafile '$2' size $3 reuse extent management local uniform size
$4 segment space management auto $bssql nologging ;"
  else
    if expr x$7 = xd > /dev/null; then
      createsql="create tablespace $1 datafile '$2' size $3 reuse extent management dictionary
nologging $bssql;"
    else
      createsql="create tablespace $1 datafile '$2' size $3 reuse extent management local uniform size
$4 segment space management manual $bssql nologging ;"
    fi
  fi
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

```
#!/bin/sh
$tpcc_sqlplus $tpcc_user_pass @${tpcc_sql_dir}/analyze > $tpcc_log_dir/junk 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

Analyze.sql

```
spool analyze.log;
set echo on;

connect tpcc/tpcc

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;
```

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;
```

Create_spacestats.sql

```
@space_init
@space_get 409000 32000
@space_rpt
spool off
exit sql.sqlcode;
```

Create_storedprocs.sql

```
spool createstoreprocs.log
@tkvcinin.sql
spool off
exit sql.sqlcode;
```

Createdb.sql

```
/* created automatically by  /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatedb.sh Tue
Jul 25 06:03:35 PDT 2006 */
spool createdb.log

set echo on

shutdown abort

startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxinstances 1
  datafile
   '/BUILD/dbs/tpcc_disks/system_1' size 400M reuse
  logfile '/BUILD/dbs/tpcc_disks/log_1_1' size 58593M reuse,
          '/BUILD/dbs/tpcc_disks/log_1_2' size 58593M reuse
  sysaux datafile '/BUILD/dbs/tpcc_disks/tpccaux' size 120M reuse ;



create undo tablespace undo_1 datafile
  '/BUILD/dbs/tpcc_disks/roll1' size 8096M reuse blocksize 8K;

set echo off
exit sql.sqlcode
```

createindex_icust1.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:48 PDT 2006 */
set timing on
    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
  storage ( buffer_pool default )
  parallel 16
  compute statistics
  tablespace icust1_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_icust2.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:48 PDT 2006 */
set timing on
    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
  storage ( buffer_pool default )
  parallel 32
  compute statistics
  tablespace icust2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_idist.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:49 PDT 2006 */
set timing on
    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 )
  parallel 1
  compute statistics
  tablespace idist_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_iitem.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:50 PDT 2006 */
set timing on
    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 )

  compute statistics
  tablespace iitem_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_inord.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:53 PDT 2006 */
set timing on
  exit 0;
```

createindex_iordr1.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:51 PDT 2006 */
set timing on
  exit 0;
```

createindex_iordr2.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:51 PDT 2006 */
set timing on
    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
  storage ( buffer_pool default )
  parallel 32
  compute statistics
  tablespace iordr2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_istok.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:50 PDT 2006 */
set timing on
    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
  storage ( buffer_pool default )
  parallel 32
  compute statistics
  tablespace istok_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

Createindex_iware.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:47 PDT 2006 */
set timing on
    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 )
  parallel 1
  compute statistics
  tablespace iware_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createmisc.sh

```
#!/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

```
#!/bin/sh
cd $tpcc_genscripts_dir
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_genscripts_dir/createspacestats > junk 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

createstats.sh

```
#!/bin/sh

cstat=c_stat
if test $tpcc_np -gt 1 ; then
  cstat=c_stat_rac
fi

$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_0 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 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

define default_tablespace='sp_0'

define temporary_tablespace='temp_0'

@$ORACLE_HOME/rdbms/admin/spcreate
perfstat

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

@$tpcc_sql_dir/${cstat}
@$tpcc_sql_dir/pst_c

!
```

createstoredprocs.sh

```
#!/bin/sh
```

```
cd $tpcc_genscripts_dir
$tpcc_sqlplus $tpcc_user_pass @${tpcc_genscripts_dir}/createstoredprocs > junk 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

createtable_cust.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatetable.sh
Tue Jul 25 06:03:37 PDT 2006 */
set timing on
   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 960000000
  hash is ( (c_id * ( 32000 * 10 ) + c_w_id * 10 + c_d_id) )
  size 360
  pctfree 0  initrans 3
  storage ( buffer_pool recycle ) parallel ( degree 8 )
  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 char(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 /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatetable.sh
Tue Jul 25 06:03:39 PDT 2006 */
set timing on
   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 320000
  hash is ( ((d_w_id * 10) + d_id) )
  size 1448
    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 /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatetable.sh
Tue Jul 25 06:03:40 PDT 2006 */
set timing on
    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 recycle )
  tablespace hist_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_item.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatetable.sh
Tue Jul 25 06:03:42 PDT 2006 */
set timing on
    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 keep )
  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 /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatetable.sh
Tue Jul 25 06:03:45 PDT 2006 */
set timing on
    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 320000
    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 /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreateindex.sh
Tue Jul 25 06:03:51 PDT 2006 */
set timing on
  exit 0;
```

createtable_ordr.sql

```
/* created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatetable.sh
Tue Jul 25 06:03:43 PDT 2006 */
set timing on
    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 320000
    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 /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatetable.sh
Tue Jul 25 06:03:41 PDT 2006 */

alter session set "_column_compression_factor"=175;

set timing on
    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 3200000000
  hash is ( (s_i_id * 32000 + s_w_id) )
  size 275
  pctfree 0  initrans 2 maxtrans 2
  storage ( buffer_pool keep ) parallel ( degree 8 )
  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 /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatetable.sh
Tue Jul 25 06:03:35 PDT 2006 */
set timing on
    set sqlblanklines on
    spool createtable_ware.log
    set echo on
      drop cluster warecluster including tables ;

create cluster warecluster (
  w_id number
  )
  single table
  hashkeys 32000
  hash is ( (w_id - 1) )
  size 1448
    initrans 2
  storage ( buffer_pool default )
  tablespace ware_0;

create table ware (
  w_id number
, 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 /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/buildcreatets.sh Tue Jul
25 06:03:24 PDT 2006

# Tablespace ware, ts size 70M (71680K)
# each file 70M (71680K)
# extents 69504K (69504K)
# 1 files

$tpcc_createts ware 1 1       70M 69504K unix 0       0 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for ware failed.  Exiting.
      exit 0
    fi

# Tablespace cust, ts size 1075970M (1101793280K)
# each file 8090M (8284160K)
# extents 223678K (223678K)
# 133 files

$tpcc_createts cust 133 1       8090M 223678K unix 0       1 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for cust failed.  Exiting.
      exit 0
    fi

# Tablespace dist, ts size 680M (696320K)
# each file 680M (696320K)
# extents 685824K (685824K)
# 1 files

$tpcc_createts dist 1 1       680M 685824K unix 0       134 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for dist failed.  Exiting.
      exit 0
    fi

# Tablespace hist, ts size 102180M (104632320K)
# each file 7860M (8048640K)
# extents 103060K (103060K)
# 13 files
```

```
$tpcc_createts hist 13 1       7860M 103060K unix 0       135 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for hist failed.  Exiting.
     exit 0
   fi

# Tablespace stok, ts size 1342680M (1374904320K)
# each file 8040M (8232960K)
# extents 283824K (283824K)
# 167 files

$tpcc_createts stok 167 1       8040M 283824K unix 0       148 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for stok failed.  Exiting.
     exit 0
   fi

# Tablespace item, ts size 20M (20480K)
# each file 20M (20480K)
# extents 16892K (16892K)
# 1 files

$tpcc_createts item 1 1       20M 16892K unix 0       315 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for item failed.  Exiting.
     exit 0
   fi

# Tablespace ordr, ts size 1482580M (1518161920K)
# each file 64460M (66007040K)
# extents 103280K (103280K)
# 23 files

$tpcc_createts ordr 23 1       64460M 103280K unix 0       316 8 16K t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for ordr failed.  Exiting.
     exit 0
   fi

# Tablespace nord, ts size 11180M (11448320K)
# each file 5590M (5724160K)
# extents 571690K (571690K)
# 2 files

$tpcc_createts nord 2 1       5590M 571690K unix 0       339 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for nord failed.  Exiting.
     exit 0
   fi

# Tablespace iware, ts size 50M (51200K)
# each file 50M (51200K)
# extents 41024K (41024K)
# 1 files

$tpcc_createts iware 1 1       50M 41024K unix 0       341 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for iware failed.  Exiting.
     exit 0
   fi

# Tablespace icust1, ts size 22570M (23111680K)
# each file 22570M (23111680K)
# extents 361024K (361024K)
# 1 files

$tpcc_createts icust1 1 1       22570M 361024K unix 0       342 8 16K t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for icust1 failed.  Exiting.
     exit 0
   fi

# Tablespace icust2, ts size 56770M (58132480K)
# each file 8110M (8304640K)
# extents 129596K (129596K)
# 7 files

$tpcc_createts icust2 7 1       8110M 129596K unix 0       343 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for icust2 failed.  Exiting.
     exit 0
   fi

# Tablespace idist, ts size 160M (163840K)
# each file 160M (163840K)
# extents 161024K (161024K)
# 1 files

$tpcc_createts idist 1 1       160M 161024K unix 0       350 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for idist failed.  Exiting.
     exit 0
   fi

# Tablespace istok, ts size 66800M (68403200K)
# each file 33400M (34201600K)
# extents 534368K (534368K)
# 2 files

$tpcc_createts istok 2 1       33400M 534368K unix 0       351 8 16K t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for istok failed.  Exiting.
     exit 0
   fi

# Tablespace iitem, ts size 20M (20480K)
# each file 20M (20480K)
# extents 11264K (11264K)
# 1 files

$tpcc_createts iitem 1 1       20M 11264K unix 0       353 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for iitem failed.  Exiting.
     exit 0
   fi

# Tablespace iordr2, ts size 57120M (58490880K)
# each file 7140M (7311360K)
# extents 102926K (102926K)
# 8 files

$tpcc_createts iordr2 8 1       7140M 102926K unix 0       354 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for iordr2 failed.  Exiting.
     exit 0
   fi

# Tablespace temp, ts size 167580M (171601920K)
# each file 7980M (8171520K)
# extents 204198K (204198K)
# 21 files

$tpcc_createts temp 21 1       7980M 204198K unix 1       362 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for temp failed.  Exiting.
     exit 0
   fi
```

createuser.sh

```
#!/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

```
#!/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
```

ddview.sh

```
#!/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

REM @$ORACLE_HOME/rdbms/admin/catparr

REM fi

spool off
!

#sh $tpcc_scripts/queue.sh
```

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;
```

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
  echo or running steps: $tpcc_steps
  echo "use the 'only' option to only do that step (otherwise all steps after will also be executed.)"
  echo "   (e.g.  $0 listfiles only)"
  echo "use the 'through' option to do a sequence of steps (inclusively.)"
  echo "   (e.g.  $0 shutdowndb through startupdb-p_build)"
  exit 1
fi

startstep=$1
controlcmd=$2
endstep=$3

# Aliases for special steps
if test $startstep = buildcreate; then
  startstep=`echo $tpcc_create_steps | cut -d' ' -f1`
fi

if test $startstep = create; then
  startstep=`echo $tpcc_steps | cut -d' ' -f1`
fi

if test "x$controlcmd" = x; then
  endstep=
  # Since endstep is null it won't match any other steps, so we keep going.
elif test "x$controlcmd" = xonly; then
  controlcmd=only
  # this is allowed
elif test "x$controlcmd" = xthrough; then
  actualstep=f
  for step in $tpcc_create_steps $tpcc_steps ; do
    if test "x$step" = "x$endstep"; then
      actualstep=t
    fi
  done
  if test $actualstep = f; then
    echo "Invalid step $endstep.  Use $0 steps  to show steps."
```

```
    exit 1
  fi
else
  echo "Invalid syntax.  Use $0 by itself for help."
  exit 1
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: $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 -n "`find $tpcc_bench/scripts -name '*.log`"; then
                    mv -f *.log `find $tpcc_bench/scripts -name '*.log` $tpcc_bench/log/
    else
      if test -n "`find $tpcc_bench/ -name '*.log`"; then
        mv -f *.log $tpcc_bench/log/
      fi
    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 $step has completed and requested stop.  Stopping driver.
        exit 0
      fi
    fi
    if test "x$controlcmd" = xonly; then
      exit 0
    fi
    if test "x$endstep" = "x$step"; then
      echo The driver reached the last desired step.  Stopping driver.
      exit 0
    fi
  fi
done

if expr $dostep = f > /dev/null; then
  echo No such step: $1
fi
```

extent.sql

```
REM          Copyright (c) 1994  Oracle Corp, Belmont, CA         |
REM               OPEN SYSTEMS PERFORMANCE GROUP                  |
REM                    All Rights Reserved                        |
REM================================================================+
REM FILENAME
REM      extent.sql
REM DESCRIPTION
REM      List all extents in all the TPCC tablespaces.
REM
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @extent
REM================================================================*/
    set space 2
    set pagesize 2000
    set echo off
    set termout off
    set verify off
    set feedback off
    spool extent.rpt
    select substr(e.tablespace_name,1,8) tspace,
      substr(segment_name,1,11) segment, substr(segment_type,1,15) type,
      substr(extent_id,1,5) eid, substr(file_id,1,5) fid, blocks,
      blocks * t.block_size / 1048576 size_MB
    from   dba_extents e, 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
    order by e.tablespace_name, segment_name, extent_id, file_id;

    select substr(e.tablespace_name,1,8) tspace,
      substr(segment_name,1,11) segment,
      sum(blocks) tot_blk, sum(blocks) * t.block_size / 1048576 size_MB
    from   dba_extents e, 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 e.tablespace_name, segment_name, t.block_size
    order by e.tablespace_name, segment_name;
    spool off;
```

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;
```

initpay.sql

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

exit;
```

loadcust.sh

```
#created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/evenload.sh Tue Jul 25
06:03:46 PDT 2006
rm -f loadcust*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 32000 -C  -l 1 -m 187 >> loadcust0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 188 -m 374 >> loadcust1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 375 -m 561 >> loadcust2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 562 -m 748 >> loadcust3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 749 -m 935 >> loadcust4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 936 -m 1122 >> loadcust5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 1123 -m 1309 >> loadcust6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 1310 -m 1496 >> loadcust7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 1497 -m 1684 >> loadcust8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 1685 -m 1872 >> loadcust9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 1873 -m 2060 >> loadcust10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 2061 -m 2248 >> loadcust11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 2249 -m 2436 >> loadcust12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 2437 -m 2624 >> loadcust13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 2625 -m 2812 >> loadcust14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -C  -l 2813 -m 3000 >> loadcust15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loaddist.sh

```
#created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/evenload.sh Tue Jul 25
06:03:46 PDT 2006
rm -f loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 32000 -h  -b 1 -e 2000 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 2001 -e 4000 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 4001 -e 6000 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 6001 -e 8000 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 8001 -e 10000 >> loadhist4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 10001 -e 12000 >> loadhist5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 12001 -e 14000 >> loadhist6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 14001 -e 16000 >> loadhist7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 16001 -e 18000 >> loadhist8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 18001 -e 20000 >> loadhist9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 20001 -e 22000 >> loadhist10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 22001 -e 24000 >> loadhist11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 24001 -e 26000 >> loadhist12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 26001 -e 28000 >> loadhist13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 28001 -e 30000 >> loadhist14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 30001 -e 32000 >> loadhist15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadhist.sh

```
#created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/evenload.sh Tue Jul 25
06:03:46 PDT 2006
rm -f loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 32000 -h  -b 1 -e 2000 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 2001 -e 4000 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 4001 -e 6000 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 6001 -e 8000 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 8001 -e 10000 >> loadhist4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 10001 -e 12000 >> loadhist5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 12001 -e 14000 >> loadhist6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 14001 -e 16000 >> loadhist7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 16001 -e 18000 >> loadhist8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 18001 -e 20000 >> loadhist9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 20001 -e 22000 >> loadhist10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 22001 -e 24000 >> loadhist11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 24001 -e 26000 >> loadhist12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 26001 -e 28000 >> loadhist13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 28001 -e 30000 >> loadhist14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -h  -b 30001 -e 32000 >> loadhist15.log 2>&1 &
allprocs="$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 /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/evenload.sh Tue Jul 25
06:03:46 PDT 2006
rm -f loadnord*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 32000 -n  -b 1 -e 2000 >> loadnord0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 2001 -e 4000 >> loadnord1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 4001 -e 6000 >> loadnord2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 6001 -e 8000 >> loadnord3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 8001 -e 10000 >> loadnord4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 10001 -e 12000 >> loadnord5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 12001 -e 14000 >> loadnord6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 14001 -e 16000 >> loadnord7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 16001 -e 18000 >> loadnord8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 18001 -e 20000 >> loadnord9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 20001 -e 22000 >> loadnord10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 22001 -e 24000 >> loadnord11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 24001 -e 26000 >> loadnord12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 26001 -e 28000 >> loadnord13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 28001 -e 30000 >> loadnord14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -n  -b 30001 -e 32000 >> loadnord15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadordrordl.sh

```
#created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/evenload.sh Tue Jul 25
06:03:46 PDT 2006
rm -f loadordrordl*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy0.dat -b 1 -e 2000 >> loadordrordl0.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy1.dat -b 2001 -e 4000 >> loadordrordl1.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy2.dat -b 4001 -e 6000 >> loadordrordl2.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy3.dat -b 6001 -e 8000 >> loadordrordl3.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy4.dat -b 8001 -e 10000 >> loadordrordl4.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy5.dat -b 10001 -e 12000 >>
loadordrordl5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy6.dat -b 12001 -e 14000 >>
loadordrordl6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy7.dat -b 14001 -e 16000 >>
loadordrordl7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy8.dat -b 16001 -e 18000 >>
loadordrordl8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy9.dat -b 18001 -e 20000 >>
loadordrordl9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy10.dat -b 20001 -e 22000 >>
loadordrordl10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy11.dat -b 22001 -e 24000 >>
loadordrordl11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy12.dat -b 24001 -e 26000 >>
loadordrordl12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy13.dat -b 26001 -e 28000 >>
loadordrordl13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy14.dat -b 28001 -e 30000 >>
loadordrordl14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -o ${tpcc_disks_location}dummy15.dat -b 30001 -e 32000 >>
loadordrordl15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadstok.sh

```
#created automatically by /project/oracle/build32k/tpcc_32kwh_2kbs/scripts/evenload.sh Tue Jul 25
06:03:46 PDT 2006
rm -f loadstok*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 32000 -S  -j 1 -k 3125 >> loadstok0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 3126 -k 6250 >> loadstok1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 6251 -k 9375 >> loadstok2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 9376 -k 12500 >> loadstok3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 12501 -k 15625 >> loadstok4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 15626 -k 18750 >> loadstok5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 18751 -k 21875 >> loadstok6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 21876 -k 25000 >> loadstok7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 25001 -k 28125 >> loadstok8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 28126 -k 31250 >> loadstok9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 31251 -k 34375 >> loadstok10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 34376 -k 37500 >> loadstok11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 37501 -k 40625 >> loadstok12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 40626 -k 43750 >> loadstok13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 43751 -k 46875 >> loadstok14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 46876 -k 50000 >> loadstok15.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 50001 -k 53125 >> loadstok16.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 53126 -k 56250 >> loadstok17.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 56251 -k 59375 >> loadstok18.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 59376 -k 62500 >> loadstok19.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 62501 -k 65625 >> loadstok20.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 65626 -k 68750 >> loadstok21.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 68751 -k 71875 >> loadstok22.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 71876 -k 75000 >> loadstok23.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 75001 -k 78125 >> loadstok24.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 78126 -k 81250 >> loadstok25.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 81251 -k 84375 >> loadstok26.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 84376 -k 87500 >> loadstok27.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 87501 -k 90625 >> loadstok28.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 90626 -k 93750 >> loadstok29.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 93751 -k 96875 >> loadstok30.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 32000 -S  -j 96876 -k 100000 >> loadstok31.log 2>&1 &
allprocs="$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=tpcc


#folder location of the database files (or links to raw partitions)
tpcc_disks_location=/BUILD/dbs/tpcc_disks

#FOR NT
#tpcc_disks_location=\\\\.\\



#FOR RAC

#node id
#tpcc_rac_id=1


# How many creates_node*.sh will be run in this node, started from tpcc_rac_id
# eq. if tpcc_rac_id is 3 and tpcc_rac_creates_count is 2
# creates_node3.sh and creates_node4.sh will be executed

#tpcc_rac_creates_count=1


#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;
```

p_build.ora

```
compatible = 10.1.0.0.0
db_name = tpcc
control_files = (/BUILD/dbs/tpcc_disks/control_001,/BUILD/dbs/tpcc_disks/control_002)
parallel_max_servers = 100
recovery_parallelism = 40
db_files = 483
db_cache_size   = 16384M
db_8k_cache_size  = 6144M
db_16k_cache_size  = 16384M
dml_locks = 500
statistics_level = basic
log_buffer = 1048576
processes = 200
sessions = 200
transactions = 200
shared_pool_size = 3072M
cursor_space_for_time = TRUE
db_block_size = 2048
undo_management = auto
undo_retention = 2
_in_memory_undo=false
_undo_autotune=false
plsql_optimize_level=2

UNDO_TABLESPACE = undo_1
db_4k_cache_size = 20M

# enable async io on raw devices, and directio
disk_asynch_io                  = true
filesystemio_options            = SetAll
```

p_create.ora

```
compatible = 10.1.0.0.0
db_name = tpcc
control_files = (/BUILD/dbs/tpcc_disks/control_001, /BUILD/dbs/tpcc_disks/control_002)
db_block_size = 2048
db_cache_size = 16384M
db_8k_cache_size = 6144M
log_buffer = 1048576
db_16k_cache_size = 16384M
undo_management = manual
statistics_level = basic
shared_pool_size = 3072M
_in_memory_undo=false
plsql_optimize_level=2
db_4k_cache_size = 20M
```

pay.sql

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

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;
```

Pst_c.sql

```
rem
rem
rem ================================================================+
rem          Copyright (c) 1992  Oracle Corp, Belmont, CA          |
rem                OPEN SYSTEMS PERFORMANCE GROUP                   |
rem                     All Rights Reserved                         |
rem ================================================================+
rem  FILENAME
rem      pst_c.sql
rem  DESCRIPTION
rem      Create Table for OS Specific Process Stats
rem ================================================================*/
rem
rem  Tables for Unix-specific process statistics
rem
rem  Usage: sqlplus internal/internal @pst_c
rem

      connect tpcc/tpcc;
      set echo on;
      DROP TABLE proc_resource;
      DROP TABLE os_stat;

rem
rem  Resource usage for a process.
rem

      CREATE TABLE proc_resource
      (
         config       VARCHAR2(10),
         run          NUMBER,
         proc         NUMBER,
```

```
    child        NUMBER,
    user_cpu_ms   NUMBER,
    system_cpu_ms NUMBER,
    maxrss        NUMBER,
    pagein        NUMBER,
    reclaim       NUMBER,
    zerofill      NUMBER,
    pffincr       NUMBER,
    pffdecr       NUMBER,
    swap          NUMBER,
    syscall       NUMBER,
    volcsw        NUMBER,
    involcsw      NUMBER,
    signal        NUMBER,
    lread         NUMBER,
    lwrite        NUMBER,
    bread         NUMBER,
    bwrite        NUMBER,
    phread        NUMBER,
    phwrite       NUMBER
  );

rem
rem  OS statistics.
rem  These results are from the measurement interval only.
rem

  CREATE TABLE os_stat
  (
    config         VARCHAR2(10),
    run            NUMBER,
    hid            NUMBER,
    syscall        NUMBER,
    intr           NUMBER,
    cswitch        NUMBER,
    pagefault      NUMBER,
    usr            NUMBER,
    sys            NUMBER,
    idl            NUMBER,
    wio            NUMBER
  );

  set echo off;
```

Shutdowndb.sql

```
#!/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,1,18), substr(segment_type,1,15),
        sum(blocks), t.block_size,
        round(sum(blocks) * 0.05), 0,
        sum(blocks) + round(sum(blocks) * 0.05)
  from   dba_extents e, 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 <> 'SP_0'
        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   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name = 'SYSTEM' and t.tablespace_name = f.tablespace_name
    group by t.block_size;

  insert into tpcc_data
    select 'SYSAUX', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name = 'SYSAUX' and t.tablespace_name = f.tablespace_name
    group by t.block_size;

  insert into tpcc_data
    select 'ROLL_SEG', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name like '%UNDO_TS%' and f.tablespace_name = t.tablespace_name
    group by f.tablespace_name, t.block_size;

  insert into tpcc_data
    select 'DB_STAT', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name like '%SP_0%' 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 = 'ORDRCLUSTER_QUEUE' OR
         segment = 'IORDL';
```

```
insert into tpcc_space
select substr(ex$.name,1,18), sum(sp$.sz_blocks), sp$.block_size, 0, 0, 0, 0
from
  (select f.tablespace_name , sum(blocks) sz_blocks, t.block_size block_size
   from dba_data_files f, dba_tablespaces t
   where f.tablespace_name <> 'SYSTEM' and f.tablespace_name = t.tablespace_name
   group by f.tablespace_name, t.block_size
  ) sp$,
  (select distinct tablespace_name, segment_name name
   from 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,1,18), sum(blocks), t.block_size, 0, 0, 0, 0
from dba_data_files f, dba_tablespaces t
where (f.tablespace_name = 'SYSTEM' or f.tablespace_name = 'SYSAUX')
     and f.tablespace_name = t.tablespace_name
group by f.tablespace_name, t.block_size;

insert into tpcc_space
select 'ROLL_SEG', sum(blocks), t.block_size, 0, 0, 0, 0
from dba_data_files f, dba_tablespaces t
where f.tablespace_name = 'UNDO_TS' and f.tablespace_name = t.tablespace_name
group by f.tablespace_name, t.block_size;

insert into tpcc_space
select 'DB_STAT', sum(blocks), t.block_size, 0, 0, 0, 0
from dba_data_files f, dba_tablespaces t
where f.tablespace_name = 'SP_0' 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', 'ORDRCLUSTER_QUEUE', 'IORDL');

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(18),
    type         varchar2(15),
    blocks       number,
    block_size   number,
    five_pct     number,
    daily_grow   number,
    total        number
  );
  create table tpcc_space (
    segment      varchar2(18),
    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_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 static, dynamic, oversize, daily_grow, daily_spre, space60
    from tpcc_totspace;
  spool off;
```

startupdb.sql

```
#!/bin/sh

echo "Starting up database using $1..."

init_file=${1}.ora

if test $tpcc_np -gt 1 ; then
  init_file=build_init_${tpcc_rac_id}.ora
fi

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool startdb.log

set echo on

startup pfile=$init_file 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_estsize=$tpcc_scripts/estsize.sh
tpcc_notneg=$tpcc_scripts/notneg.sh
tpcc_isneg=$tpcc_scripts/isneg.sh


# need a better way to check for bc, may
# resort to checking each directory in path
# if this doesn't work
#11/7/02 - alex.ni this is causing too many problems
#because systems have bc in some odd place.  typically
#mangled cygwin installs w/ mksnt/cygwin mixes
#if test -x /usr/bin/bc -o -x /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 /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=sqlplus

# import options generated by gui
. ${tpcc_bench}/options.sh

#8gb oracle filesize limit (in k)
tpcc_fsize_limit_k=8243200
#2gb - 1k oracle extent limit (in k)
tpcc_extent_limit_k=2048000
#file number limit: 1024
tpcc_file_number_limit=1024

# Runlen calculations should be in hours, but
# this was the old calculation, which assumed
# minutes, and also 8 times:
# tpcc_runlen=`$tpcc_bcexpr 8 \* 60 \* $tpcc_runlen`
# we just want to keep the value as it is.

tpcc_system_size=400M
tpcc_kilo_bytes=1024
#tpcc_logfile_size=`$tpcc_bcexpr 20 + \( $tpcc_scale \)`

if test $tpcc_np -gt 1 ; then
  # 4.69k per commit * 2.1 commit per TPMC ~ 9.85K
  # 9.85k * 30 minutes * 12.5 TPMC per Warehouse = 3693
  tpcc_logfile_size=`$tpcc_bcexpr \( $tpcc_scale \* 3693 \) / $tpcc_kilo_bytes`
else
  # 2.4k per commit * 2.1 commit per TPMC ~ 5k
  # 5k * 30 minutes * 12.5 TPMC per Warehouse = 1875
  tpcc_logfile_size=`$tpcc_bcexpr \( $tpcc_scale \* 1875 \) / $tpcc_kilo_bytes`
fi

if test $tpcc_logfile_size -lt 1024; then
  tpcc_logfile_size=1024
fi
tpcc_logfile_size="${tpcc_logfile_size}M"

tpcc_undo_size=`$tpcc_bcexpr 2 \* $tpcc_scale`
if test $tpcc_undo_size -gt 8096; then
  tpcc_undo_size=8096
fi
if test $tpcc_undo_size -lt 512; then
  tpcc_undo_size=512
fi
tpcc_undo_size="${tpcc_undo_size}M"

tpcc_undo_bs=8K

tpcc_statspack_size=`$tpcc_bcexpr 1 \* $tpcc_scale`
if test $tpcc_statspack_size -gt 2048; then
  tpcc_statspack_size=2048
fi
if test $tpcc_statspack_size -lt 300; then
  tpcc_statspack_size=300
fi
tpcc_statspack_size="${tpcc_statspack_size}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=regular
#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
  if test $tpcc_numfiles = 0 ; then
    tpcc_numfiles=256
  fi
  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
  for table in $tpcc_table_list $tpcc_index_list temp; do
    eval "tpcc_${table}_tsfileinc="
  done
fi
  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
  tpcc_use_sysaux=t
else
  tpcc_queue_avail=f
  tpcc_use_sysaux=f
fi

# for NT, ORACLE does not like $variables in sql scripts, so we must
# hardcode these things for it.
if test x$tpcc_os = xnt; then
  tpcc_hardcode=t
else
  tpcc_hardcode=f
fi

# if this is unset we need to make sure it's something anyway
if test x$tpcc_defbs = x; then
  tpcc_defbs=2
fi

# used for loading program
if test x$tpcc_hash_overflow = xt; then
  tpcc_hash_overflow=t
else
  unset tpcc_hash_overflow
fi
if test x$tpcc_overflow = xt; then
  tpcc_hash_overflow=t
else
  unset tpcc_hash_overflow
fi

tpcc_create_steps="buildtpccflags 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 \
buildcreateindex-iware buildcreateindex-icust1 buildcreateindex-icust2 buildcreateindex-idist
buildcreateindex-istok buildcreateindex-iitem buildcreateindex-iordr1 buildcreateindex-iordr2
buildcreateindex-iordl buildcreateindex-inord \
buildstoreprocsql buildspacestats listfiles
"

# remove runscript-loadfixordrordl - shuang, 030626

tpcc_steps="runsqllocal-createdb shutdowndb startupdb-p_build createuser ddview runscript-createts
assigntemp \
  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-loadordrordrl runscript-loadcust runscript-loadstok \
analyze 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 \
createstats createstoredprocs createspacestats createmisc"


tpcc_total_files=524

# 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

if test x$tpcc_hardcode != xt; then
  tpcc_disks_location=${tpcc_disks_location}/
#  tpcc_sql_dir='$tpcc_sql_dir'
#  tpcc_statspack_size='$tpcc_statspack_size'
#  tpcc_genscripts_dir='$tpcc_genscripts_dir'
fi
```

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
/
```

Appendix C Tunable Parameters

The HP-UX operating system tunable parameters employed to generate the kernel for the HP Integrity rx6600 - 4p/8c and the 8 HP 9000 Model rx2620 clients are listed below. Included as well are the Oracle Database 10g Release 2 Enterprise Edition and TUXEDO 8.0 parameters.

C.1 HP-UX Configuration - Clients

Config/Client2/ostune.ver

```
*
* Created on Tue Oct  3 11:52:25 2006
*
version 1
configuration nextboot "imported from /stand/system" [4522b164]
*
* Module entries
*
module gvid_info loaded 0.1.0
module rng loaded 0.1.0
module dev_config best [3F56E2F0]
module dmem best [3F56E2F0]
module diag2 best [3F56E2F0]
module pdh best [3F56E2F0]
module lion_psm best [3F56E2F0]
module ia64_psm best [3F56E2F0]
module wxb_hp best [3F56E2F0]
module sac best [3F56E2F0]
module acpi_node best [3F56E2F0]
module LCentIf best [3F56E2F0]
module ipmi best [3F56E2F0]
module pty1 best [3F56E2F0]
module pty0 best [3F56E2F0]
module azusa_psm best [3F56E2F0]
module ipf loaded 0.1.0
module pfil auto 0.1.0
module mpt best [3F4A8371]
module vols best [3F41B706]
module vol best [3F41B706]
module vxdmp best [3F41B577]
module vxvm best [3F41B706]
module iether best [3F4542A0]
module gelan best [3F454178]
module fddi4 best [3F4122D1]
module td best [3F533FD9]
module cifs best [3F465E27]
module cdfs best 0.1.0
module gvid_core best [3F56E2F0]
module asio0 best [3F56E2F0]
module igelan best [3F454271]
module sctl best [3F56E2F0]
module sdisk best [3F56E2F0]
module side best [3F56E2F0]
module side_multi best [3F56E2F0]
module ehci best [3F56E2F0]
module hub best [3F56E2F0]
module hcd best [3F56E2F0]
module lba best [3F56E2F0]
module sba best [3F56E2F0]
module drmfgl auto 0.1.0
module gvid_him_rad auto 0.1.0
module gvid_him_fgl auto 0.1.0
module inet best [3F559170]
module uipc best [3F56E2F0]
module tun best [3F559170]
module telm best [3F559170]
module tels best [3F559170]
module btlan best [3F559170]
module intl100 best [3F559170]
module dlpi best [3F559170]
module token_arp best [3F559170]
module netdiag1 best [3F56E2F0]
module nms best [3F559170]
module nfs_core best [3F559170]
module nfs_server best [3F559170]
module nfs_client best [3F559170]
module nfsm best [3F559170]
module rpcmod best [3F559170]
module autofsc best [3F559170]
module cachefsc best [3F559170]
module hpstreams best [3F559170]
module clone best [3F559170]
module strlog best [3F559170]
module sad best [3F559170]
module echo best [3F559170]
module sc best [3F559170]
module timod best [3F559170]
module tirdwr best [3F559170]
module pipedev best [3F559170]
module pipemod best [3F559170]
module ffs best [3F559170]
module ldterm best [3F559170]
module ptem best [3F559170]
module pts best [3F559170]
module ptm best [3F559170]
module pckt best [3F559170]
module vxfs best [3F559170]
module vxportal best [3F559170]
module lvm best [3F559170]
module lv best [3F559170]
module drmfglrx auto 0.1.0
*
* Swap entries
*
*
* Dump entries
*
dump lvol
*
* Driver binding entries
*
*
* Tunables entries
*
tunable nhtbl_scale 1
tunable aio_proc_thread_pct 70
tunable maxdsiz_64bit 4294967296
tunable vps_ceiling 4
tunable shmseg 16
tunable shmmni 16
tunable shmmax 0x40000000
tunable semmni 32
tunable semvmx 62000
tunable semume 4
tunable semmsl 2048
tunable semaem 16384
tunable semmns 4096
tunable semmnu 256
tunable nstrpty 200
tunable npty 128
tunable maxuprc 1334
tunable nproc (100+maxuprc)
tunable ninode 1024
tunable nfile 62000
tunable msgmnb 2097152
tunable msgmax 32768
tunable msgssz 512
tunable msgseg (msgmni*2)
tunable msgmni nkthread/16
tunable msgtql 4096
tunable msgmap (msgtql+2)
tunable nkthread 75000
tunable max_thread_proc 1330
tunable maxswapchunks 4096
tunable maxssiz 0x10000000
tunable maxdsiz 0x80000000
tunable maxfiles_lim 2048
tunable maxfiles 2048
tunable fs_async 1
tunable default_disk_ir 1
tunable create_fastlinks 1
tunable STRMSGSZ 65535
tunable vxfs_ifree_timelag 0xf000000
tunable vxfs_bc_bufhwm 0
tunable secure_sid_scripts 0
tunable vx_ninode 20000
tunable dbc_max_pct 5
tunable dbc_min_pct 1
tunable bufpages 8192
tunable swapmem_on 1
```

C.2 HP-UX Configurtion – Server

Config/Server/ostune.ver

```
## /etc/modprobe.conf
alias eth0 e1000
alias eth1 e1000
alias scsi_hostadapter cciss
alias usb-controller ehci-hcd
alias usb-controller1 ohci-hcd
alias scsi_hostadapter1 qla2400

options e1000 InterruptThrottleRate=3000,3000

## /etc/elilo.conf
prompt
timeout=20
default=linux
relocatable
append="rhgb quiet elevator=noop noirqdebug nohalt"

image=vmlinuz-2.6.9-42.EL
        label=linux
        initrd=initrd-2.6.9-42.EL.img
        read-only
        root=/dev/VolGroup00/lvroot

## sysctl -A
sunrpc.tcp_slot_table_entries = 16
sunrpc.udp_slot_table_entries = 16
sunrpc.max_resvport = 1023
sunrpc.min_resvport = 650
sunrpc.nlm_debug = 0
sunrpc.nfsd_debug = 0
sunrpc.nfs_debug = 0
sunrpc.rpc_debug = 0
dev.scsi.logging_level = 0
dev.raid.speed_limit_max = 200000
dev.raid.speed_limit_min = 1000
dev.cdrom.check_media = 0
dev.cdrom.lock = 1
dev.cdrom.debug = 0
dev.cdrom.autoeject = 0
dev.cdrom.autoclose = 1
dev.cdrom.info = CD-ROM information, Id: cdrom.c 3.20 2003/12/17
dev.cdrom.info =
dev.cdrom.info = drive name:                sr0
dev.cdrom.info = drive speed:                       24
dev.cdrom.info = drive # of slots:          1
dev.cdrom.info = Can close tray:                    1
dev.cdrom.info = Can open tray:                     1
dev.cdrom.info = Can lock tray:                     1
dev.cdrom.info = Can change speed:          1
dev.cdrom.info = Can select disk:           0
dev.cdrom.info = Can read multisession:     1
dev.cdrom.info = Can read MCN:                      1
dev.cdrom.info = Reports media changed:     1
dev.cdrom.info = Can play audio:                    1
dev.cdrom.info = Can write CD-R:                    1
dev.cdrom.info = Can write CD-RW:           1
dev.cdrom.info = Can read DVD:                      1
dev.cdrom.info = Can write DVD-R:           1
dev.cdrom.info = Can write DVD-RAM:         0
dev.cdrom.info = Can read MRW:                      1
dev.cdrom.info = Can write MRW:                     1
dev.cdrom.info = Can write RAM:                     1
dev.cdrom.info =
dev.cdrom.info =
net.ipv6.conf.default.max_addresses = 16
net.ipv6.conf.default.max_desync_factor = 600
net.ipv6.conf.default.regen_max_retry = 5
net.ipv6.conf.default.temp_prefered_lft = 86400
net.ipv6.conf.default.temp_valid_lft = 604800
net.ipv6.conf.default.use_tempaddr = 0
net.ipv6.conf.default.force_mld_version = 0
net.ipv6.conf.default.router_solicitation_delay = 1
net.ipv6.conf.default.router_solicitation_interval = 4
net.ipv6.conf.default.router_solicitations = 3
net.ipv6.conf.default.dad_transmits = 1
net.ipv6.conf.default.autoconf = 1
net.ipv6.conf.default.accept_redirects = 1
net.ipv6.conf.default.accept_ra = 1
net.ipv6.conf.default.mtu = 1280
net.ipv6.conf.default.hop_limit = 64
net.ipv6.conf.default.forwarding = 0
net.ipv6.conf.all.max_addresses = 16
net.ipv6.conf.all.max_desync_factor = 600
net.ipv6.conf.all.regen_max_retry = 5
net.ipv6.conf.all.temp_prefered_lft = 86400
net.ipv6.conf.all.temp_valid_lft = 604800
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.all.force_mld_version = 0
net.ipv6.conf.all.router_solicitation_delay = 1
net.ipv6.conf.all.router_solicitation_interval = 4
net.ipv6.conf.all.router_solicitations = 3
net.ipv6.conf.all.dad_transmits = 1
net.ipv6.conf.all.autoconf = 1
net.ipv6.conf.all.accept_redirects = 1
net.ipv6.conf.all.accept_ra = 1
net.ipv6.conf.all.mtu = 1280
net.ipv6.conf.all.hop_limit = 64
net.ipv6.conf.all.forwarding = 0
net.ipv6.conf.eth1.max_addresses = 16
net.ipv6.conf.eth1.max_desync_factor = 600
net.ipv6.conf.eth1.regen_max_retry = 5
net.ipv6.conf.eth1.temp_prefered_lft = 86400
net.ipv6.conf.eth1.temp_valid_lft = 604800
net.ipv6.conf.eth1.use_tempaddr = 0
net.ipv6.conf.eth1.force_mld_version = 0
net.ipv6.conf.eth1.router_solicitation_delay = 1
net.ipv6.conf.eth1.router_solicitation_interval = 4
net.ipv6.conf.eth1.router_solicitations = 3
net.ipv6.conf.eth1.dad_transmits = 1
net.ipv6.conf.eth1.autoconf = 1
net.ipv6.conf.eth1.accept_redirects = 1
net.ipv6.conf.eth1.accept_ra = 1
net.ipv6.conf.eth1.mtu = 1500
net.ipv6.conf.eth1.hop_limit = 64
net.ipv6.conf.eth1.forwarding = 0
net.ipv6.conf.eth0.max_addresses = 16
net.ipv6.conf.eth0.max_desync_factor = 600
net.ipv6.conf.eth0.regen_max_retry = 5
net.ipv6.conf.eth0.temp_prefered_lft = 86400
net.ipv6.conf.eth0.temp_valid_lft = 604800
net.ipv6.conf.eth0.use_tempaddr = 0
net.ipv6.conf.eth0.force_mld_version = 0
net.ipv6.conf.eth0.router_solicitation_delay = 1
net.ipv6.conf.eth0.router_solicitation_interval = 4
net.ipv6.conf.eth0.router_solicitations = 3
net.ipv6.conf.eth0.dad_transmits = 1
net.ipv6.conf.eth0.autoconf = 1
net.ipv6.conf.eth0.accept_redirects = 1
net.ipv6.conf.eth0.accept_ra = 1
net.ipv6.conf.eth0.mtu = 1500
net.ipv6.conf.eth0.hop_limit = 64
net.ipv6.conf.eth0.forwarding = 0
net.ipv6.conf.lo.max_addresses = 16
net.ipv6.conf.lo.max_desync_factor = 600
net.ipv6.conf.lo.regen_max_retry = 5
net.ipv6.conf.lo.temp_prefered_lft = 86400
net.ipv6.conf.lo.temp_valid_lft = 604800
net.ipv6.conf.lo.use_tempaddr = -1
net.ipv6.conf.lo.force_mld_version = 0
net.ipv6.conf.lo.router_solicitation_delay = 1
net.ipv6.conf.lo.router_solicitation_interval = 4
net.ipv6.conf.lo.router_solicitations = 3
net.ipv6.conf.lo.dad_transmits = 1
net.ipv6.conf.lo.autoconf = 1
net.ipv6.conf.lo.accept_redirects = 1
net.ipv6.conf.lo.accept_ra = 1
net.ipv6.conf.lo.mtu = 16436
net.ipv6.conf.lo.hop_limit = 64
net.ipv6.conf.lo.forwarding = 0
net.ipv6.neigh.eth1.locktime = 0
net.ipv6.neigh.eth1.proxy_delay = 819
net.ipv6.neigh.eth1.anycast_delay = 1024
net.ipv6.neigh.eth1.proxy_qlen = 64
net.ipv6.neigh.eth1.unres_qlen = 3
net.ipv6.neigh.eth1.gc_stale_time = 60
net.ipv6.neigh.eth1.delay_first_probe_time = 5
net.ipv6.neigh.eth1.base_reachable_time = 30
net.ipv6.neigh.eth1.retrans_time = 1024
net.ipv6.neigh.eth1.app_solicit = 0
net.ipv6.neigh.eth1.ucast_solicit = 3
net.ipv6.neigh.eth1.mcast_solicit = 3
net.ipv6.neigh.eth0.locktime = 0
net.ipv6.neigh.eth0.proxy_delay = 819
net.ipv6.neigh.eth0.anycast_delay = 1024
net.ipv6.neigh.eth0.proxy_qlen = 64
net.ipv6.neigh.eth0.unres_qlen = 3
net.ipv6.neigh.eth0.gc_stale_time = 60
net.ipv6.neigh.eth0.delay_first_probe_time = 5
net.ipv6.neigh.eth0.base_reachable_time = 30
net.ipv6.neigh.eth0.retrans_time = 1024
net.ipv6.neigh.eth0.app_solicit = 0
net.ipv6.neigh.eth0.ucast_solicit = 3
net.ipv6.neigh.eth0.mcast_solicit = 3
net.ipv6.neigh.lo.locktime = 0
net.ipv6.neigh.lo.proxy_delay = 819
net.ipv6.neigh.lo.anycast_delay = 1024
net.ipv6.neigh.lo.proxy_qlen = 64
net.ipv6.neigh.lo.unres_qlen = 3
net.ipv6.neigh.lo.gc_stale_time = 60
```

net.ipv6.neigh.lo.delay_first_probe_time = 5
net.ipv6.neigh.lo.base_reachable_time = 30
net.ipv6.neigh.lo.retrans_time = 1024
net.ipv6.neigh.lo.app_solicit = 0
net.ipv6.neigh.lo.ucast_solicit = 3
net.ipv6.neigh.lo.mcast_solicit = 3
net.ipv6.neigh.default.gc_thresh3 = 1024
net.ipv6.neigh.default.gc_thresh2 = 512
net.ipv6.neigh.default.gc_thresh1 = 128
net.ipv6.neigh.default.gc_interval = 30
net.ipv6.neigh.default.locktime = 0
net.ipv6.neigh.default.proxy_delay = 819
net.ipv6.neigh.default.anycast_delay = 1024
net.ipv6.neigh.default.proxy_qlen = 64
net.ipv6.neigh.default.unres_qlen = 3
net.ipv6.neigh.default.gc_stale_time = 60
net.ipv6.neigh.default.delay_first_probe_time = 5
net.ipv6.neigh.default.base_reachable_time = 30
net.ipv6.neigh.default.retrans_time = 1024
net.ipv6.neigh.default.app_solicit = 0
net.ipv6.neigh.default.ucast_solicit = 3
net.ipv6.neigh.default.mcast_solicit = 3
net.ipv6.mld_max_msf = 10
net.ipv6.ip6frag_secret_interval = 600
net.ipv6.ip6frag_time = 60
net.ipv6.ip6frag_low_thresh = 196608
net.ipv6.ip6frag_high_thresh = 262144
net.ipv6.bindv6only = 0
net.ipv6.icmp.ratelimit = 1024
net.ipv6.route.min_adv_mss = 1
net.ipv6.route.mtu_expires = 600
net.ipv6.route.gc_elasticity = 0
net.ipv6.route.gc_interval = 30
net.ipv6.route.gc_timeout = 60
net.ipv6.route.gc_min_interval = 0
net.ipv6.route.max_size = 4096
net.ipv6.route.gc_thresh = 1024
net.unix.max_dgram_qlen = 10
net.token-ring.rif_timeout = 614400
net.ipv4.conf.eth1.force_igmp_version = 0
net.ipv4.conf.eth1.disable_policy = 0
net.ipv4.conf.eth1.disable_xfrm = 0
net.ipv4.conf.eth1.arp_ignore = 0
net.ipv4.conf.eth1.arp_announce = 0
net.ipv4.conf.eth1.arp_filter = 0
net.ipv4.conf.eth1.tag = 0
net.ipv4.conf.eth1.log_martians = 0
net.ipv4.conf.eth1.bootp_relay = 0
net.ipv4.conf.eth1.medium_id = 0
net.ipv4.conf.eth1.proxy_arp = 0
net.ipv4.conf.eth1.accept_source_route = 0
net.ipv4.conf.eth1.send_redirects = 1
net.ipv4.conf.eth1.rp_filter = 1
net.ipv4.conf.eth1.shared_media = 1
net.ipv4.conf.eth1.secure_redirects = 1
net.ipv4.conf.eth1.accept_redirects = 1
net.ipv4.conf.eth1.mc_forwarding = 0
net.ipv4.conf.eth1.forwarding = 0
net.ipv4.conf.eth0.force_igmp_version = 0
net.ipv4.conf.eth0.disable_policy = 0
net.ipv4.conf.eth0.disable_xfrm = 0
net.ipv4.conf.eth0.arp_ignore = 0
net.ipv4.conf.eth0.arp_announce = 0
net.ipv4.conf.eth0.arp_filter = 0
net.ipv4.conf.eth0.tag = 0
net.ipv4.conf.eth0.log_martians = 0
net.ipv4.conf.eth0.bootp_relay = 0
net.ipv4.conf.eth0.medium_id = 0
net.ipv4.conf.eth0.proxy_arp = 0
net.ipv4.conf.eth0.accept_source_route = 0
net.ipv4.conf.eth0.send_redirects = 1
net.ipv4.conf.eth0.rp_filter = 1
net.ipv4.conf.eth0.shared_media = 1
net.ipv4.conf.eth0.secure_redirects = 1
net.ipv4.conf.eth0.accept_redirects = 1
net.ipv4.conf.eth0.mc_forwarding = 0
net.ipv4.conf.eth0.forwarding = 0
net.ipv4.conf.lo.force_igmp_version = 0
net.ipv4.conf.lo.disable_policy = 0
net.ipv4.conf.lo.disable_xfrm = 0
net.ipv4.conf.lo.arp_ignore = 0
net.ipv4.conf.lo.arp_announce = 0
net.ipv4.conf.lo.arp_filter = 0
net.ipv4.conf.lo.tag = 0
net.ipv4.conf.lo.log_martians = 0
net.ipv4.conf.lo.bootp_relay = 0
net.ipv4.conf.lo.medium_id = 0
net.ipv4.conf.lo.proxy_arp = 0
net.ipv4.conf.lo.accept_source_route = 0
net.ipv4.conf.lo.send_redirects = 1
net.ipv4.conf.lo.rp_filter = 1
net.ipv4.conf.lo.shared_media = 1
net.ipv4.conf.lo.secure_redirects = 1
net.ipv4.conf.lo.accept_redirects = 1
net.ipv4.conf.lo.mc_forwarding = 0
net.ipv4.conf.lo.forwarding = 0
net.ipv4.conf.default.force_igmp_version = 0
net.ipv4.conf.default.disable_policy = 0
net.ipv4.conf.default.disable_xfrm = 0
net.ipv4.conf.default.arp_ignore = 0
net.ipv4.conf.default.arp_announce = 0
net.ipv4.conf.default.arp_filter = 0
net.ipv4.conf.default.tag = 0
net.ipv4.conf.default.log_martians = 0
net.ipv4.conf.default.bootp_relay = 0
net.ipv4.conf.default.medium_id = 0
net.ipv4.conf.default.proxy_arp = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.shared_media = 1
net.ipv4.conf.default.secure_redirects = 1
net.ipv4.conf.default.accept_redirects = 1
net.ipv4.conf.default.mc_forwarding = 0
net.ipv4.conf.default.forwarding = 0
net.ipv4.conf.all.force_igmp_version = 0
net.ipv4.conf.all.disable_policy = 0
net.ipv4.conf.all.disable_xfrm = 0
net.ipv4.conf.all.arp_ignore = 0
net.ipv4.conf.all.arp_announce = 0
net.ipv4.conf.all.arp_filter = 0
net.ipv4.conf.all.tag = 0
net.ipv4.conf.all.log_martians = 0
net.ipv4.conf.all.bootp_relay = 0
net.ipv4.conf.all.medium_id = 0
net.ipv4.conf.all.proxy_arp = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.send_redirects = 1
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.all.shared_media = 1
net.ipv4.conf.all.secure_redirects = 1
net.ipv4.conf.all.accept_redirects = 1
net.ipv4.conf.all.mc_forwarding = 0
net.ipv4.conf.all.forwarding = 0
net.ipv4.neigh.eth1.locktime = 1024
net.ipv4.neigh.eth1.proxy_delay = 819
net.ipv4.neigh.eth1.anycast_delay = 1024
net.ipv4.neigh.eth1.proxy_qlen = 64
net.ipv4.neigh.eth1.unres_qlen = 3
net.ipv4.neigh.eth1.gc_stale_time = 60
net.ipv4.neigh.eth1.delay_first_probe_time = 5
net.ipv4.neigh.eth1.base_reachable_time = 30
net.ipv4.neigh.eth1.retrans_time = 1024
net.ipv4.neigh.eth1.app_solicit = 0
net.ipv4.neigh.eth1.ucast_solicit = 3
net.ipv4.neigh.eth1.mcast_solicit = 3
net.ipv4.neigh.eth0.locktime = 1024
net.ipv4.neigh.eth0.proxy_delay = 819
net.ipv4.neigh.eth0.anycast_delay = 1024
net.ipv4.neigh.eth0.proxy_qlen = 64
net.ipv4.neigh.eth0.unres_qlen = 3
net.ipv4.neigh.eth0.gc_stale_time = 60
net.ipv4.neigh.eth0.delay_first_probe_time = 5
net.ipv4.neigh.eth0.base_reachable_time = 30
net.ipv4.neigh.eth0.retrans_time = 1024
net.ipv4.neigh.eth0.app_solicit = 0
net.ipv4.neigh.eth0.ucast_solicit = 3
net.ipv4.neigh.eth0.mcast_solicit = 3
net.ipv4.neigh.lo.locktime = 1024
net.ipv4.neigh.lo.proxy_delay = 819
net.ipv4.neigh.lo.anycast_delay = 1024
net.ipv4.neigh.lo.proxy_qlen = 64
net.ipv4.neigh.lo.unres_qlen = 3
net.ipv4.neigh.lo.gc_stale_time = 60
net.ipv4.neigh.lo.delay_first_probe_time = 5
net.ipv4.neigh.lo.base_reachable_time = 30
net.ipv4.neigh.lo.retrans_time = 1024
net.ipv4.neigh.lo.app_solicit = 0
net.ipv4.neigh.lo.ucast_solicit = 3
net.ipv4.neigh.lo.mcast_solicit = 3
net.ipv4.neigh.default.gc_thresh3 = 1024
net.ipv4.neigh.default.gc_thresh2 = 512
net.ipv4.neigh.default.gc_thresh1 = 128
net.ipv4.neigh.default.gc_interval = 30
net.ipv4.neigh.default.locktime = 1024
net.ipv4.neigh.default.proxy_delay = 819
net.ipv4.neigh.default.anycast_delay = 1024
net.ipv4.neigh.default.proxy_qlen = 64
net.ipv4.neigh.default.unres_qlen = 3
net.ipv4.neigh.default.gc_stale_time = 60
net.ipv4.neigh.default.delay_first_probe_time = 5
net.ipv4.neigh.default.base_reachable_time = 30
net.ipv4.neigh.default.retrans_time = 1024
net.ipv4.neigh.default.app_solicit = 0
net.ipv4.neigh.default.ucast_solicit = 3
net.ipv4.neigh.default.mcast_solicit = 3
net.ipv4.tcp_bic_beta = 819
net.ipv4.tcp_tso_win_divisor = 8
net.ipv4.tcp_moderate_rcvbuf = 1
net.ipv4.tcp_bic_low_window = 14

```
net.ipv4.tcp_bic_fast_convergence = 1
net.ipv4.tcp_bic = 1
net.ipv4.tcp_vegas_gamma = 2
net.ipv4.tcp_vegas_beta = 6
net.ipv4.tcp_vegas_alpha = 2
net.ipv4.tcp_vegas_cong_avoid = 0
net.ipv4.tcp_westwood = 0
net.ipv4.tcp_no_metrics_save = 0
net.ipv4.ipfrag_secret_interval = 600
net.ipv4.tcp_low_latency = 0
net.ipv4.tcp_frto = 0
net.ipv4.tcp_tw_reuse = 0
net.ipv4.icmp_ratemask = 6168
net.ipv4.icmp_ratelimit = 1024
net.ipv4.tcp_adv_win_scale = 2
net.ipv4.tcp_app_win = 31
net.ipv4.tcp_rmem = 4096    87380    174760
net.ipv4.tcp_wmem = 4096    16384    131072
net.ipv4.tcp_mem = 25165824    33554432    50331648
net.ipv4.tcp_dsack = 1
net.ipv4.tcp_ecn = 0
net.ipv4.tcp_reordering = 3
net.ipv4.tcp_fack = 1
net.ipv4.tcp_orphan_retries = 0
net.ipv4.inet_peer_gc_maxtime = 120
net.ipv4.inet_peer_gc_mintime = 10
net.ipv4.inet_peer_maxttl = 600
net.ipv4.inet_peer_minttl = 120
net.ipv4.inet_peer_threshold = 65664
net.ipv4.igmp_max_msf = 10
net.ipv4.igmp_max_memberships = 20
net.ipv4.route.secret_interval = 600
net.ipv4.route.min_adv_mss = 256
net.ipv4.route.min_pmtu = 552
net.ipv4.route.mtu_expires = 600
net.ipv4.route.gc_elasticity = 8
net.ipv4.route.error_burst = 5120
net.ipv4.route.error_cost = 1024
net.ipv4.route.redirect_silence = 20480
net.ipv4.route.redirect_number = 9
net.ipv4.route.redirect_load = 20
net.ipv4.route.gc_interval = 60
net.ipv4.route.gc_timeout = 300
net.ipv4.route.gc_min_interval = 0
net.ipv4.route.max_size = 134217728
net.ipv4.route.gc_thresh = 8388608
net.ipv4.route.max_delay = 10
net.ipv4.route.min_delay = 2
net.ipv4.icmp_errors_use_inbound_ifaddr = 0
net.ipv4.icmp_ignore_bogus_error_responses = 0
net.ipv4.icmp_echo_ignore_broadcasts = 0
net.ipv4.icmp_echo_ignore_all = 0
net.ipv4.ip_local_port_range = 1024    65000
net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_rfc1337 = 0
net.ipv4.tcp_stdurg = 0
net.ipv4.tcp_abort_on_overflow = 0
net.ipv4.tcp_tw_recycle = 0
net.ipv4.tcp_syncookies = 0
net.ipv4.tcp_fin_timeout = 60
net.ipv4.tcp_retries2 = 15
net.ipv4.tcp_retries1 = 3
net.ipv4.tcp_keepalive_intvl = 75
net.ipv4.tcp_keepalive_probes = 9
net.ipv4.tcp_keepalive_time = 7200
net.ipv4.ipfrag_time = 30
net.ipv4.ip_dynaddr = 0
net.ipv4.ipfrag_low_thresh = 196608
net.ipv4.ipfrag_high_thresh = 262144
net.ipv4.tcp_max_tw_buckets = 180000
net.ipv4.tcp_max_orphans = 8388608
net.ipv4.tcp_synack_retries = 5
net.ipv4.tcp_syn_retries = 5
net.ipv4.ip_nonlocal_bind = 0
net.ipv4.ip_no_pmtu_disc = 0
net.ipv4.ip_autoconfig = 0
net.ipv4.ip_default_ttl = 64
net.ipv4.ip_forward = 0
net.ipv4.tcp_retrans_collapse = 1
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 1
net.core.somaxconn = 1024
net.core.divert_version = 0.46
net.core.optmem_max = 20480
net.core.message_burst = 10
net.core.message_cost = 5
net.core.mod_cong = 290
net.core.lo_cong = 100
net.core.no_cong = 20
net.core.no_cong_thresh = 10
net.core.netdev_max_backlog = 300
net.core.dev_weight = 64
net.core.rmem_default = 1048576
net.core.wmem_default = 262144
net.core.rmem_max = 1048576
net.core.wmem_max = 262144
vm.percpu_pagelist_fraction = 0
vm.max_queue_depth = 0
vm.oom-kill = 1
vm.vfs_cache_pressure = 100
vm.block_dump = 0
vm.laptop_mode = 0
vm.max_map_count = 65536
vm.min_free_kbytes = 14186
vm.lower_zone_protection = 0
vm.hugetlb_shm_group = 500
vm.nr_hugepages = 750
vm.swappiness = 60
vm.nr_pdflush_threads = 2
vm.dirty_expire_centisecs = 3000
vm.dirty_writeback_centisecs = 500
vm.dirty_ratio = 40
vm.dirty_background_ratio = 10
vm.page-cluster = 3
vm.overcommit_ratio = 50
vm.overcommit_memory = 0
kernel.perfmon.debug_ovfl = 0
kernel.perfmon.debug = 0
kernel.panic_on_unrecovered_nmi = 0
kernel.wake_balance = 0
kernel.suid_dumpable = 0
kernel.ngroups_max = 65536
kernel.printk_ratelimit_burst = 10
kernel.printk_ratelimit = 5
kernel.panic_on_oops = 1
kernel.pid_max = 32768
kernel.sercons_esc = -1
kernel.overflowgid = 65534
kernel.overflowuid = 65534
kernel.pty.nr = 1
kernel.pty.max = 4096
kernel.random.uuid = 84461122-1fb9-4744-9a2d-1da673d3064b
kernel.random.boot_id = c1050a2a-0650-410f-a87f-1b05be0ab005
kernel.random.write_wakeup_threshold = 128
kernel.random.read_wakeup_threshold = 64
kernel.random.entropy_avail = 3584
kernel.random.poolsize = 512
kernel.threads-max = 786196
kernel.sysrq = 0
kernel.sem = 250    32000    100    128
kernel.msgmnb = 16384
kernel.msgmni = 16
kernel.msgmax = 8192
kernel.shmmni = 4096
kernel.shmall = 12582912
kernel.shmmax = 206158430208
kernel.acct = 4    2    30
kernel.hotplug = /sbin/hotplug
kernel.modprobe = /sbin/modprobe
kernel.printk = 5    4    1    7
kernel.ctrl-alt-del = 0
kernel.real-root-dev = 0
kernel.tainted = 0
kernel.core_pattern = core
kernel.core_uses_pid = 1
kernel.print-fatal-signals = 0
kernel.exec-shield-randomize = 1
kernel.exec-shield = 1
kernel.panic = 0
kernel.domainname = (none)
kernel.hostname = sp6952.cup.hp.com
kernel.version = #1 SMP Wed Jul 12 23:25:09 EDT 2006
kernel.osrelease = 2.6.9-42.EL
kernel.ostype = Linux
fs.mqueue.msgsize_max = 8192
fs.mqueue.msg_max = 10
fs.mqueue.queues_max = 256
fs.quota.syncs = 18
fs.quota.free_dquots = 0
fs.quota.allocated_dquots = 0
fs.quota.cache_hits = 0
fs.quota.writes = 0
fs.quota.reads = 0
fs.quota.drops = 0
fs.quota.lookups = 0
fs.aio-max-nr = 1048576
fs.aio-nr = 3072
fs.lease-break-time = 45
fs.dir-notify-enable = 1
fs.leases-enable = 1
fs.overflowgid = 65534
fs.overflowuid = 65534
fs.dentry-state = 13474    3520    45    0    0    0
fs.file-max = 4984409
fs.file-nr = 4599    0    4984409
fs.inode-state = 15904    1100    0    0    0    0
    0
fs.inode-nr = 15904    1100
fs.binfmt_misc.status = enabled
```

Config/Server/dbtune.ver

```
UNDO_TABLESPACE         = undo_1
aq_tm_processes = 0
compatible = 10.1.0.0.0
control_files = (/BUILD/dbs/tpcc_disks/control_001,/BUILD/dbs/tpcc_disks/control_002)
cursor_space_for_time = TRUE
db_block_checking=false
db_block_checksum       = false
dml_locks = 70

db_block_size         = 2048
shared_pool_size      = 5970M

db_cache_size = 25598M
db_8k_cache_size = 320M
db_16k_cache_size = 20032M
db_keep_cache_size = 105665M
db_recycle_cache_size =  34148M

db_files = 483
db_file_multiblock_read_count=1
db_name = tpcc
db_writer_processes     = 3
disk_asynch_io          = true
filesystemio_options            = SetAll

fast_start_mttr_target = 0
java_pool_size          = 0
job_queue_processes = 0
lock_sga        =TRUE
log_buffer = 10485760
log_checkpoint_interval  = 0
log_checkpoint_timeout   = 0
log_checkpoints_to_alert = true
parallel_max_servers = 0
pga_aggregate_target = 0
plsql_optimize_level = 2
processes = 300
query_rewrite_enabled = false
recovery_parallelism = 8
replication_dependency_tracking = false
sessions = 500
statistics_level        = basic
timed_statistics = false
trace_enabled=false
transactions = 500
undo_management         = auto
undo_retention          = 1
```

C.3 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          plebe56

  MAXACCESSERS          2266     # num_users + 50
  MAXGTT    1024
  MAXSERVERS            36    # num_servers + 5
  MAXSERVICES           165   # num_servers * #-of-services-each-server(5) + 10( for
BBL)
  MODEL     SHM
  LDBAL     N
  OPTIONS     NO_AA,NO_XA


# 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/tpcc/tuxconfs/TUXconfig.plebe56"
                                ROOTDIR="/project/tuxedo"
                                APPDIR="/project/tpcc/bin"
                                ULOGPFX="/tmp/TUXEDO_LOG"

# for debugging, put both into the same log on the same machine
#                               ULOGPFX="/home/tuxedo/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?
plebe56         LMID=plebe56
      TUXCONFIG="/project/tpcc/tuxconfs/TUXconfig.plebe56"
#---------------------------------------------------------------------
*GROUPS
#---------------------------------------------------------------------
group1          LMID=plebe56
              GRPNO=1

#---------------------------------------------------------------------
#---------------------------------------------------------------------

#---------------------------------------------------------------------
*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"
      MIN=31 MAX=31 RQADDR=tpcc_1 REPLYQ=Y SRVID=1
#---------------------------------------------------------------------
*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: 2005/04/11 09:40:08 $

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

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

#include "random.h"
#include "shm_lookup.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 */

void
neworder_gen(neworder_trans *t, ID warehouse, ID district)
   {
   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, db_size, 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;
   }

void
payment_gen(payment_trans *t, ID warehouse, ID district)
   {

   /* 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, db_size, 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);

   }

void
ordstat_gen(ordstat_trans *t, ID warehouse, ID district)
   {

   /* 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);
   }

void
delivery_gen(delivery_trans *t, ID warehouse, ID district)
{
   t->del.W_ID = warehouse;
   t->del.O_CARRIER_ID = RandomNumber(1,10);
}

void
stocklev_gen(stocklev_trans *t, ID warehouse, ID district)
   {
   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 const double weight[] = { 0.0,  0.0, .4301, .0401, .0402, .0401};
   int type;
   double r;

   /* choose a random number between 0.0 and 1.0 */
   if (trans_type == 0) {
                r = RandomValue();
      /*
       * 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;
   }
   /* return the value of the selected card, or NEWORDER if none selected */
   return type;
   }
```

Appendix E Disk Storage

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 minutes, 20 seconds during the steady state portion of the run. Each log file is 53730M.
So, to run for 8 hours, we need:

((8 * 60) / 29.3) * 53730 / 1024 = 859.59 GB (must be mirrored)

The log disk array has a total of 1435.54 GB (mirrored) available for the 8-hour recovery log.

The calculation for 60 day space at 409000 tpmC at 32,000 wh yields 12021.07GB. Each of the 12 data arrays have an available capacity of 1423.81GB for a total of 17085.72GB.

TPC-C 60-Day Space Requirements

TPM	**409,000**
NWARE	**32,000**

SEGMENT	TYPE	BLOCKS	BLOCK_SIZE	FIVE_PCT	DAILY_GROW	TOTAL
CUSTCLUSTER	CLUSTER	512,558,137	2,048	25,627,907	0	538,186,044
DB_STAT	SYS	1,048,576	2,048	0	0	1,048,576
DISTCLUSTER	CLUSTER	342,912	2,048	17,146	0	360,058
HIST	TABLE	29,681,280	2,048	0	6,069,822	35,751,102
ICUST1	INDEX	1,444,096	16,384	72,205	0	1,516,301
ICUST2	INDEX	27,020,766	2,048	1,351,038	0	28,371,804
IDIST	INDEX	80,512	2,048	4,026	0	84,538
IITEM	INDEX	5,632	2,048	282	0	5,914
IORDR2	INDEX	20,224,959	2,048	1,011,248	0	21,236,207
ISTOK	INDEX	4,274,944	16,384	213,747	0	4,488,691
ITEMCLUSTER	CLUSTER	8,446	2,048	422	0	8,868
IWARE	INDEX	20,512	2,048	1,026	0	21,538
NORDCLUSTER_QUEUE	CLUSTER	3,715,985	2,048	185,799	0	3,901,784
ORDRCLUSTER_QUEUE	CLUSTER	49,045,090	16,384	0	10,029,721	59,074,811
STOKCLUSTER	CLUSTER	457,666,200	2,048	22,883,310	0	480,549,510
SYSAUX	SYS	61,440	2,048	0	0	61,440
SYSTEM	SYS	204,800	2,048	0	0	204,800
SYS_IQ0000009628$$	INDEX	206,560	16,384	10,328	0	216,888
SYS_IQ0000009632$$	INDEX	285,845	2,048	14,292	0	300,137
WARECLUSTER	CLUSTER	34,752	2,048	1,738	0	36,490

SEGMENT	BLOCKS	BLOCK_SIZE	REQUIRED	STATIC	DYNAMIC	OVERSIZE
CUSTCLUSTER	550,896,640	2,048	538,186,044	538,186,044	0	12,710,596
DB_STAT	1,048,576	2,048	1,048,576	1,048,576	0	0
DISTCLUSTER	696,832	2,048	360,058	360,058	0	336,774
HIST	52,316,160	2,048	35,751,102	0	29,681,280	16,565,058
ICUST1	2,889,024	16,384	1,516,301	1,516,301	0	1,372,723
ICUST2	29,066,240	2,048	28,371,804	28,371,804	0	694,436
IDIST	164,352	2,048	84,538	84,538	0	79,814
IITEM	10,240	2,048	5,914	5,914	0	4,326
IORDR2	29,245,440	2,048	21,236,207	21,236,207	0	8,009,233
ISTOK	6,412,864	16,384	4,488,691	4,488,691	0	1,924,173
ITEMCLUSTER	10,240	2,048	8,868	8,868	0	1,372
IWARE	25,600	2,048	21,538	21,538	0	4,062
NORDCLUSTER_QUEUE	5,724,160	2,048	3,901,784	3,901,784	0	1,822,376
ORDRCLUSTER_QUEUE	94,885,120	16,384	59,074,811	0	49,045,090	35,810,309
STOKCLUSTER	687,452,160	2,048	480,549,510	480,549,510	0	206,902,650
SYSAUX	61,440	2,048	61,440	61,440	0	0
SYSTEM	204,800	2,048	204,800	204,800	0	0
SYS_IQ0000009628$$	94,885,120	16,384	216,888	216,888	0	94,668,232
SYS_IQ0000009632$$	5,724,160	2,048	300,137	300,137	0	5,424,023
WARECLUSTER	72,192	2,048	36,490	36,490	0	35,702

DYNAMIC	844,084,000	Initial MB for (History+Orders)
STATIC	2,248,305,496	Initial Blocks + 5% - Dynamic
DAILY_GROW	172,615,180	Total Dynamic [(calc. as (Initial Blocks)*tpmC/(WHS*62.5)]
DAILY_SPREAD	0	Oracle may be configured so that daily spread is 0
SPACE60	12,605,000,000	Static + 60*(Daily Growth+Daily Spread)
SPACE60 GB	12,309,570	Excludes OS, Paging and RDBMS Logs
OVERSIZE	2,645,587,836	
8-hour log (GB)	840.00	RDBMS Logs
Server swap (GB)	2	OS: Paging
Server OS (GB)	15	OS: UNIX File System
Total Space Needed	12,310,427	GB

Priced-Sytem Configuration	**Size in GB**	**Quantity**	**Total (GB)**
Log arrays: MSA1000 with 42 72.8GB disk drives in RAID 1 mode	1,435.540	1	1,435.54
Data arrays: MSA1000 with 504 36.4GB disk drives in RAID 0 mode	1,423.810	12	17,085.72
Total Storage in Priced System (GB)			18,521.26

Appendix F Price Quotes

The following pages contain the price quotes for the hardware included in this FDR.

TO: **Curt Thiem**

HP

Cupertino, CA 95014

10/16/2006

HP Unix Sales Development

19111 Pruneridge Avenue

Cupertino, CA 95014

(408) 447-2320

Description	Part Number	Brand	Price Key	US List Price	Qty	Price	3Year Main.Price
Server Hardware							
HP Integrity rx6600 Base System with 4 1.6GHz/24MB	AD134A, Opt 180		1	43,845	1	43,845	
I/O backplane	AD296A		1	0	1	0	
48-DIMM memory carrier board	AD127A		1	4,495	1	4,495	
72GB/10kpm SAS hard disk drive	AD140A		1	356	4	1,424	
DVD-ROM slimline drive	AD142A		1	230	1	230	
16GB memory quad (4x4GB DIMMS)	AB566A**		1	18,977	12	227,724	
3 Year Svc & Support Price (Hardware and Software)	HA110A3						$7,267
PCI 4GB Fibre Channel Adapter (dual port)	AB379A		1	3,995	2	7,990	
Redundant Power Supply	AD052A		1	749	1	749	
HP Server Thin Client (monitor, keyboard/mouse, cable i	AB300B		1	500	1	500	
Rack Model 5642	358254-B21		1	689	1	689	
				Subtotal		**287,646**	**7,267**
Server Software							
Red Hat Enterprise Linux AS 4	T2763AA**		1	0	1	0	
RHEL4 Update 4	T2763AA, Opt 004**		1	0	1	0	
Red Hat Linux Subscription Licience for 3 years	T2763AA, Opt 324**			1,908	1	1,908	
HP Support for Red Hat 7x24	HA107A3-6L4**		1		1		4,839
				Subtotal		**1,908**	**4,839**
Storage							
Rack System/E R3000 XR UPS	AF422A		1	1,366	7	9,562	
HP StorageWorks SAN Switch 2/8 (incl 4 SFP)	288247-B21		1	3,599	3	10,797	
2meter Fibre Optic Cable	221691-B21		1	77	12	924	
5 meter Fibre Optic Cable	221691-B22		1	82	4	328	
HP StorageWorks MSA 1000 (13 + 2 spares)	201723-B22		1	6,995	15	104,925	Included
HP StorageWorks MSA 1000 Controller	218231-B22		1	4,290	1	4,290	
HP StorageWorks Enclosures MSA30 (26 + 3 spares)	302969-B21		1	2,829	29	82,041	
36GB 15K Ultra320 Hard Drive (504 + 51 spares)	286776-B22		1	269	555	149,295	
73GB 15K Ultra320 Hard Drive (42 + 5 spares)	286778-B22		1	399	47	18,753	
10642 (G2) Rack Cabinet	AF001A		1	1,249	4	4,996	
200 - 240 volts North America	A5137AZ AW4		1	94	16	1,504	
				Subtotal		**387,415**	**0**
Client Hardware							
HP server rx2620 w 1.3GHz Intel Itanium 2 Processor	AB333A		1	3,795	8	30,360	
1.3GHz Intel Itanium 2 Processor	AB336A		1	1,650	8	13,200	
3 Year Support Price (Hardware & Software)	HA110A3						31,616
36GB 15K RPM Ultra320 SCSI Internal Disk	AD186A		1	389	8	3,112	
4GB Memory Module (4 x 1GB DIMMS)	AB397A		1	2,550	16	40,800	
HP Server Thin Client (monitor, keyboard/mouse, cable i	AB300B		1	500	1	500	
HP CAT5 Lan Cables (qty 4)	263474-B24		1	29	3	87	
10642 (42U) Rack Cabinet	AF001A		1	1,249	1	1,249	
200-240 Volts Power Option	A5137AZ, Opt AW4		1	94	4	376	
HP-UX Fndn OE DVD Media	B9106AA, Opt. AJR		1	565	1	565	
HP-UX Fndn OE DVD Media, Factory Integrated	B9106AA, Opt. OD1		1	199	8	1,592	
HP-UX Fndn OE Integrity PPL max2CPU w/sys	B9430AC		1	455	16	7,280	
				Subtotal		**99,121**	**31,616**
Client Software							
HP C++/ANSI C Developer's Bundle	B9007AA, Opt. 2AH		1	966	1	966	256
				Subtotal		**966**	**256**
User Connectivity							
HP ProCurve Switch 2724	J4897A		1	1,599	1	1,599	**636**
				Subtotal		**1,599**	**636**
HP's Large configuration Discount and Support Prepayment*			4			($1,853,714)	($13,107)
				Total		**(1,075,059)**	**31,507**

*A 24.1% discount was based on the overall value of the specific components from HP (Price Key) in this single quotation. Discounts for similarly sized configurations will be similar to those quoted here, but may vary based on the components in the quotation.

From: Mary.Beth Pierantoni [mailto:mary.beth.pierantoni@oracle.com]
Sent: Thursday, October 05, 2006 10:03 PM
To: Boushey, Lucille
Subject: Oracle Pricing

Product	Price	Quantity	Extended Price
Oracle Database 10g Enterprise Edition, Per Processor, Unlimited Users, 3 years	$20,000	4*	$80,000
Oracle Database Server Support Package for 3 years	$2,000	3	$6,000
Oracle Mandatory E-Business Discount			<$8,600>
Oracle TOTAL			$77,400

*** 4 = 0.50 * 8. Explanation: For the purposes of counting the number of processors which require licensing, an Intel multicore chip with "n" cores shall be determined by multiplying "n" cores by a factor of 0.50.**

Oracle pricing contact: MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com, 916-315-5081

This quote is valid for 90 days.

October 12, 2006

Lucille Boushey
Hewlett Packard Company
(408) 447-7364; Phone
(408) 447-5958; Fax

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, 7.1,8.0,8.1,9.0, and 9.1. Please note that Tuxedo 9.0 is our most recent version of Tuxedo. Core functionality services (CFS)-R pricing is appropriate for your activities. As per the table below HP/Compaq systems are classified as either a Tier 1, 2, 3, 4 or 5 systems depending on the performance and CPU capacity of the system. The HP RX 2620 machines are Tier 1 machines – price is $1,200 per server (License), eligible for a 5% discount = $1,140 per server + $252 per server (7x24) for support – support is non discountable. This quote is valid for 60 days from the date of this letter.

Tuxedo Core Functionality Services (CFS-R) Program Product Pricing and Description

TUX-CFS-R 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-R 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, 6.5,7.1,8.0, 8.1,9.0, and 9.1. Prices range from $1,200 for Tier 1 to $100,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS-R at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

Very Truly Yours,

Robert J. Gieringer

Rob Gieringer,
Worldwide Pricing Manager

BEA Tux/CFS-R 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**	**$1,200.00**	**$216**	**$252**
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations with up to 2 CPUs	**Unlimited**	**$4,800.00**	**$864**	**$1,008**
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	**Unlimited**	**$12,000.00**	**$2,160**	**$2,520**
Tier 4 - Large (more than 8, less than 32 CPUs)	**Unlimited**	**$40,000.00**	**$7,200**	**$8,400**
Tier 5 - Massively Parallel Systems, > 32 processors	**Unlimited**	**$100,000.00**	**$18,000**	**$21,000**

	Tier 1	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Operating System						
Linux AS4	**Uni-processor Workstation** **B Class - 132/180/2000** **C Class (3000/3600 / 3700)** **2P Client Machines** **Compaq DL360**	**9000/E25** **9000/E35** **9000/E45** **9000/E55** **9000/G30** **9000/G40** **9000/A180** **9000/A180C** **9000 /A400** **RX 2600** **RX 2620**	**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**	**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** **RP8400** **Superdome < 32 CPU's** **RP8420** **SuperDome (PA-8800 bases) w/ < 32 dual-core CPU's**	**9000/V series all models** **X-Class** **Superdome >= 32 CPU's** **SuperDome (PA-8800 bases) w/ >= 32 dual-core CPU's**