**hp server rx5670**
***using***
**HP-UX 11.i, v2, 64-bit Base**
***and***
**Oracle Database 10G  Standard Edition**

# TPC Benchmark® C Full Disclosure Report

**Second Edition**

**Submitted for Review**
**December 31, 2003**

Second Edition - December 31, 2003

Printed in U.S.A., December 31, 2003.

# Abstract

## Overview

This report documents the methodology and results of the TPC Benchmark® C test conducted on the hp server rx5670 in a client/server configuration, using Oracle Database 10G Standard Edition and the TUXEDO 8.0 transaction monitor. The operating system used for the benchmark was Hewlett-Packard's HP-UX 11.i, v2, 64-bit Base . The application was written in C and compiled using HP C/ANSI C/HP-UX.

## TPC Benchmark C Metrics

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

## Standard and Executive Summary Statements

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

## Auditor

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

## Standard System Summary

| **Company Name** | **System Name** | **Database Software** | **Operating System Software** |
|---|---|---|---|
| Hewlett-Packard Company | hp server rx5670 | Oracle Database 10G Standard Edition | HP-UX 11.i, v2, 64-bit Base |
| HP H/W Availability Date - Now<br>S/W Availability Date - January 30, 2004 | | | |
| **Total System Cost** | **TPC-C® Throughput** | **Price/Performance** | |
| Hardware<br>Software<br>3-year maintenance | Sustained maximum throughput of System running TPC-C® expressed in transactions per minute | Total system cost/tpmC<br>($954,061/131639.8) | |
| **$954,061** | **131,639.80 tpmC** | **$7.25 per tpmC** | |

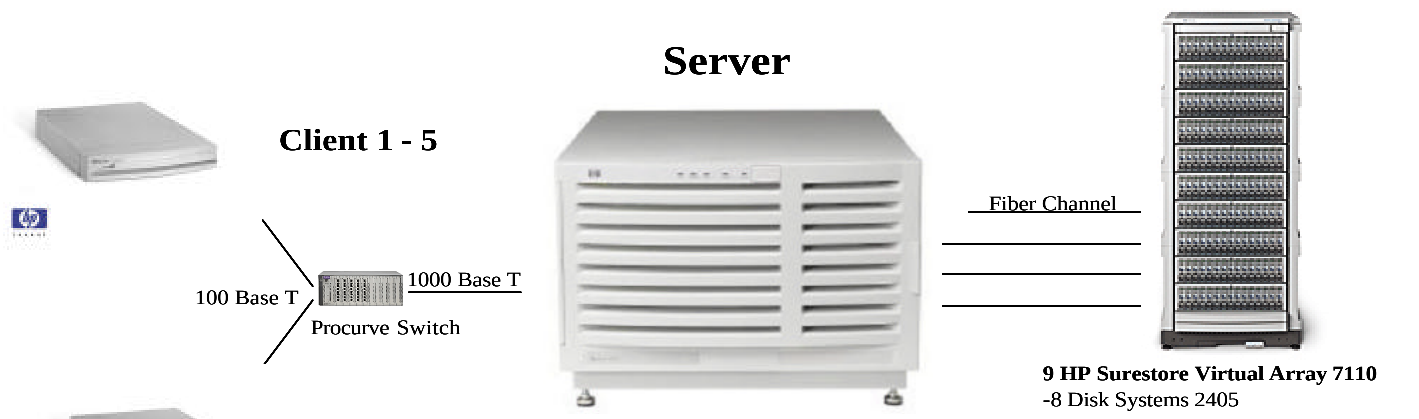
# hp server rx5670

TPC-C Revision 5

Report Date: December 31, 2003

| Total System Cost | TPC Throughput | Price/Performance | Availability Date |
|---|---|---|---|
| **$954,061** | **131,639.80 tpmC** | **$7.25/tpmC** | **January 30, 2004** |

| Processors | Database Manager | Operating System | Other Software | Number of Users |
|---|---|---|---|---|
| **4 Intel Itanium2 1.5GHz** | **Oracle Database 10G Standard Edition** | **HP-UX 11.i, v2, 64-bit Base** | **TUXEDO 8.0** | **104,400** |



**9 HP Surestore Virtual Array 7110**
-8 Disk Systems 2405
-240 36GB 15K RPM Disk Drives
-15 73GB 10K RPM Disk Drives

**hp server rx5670**
4 – 1.5GHz Intel Itanium2
w/6GB L3 cache
96GB Memory
3 73GB Internal Disk

**Clients – 9 hp rp2470 servers**

| ***System Components*** | ***Server (hp server rx5670)*** | | ***each Client (9 hp server rp2470)*** | |
|---|---|---|---|---|
| | ***Qty*** | ***Type*** | ***Qty*** | ***Type*** |
| **Processors** | 4 | 1.5GHz Intel Itanium2 | 1 | 750MHz PA-RISC 8700 |
| **Cache Memory** | each | 6MB L3 cache | each | .75MB I-cache/1.5 MB D-cache |
| **Memory** | 96 | GB | 1 | 3 GB |
| **Disk Controllers** | 3 | 2GB Fibre Channel Adapter | 1 | Ultra2 SCSI LVD |
| **Disk Drives** | 9 | Surestore Virtual Array 7110, with 240 36GB 15K RPM Disks, and 15 73GB 10K RPM Disks | 1 | 36 GB |
| | 3 | 73GB 15K RPM Ultra 320 SCSI Internal Disks | | |
| **Total Storage** | 4,633.65 | GB | | |
| **Tape Drives** | 1 | DVD ROM | | |
| **Terminals** | 1 | Console Terminal | 1 | Console Terminal |

# hp server rx5670

TPC-C Rev 5

Report Date: December 31, 2003

| Description | Part Number | Brand | Price Key | US List Price | Qty | Price | 3Year Main.Price |
|---|---|---|---|---|---|---|---|
| **Server Hardware** | | | | | | | |
| **Server Hardware** | | | | | | | |
| hp server rx5670 with 1.5GHz processor | A6838B | | 1 | 26,494 | 1 | 26,494 | |
| 1.5GHz Processor with 6MB Cache | A9810A | | 1 | 8,250 | 3 | 24,750 | |
| 3 Year Support Price (Hardware and Software) | | | 1 | | | | 15,259 |
| 8GB PC2100 DDR-SDRAM Memory (4x2GB DIMMs) | A6835A | | 1 | 16,000 | 12 | 192,000 | |
| Memory Carrier Board for rx5670 | A6747A | | 1 | 1,981 | 2 | 3,962 | |
| 73GB 15K RPM Ultra320 SCSI Internal Drive (incl. 10% | A9760A | | 1 | 1,362 | 5 | 6,810 | |
| DVD ROM | A5557B | | 1 | 450 | 1 | 450 | |
| Sytstem Console | C1099A | | 1 | 550 | 1 | 550 | |
| 2GB Fibre Channel Adapter | A6795A | | 1 | 2,240 | 3 | 6,720 | |
| | | | **Subtotal** | | | **261,736** | **15,259** |
| **Server Software** | | | | | | | |
| Oracle Database 10G Standard Edition | | | | | | | |
| Processor for 3 year term for 4 processors, Unlimited Users | | Oracle | 2 | 7,500 | 4 | 30,000 | |
| Oracle Database Server Support Package for 3 years | | | 2 | 6,000 | 1 | | 6,000 |
| HPUX 11i, V2 Foundation Operating Environment | B9429AC | | 1 | 2,370 | 4 | 9,480 | |
| Foundation Operating Environment Media Kit | B9106A, Opt OD1 | | 1 | 199 | 1 | 199 | |
| | | | **Subtotal** | | | **39,679** | **6,000** |
| **Storage** | | | | | | | |
| Rack System/E R3000 XR UPS | J4367A | | 1 | 1,948 | 4 | 7,792 | |
| Surestore VA 7110 w/dual controllers 512MB cache | A7294AZ | | 1 | 50,880 | 9 | 457,920 | |
| 3 Year Support Price | | | | | | | 38,714 |
| Disk System 2405 with dual 2GB link cards | A6250AZ | | 1 | 6,595 | 8 | 52,760 | |
| 36GB 15K RPM FC HDD. | A6193A, Opt 0D1 | | 1 | 1,349 | 240 | 323,760 | |
| 36GB 15K RPM FC HDD. (10% spare) | A6193A, Opt 0D1 | | 1 | 1,349 | 24 | 32,376 | |
| 73GB 10K RPM FC HDD. | A6194A, Opt 0D1 | | 1 | 2,019 | 15 | 30,285 | |
| 73GB 10K RPM FC HDD. (10% spare) | A6194A, Opt 0D1 | | 1 | 2,019 | 2 | 3,029 | |
| 2 meter LC Fibre Optic Cable | C7524A | | 1 | 215 | 24 | 5,160 | |
| 16 meter LC/SC Fibre Optic Cable | C7525A | | 1 | 260 | 3 | 780 | |
| HP FC 1GB/2GB Entry Switch 8B, Field Rack | A7346A | | 1 | 6,599 | 2 | 13,198 | |
| HP9000 Std. Rack System E41 | A4902A | | 1 | 1,910 | 3 | 5,730 | |
| Modular Power Dist. | A5137AZ | | 1 | 145 | 12 | 1,740 | |
| 200-240 Volts Power Option | A5137AZ, Opt AW4 | | 1 | 94 | 12 | 1,128 | |
| | | | **Subtotal** | | | **935,658** | **38,714** |
| **Client Hardware** | | | | | | | |
| HP server rp2470 | A6890A | | 1 | 1,865 | 9 | 16,785 | |
| 750Mhz PA-RISC 8700 CPU | A6892A | | 1 | 4,500 | 9 | 40,500 | |
| 3 Year Support Price (Hardware and Software) | | | | | | | 39,987 |
| 36GB 15K HotPlug Ultra 160 SCSI Internal Disk | A6948A | | 1 | 1,298 | 9 | 11,682 | |
| 2GB Memory Module | A6114A | | 1 | 3,400 | 9 | 30,600 | |
| 1GB Memory Module | A5841A | | 1 | 1,900 | 9 | 17,100 | |
| HP-UX 11.i Sys Media, CD-ROM | B3920EA, Opt. OD1 | | 1 | 195 | 9 | 1,755 | |
| | | | **Subtotal** | | | **118,422** | **39,987** |
| **Client Software** | | | | | | | |
| HP C/ANSI C Compiler | B3901BA, Option AH0 | | 1 | 1,600 | 1 | 1,600 | 170 |
| BEA Tuxedo 8.0 | | Bea Sys. | 3 | 1,140 | 9 | 10,260 | 6,804 |
| | | | **Subtotal** | | | **11,860** | **6,974** |
| **User Connectivity** | | | | | | | |
| HP ProCurve Switch 4000M | J4121A | | 1 | 2,379 | 1 | 2,379 | 588 |
| HP ProCurve Switch 100/1000Base-T Mod. | J4115B | | 1 | 509 | 1 | 509 | |
| | | | **Subtotal** | | | **2,888** | **588** |
| Oracle Mandatory E-Business Discount (license and support) | | | 2 | (1,800) | 1 | (1,800) | |
| HP's Large configuration Discount and Support Prepayment* | | | | | | (491,728) | (30,175) |
| *All discounts are based on US list prices and for similar quantities and configurations | | | **Total** | | | **876,714** | **77,347** |

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

Audited by Lorna Livingtree for Performance Metrics, Inc.

| | |
|---|---|
| **Three Year Cost of Ownership:** | **$954,061** |
| **tpmC Rating:** | **131,639.80** |
| **$/tpmC:** | **$7.25** |

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

# Numerical Quantities Summary for hp server rx5670

| | |
|---|---|
| **MQTH, Computed Maximum Qualified Throughput** | **131,639.80 tpmC** |

## Response Times (in seconds)

| | 90th %-ile | Maximum | Average |
|---|---|---|---|
| New-Order | 0.55s | 12.45s | 0.30s |
| Payment | 0.54s | 12.07s | 0.29s |
| Order-Status | 0.56s | 9.99s | 0.31s |
| Delivery (interactive portion) | 0.21s | 9.30s | 0.12s |
| Delivery (deferred portion) | 0.55s | 9.93s | 0.30s |
| Stock-Level | 0.56s | 10.48s | 0.31s |
| Menu | 0.1000s | 0.00271s | 0.000039s |

## Transaction Mix, in percent of total transactions

| | |
|---|---|
| New-Order | 44.97% |
| Payment | 43.00% |
| Order-Status | 4.01% |
| Delivery | 4.020% |
| Stock-Level | 4.01% |

## Keying/Think Times

| | Keying Time | | | Think Time | | |
|---|---|---|---|---|---|---|
| | Min | Avg | Max | Min | Avg | Max |
| New-Order | 18.02s | 18.03s | 18.31s | 0.01s | 12.14s | 189.93s |
| Payment | 3.01s | 3.02s | 3.31s | 0.01s | 12.06s | 205.69s |
| Order-Status | 2.01s | 2.02s | 2.31s | 0.01s | 10.1s | 144.01s |
| Delivery (interactive) | 2.01s | 2.02s | 2.31s | 0.01s | 5.06s | 82.9s |
| Stock-Level | 2.01s | 2.02s | 2.31s | 0.01s | 5.07s | 75.11s |

## Test Duration

| | | |
|---|---|---|
| Ramp up time | 42.5 | minutes |
| Measurement interval | 120 | minutes |
| Transactions during measurement interval | 35,131,261 | |
| Ramp down time | 18.083 | minutes |

## Checkpointing

| | | |
|---|---|---|
| Number of checkpoints in measurement interval | 4 | |
| Checkpoint Interval | 29.7 | minutes |

## TPC Benchmark C Overview

This is the full disclosure report for a benchmark test of the hp server rx5670 using Oracle Database 10G Standard Edition . It meets the requirements of the TPC Benchmark® C Standard Specification, Revision 5 dated March, 2001.

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

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

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

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

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

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

# 1 General Items

## 1.1 Application Code and Definition Statements

*The application program (as defined in clause 2.1.7) must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input output functions.*

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

## 1.2 Test Sponsor

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

The 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 HP-UX operating system parameters used to generate the kernel for the configuration used in this benchmark. Also included are all of the Oracle Database 10G Standard Edition database parameters and the TUXEDO 8.0 transaction monitor parameters used.

## 1.4 Configuration Diagrams

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

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

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

- 4 1.5GHz Intel Itanium2 System Processors
- 96GB of memory

- 3 73GB 15RPM  Ultra320 intenal drives
- 3 2GB Fibre Channel Adapters
- 9  Surestore Virtual Array 7110 (with 240 36GB 15K RPM, and  15 73GB 10K RPM disk drives
- Two LAN interfaces

As indicated in Figure 1.2, this benchmark configuration used Remote Terminal Emulator (RTE) programs that executed on 9 rp2470 Enterprise Server drivers to emulate TPC-C user sessions.  The emulated users on the driver systems were connected to the client systems under test via 9 separate 100 Base-T local area network (LAN) connections through a HP Procurve 4000M, with 9 separte connections to the 9 client systems.

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

**Figure 1.1: hp server rx5670 Priced Configuration**

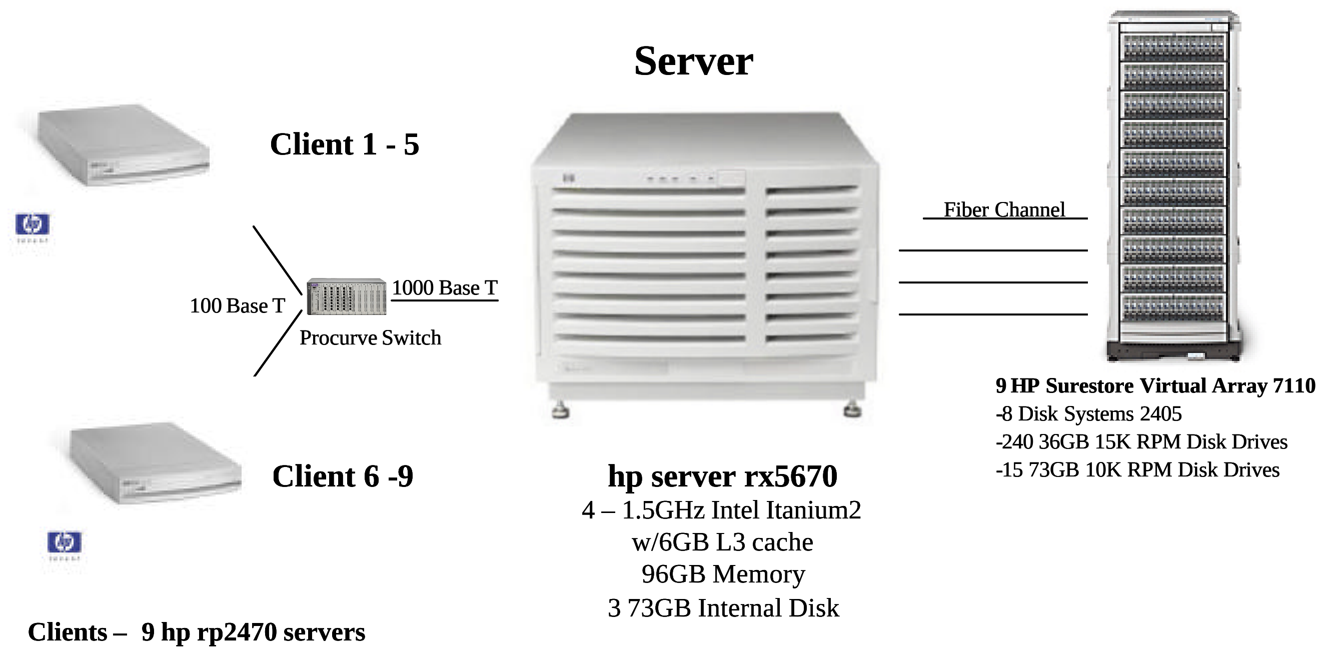
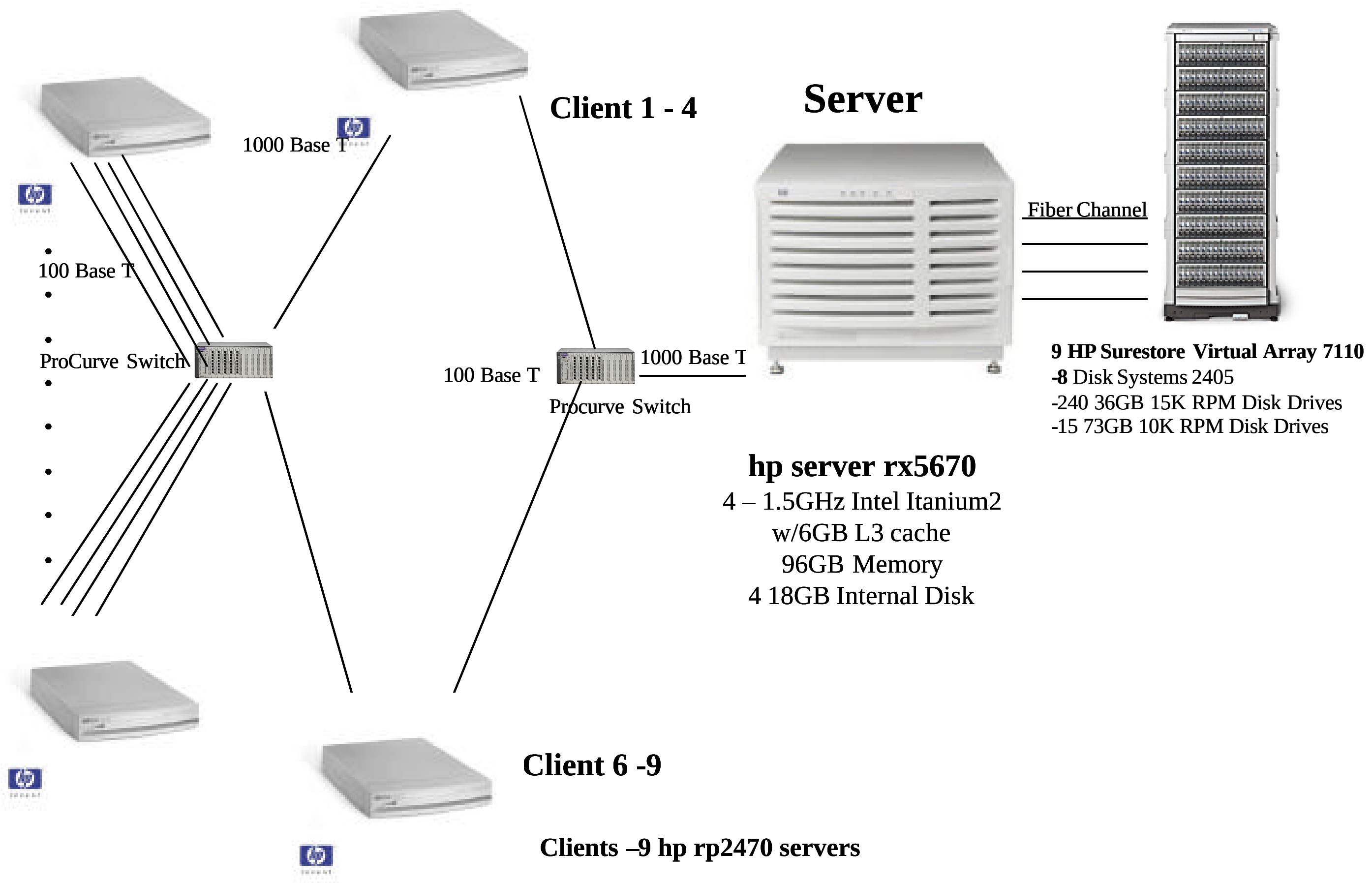


**Figure 1.2: hp server rx5670 Benchmark Configuration**



# 2 Clause 1 Related Items

## 2.1 Table Definitions

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

Appendix B describes the programs that define, create, and populate the Oracle Database 10G Standard Edition database for TPC-C® testing.

## 2.2 Physical Organization of Database

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

Space was allocated to Oracle Database 10G Standard Edition according to the data in section 5.2. The size of the database table space on each disk drive was calculated to provide even distribution of load across the disk drives.

## 2.3 Insert and Delete Operations

*It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C® transaction mix. Furthermore, any restrictions in the SUT database implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows.*

There were no restrictions on insert and delete operations to any tables.

## 2.4 Partitioning

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

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

# 3 Clause 2 Related Items

## 3.1 Random Number Generation

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

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

## 3.2 Input/Output Screen Layout

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

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

## 3.3 Priced Terminal Feature Verification

*The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained. Although not specifically priced, the type and model of the terminals used for the demonstration in 8.1.3.3 must be disclosed and commercially available (including supporting software and maintenance).*

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

## 3.4 Presentation Manager or Intelligent Terminal

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

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

A presentation manager was not used.

**Table 3.1: Transaction Statistics**

| Type | Item | Value |
|---|---|---|
| New Order | Home warehouse items | 99.00% |
| | Remote warehouse items | 1.00% |
| | Rolled back transactions | 1.00% |
| | Average items per order | 10.00 |
| Payment | Home warehouse | 85.02% |
| | Remote warehouse | 14.98% |
| | Non primary key access | 59.98% |
| Order Status | Non primary key access | 60.02% |
| Delivery | Skipped transactions | 0 |
| Transaction Mix | New Order | 44.97% |
| | Payment | 43.00% |
| | Order Status | 4.01% |
| | Delivery | 4.020% |
| | Stock Level | 4.01% |

## 3.5 Transaction Statistics

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

## 3.6 Queuing Mechanism

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

Delivery transactions were submitted to servers using the same TUXEDO mechanism that other transactions used. The only difference was that the call was asynchronous, i.e., control would return to the client process immediately and the deferred delivery part would complete asynchronously.

# 4 Clause 3 Related Items

## 4.1 Transaction System Properties (ACID)

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

The TPC Benchmark® C Standard Specification defines a set of transaction processing system properties that a system under test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation, and Durability (ACID). This section quotes the specification definition of each of these properties and describes the tests done as specified and monitored by the auditor to demonstrate compliance.

## 4.2 Atomicity

*The system under test must guarantee that transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially-completed operations leave any effects on the data.*

### 4.2.1 Completed Transaction

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

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

### 4.2.2 Aborted Transaction

*Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2.5.1.2) and substitute a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the records in the CUSTOMER, WAREHOUSE, and DISTRICT tables have NOT been changed*

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

## 4.3 Consistency

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

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

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

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

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

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

## 4.4 Isolation

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

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

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

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

### 4.4.1 Isolation Test 1

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

The execution of the above test proceeded as follows:

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

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

### 4.4.2 Isolation Test 2

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

The execution of the above test proceeded as follows:

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

### 4.4.3 Isolation Test 3

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

The execution of the above test proceeded as follows:

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

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

### 4.4.4 Isolation Test 4

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

The execution of the above test proceeded as follows:

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

### 4.4.5 Isolation Test 5

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

The execution of the above test proceeded as follows:

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

### 4.4.6 Isolation Test 6

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

The execution of the above test proceeded as follows:

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

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

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

### 4.4.7 Isolation Test 7

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

The execution of the above test proceeded as follows:

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

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

### 4.4.8 Isolation Test 8

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

The execution of the above test proceeded as follows:

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

### 4.4.9 Isolation Test 9

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

The execution of the above test proceeded as follows:

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

## 4.5 Durability

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

*List of single failures:*

- *Permanent irrecoverable failure of any single durable medium containing TPC-C database tables or recovery log data.*
- *Instantaneous interruption (system crash / system hang) in processing which requires system reboot to recover.*
- *Failure of all or part of memory (loss of contents)…*

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

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

A test was performed under a load of 104,400 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 104,400 users on the full-scale database built for 110,000 users.

### 4.5.1 Loss of Log and Data Disks

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

1. The D_NEXT_O_ID fields for all rows in the DISTRICT table were summed up to determine the initial count of the total number of orders (count1).
2. A test was initiated with 104,400 terminals. On the driver system, completed/rolled-back transactions (including New-Orders) were recorded in a "success" file.
3. After running at steady state throughput levels for 5 minutes, a individual disk containing recovery log was unplugged from the array.
4. Because of the built-in redundancy in the disk array, the test continued normally.
5. On the system log files, messages appeared indicating that a disk were missing.
6. After running again at steady state throughput levels for 5 minutes, a individual disk containing data was unplugged from the array.
7. Because of the built-in redundancy in the disk array, the test continued normally.
8. On the system data files, messages appeared indicating that a disk were missing.
9. The test was finished on the driver.
10. The contents of the "success" file on the driver and the ORDER table were spot-compared to verify that records in the "success" file for completed New-Order transactions had corresponding records in the ORDER table.
11. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 matched exactly the number of records for successful New Orders in the RTE "success" file.
12. Consistency test 3 was run on the database and the results were verified.
13. New disks were installed. The disk arrays automatically copied the mirrored-pair mate of the missing disks onto the new disks. Messages appeared in the system log files indicating redundancy was restored.

### 4.5.2 Instantaneous Interruption and Loss of Memory

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

1. The D_NEXT_O_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).
2. Transactions were started at full load. On the driver system, completed/rolled-back transactions (including New-Orders) were recorded in a "success" file.
3. After thirteen minutes, the benchmark throughput reached the steady state level and the server systems were de-powered.
4. The test was aborted on the driver.

5. The server system was restarted.
6. The database was restarted and a recovery performed using the transaction log.
7. The contents of the "success" file on the driver and the ORDERs table were spot-compared to verify that records in the "success" file for completed New-Order transactions had corresponding records in the ORDERs table.
8. Step 1 was repeated to determine the current total number of orders (count2). Count2-count1 (=339,466,987-337,045,482=2,421,505) was 15 more than the number of records for successful New Orders in the RTE "success" file (=2,445,885-24,395=2,421,490completed). *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 11,000 warehouses.

| Table | Occurrences |
|---|---:|
| Warehouse | 11,000 |
| District | 110,000 |
| Customer | 330,000,000 |
| History | 330,000,000 |
| Orders | 330,000,000 |
| New Orders | 99,000,000 |
| Order Line | 3,300,117,152 |
| Item | 100,000 |
| Stock | 1,100,000,000 |

During the measurement only 10,440 warehouses and their associated data were accessed. This was confirmed using D_NEXT_O_1D and W_YTD as described in *Clause 4.2.2 Comment (2)*.

## 5.2 Database and Growth Layout

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

Table 5.2 indicates the distribution of the databas e tables over the disks of the tested and priced systems.

I) root, swap, file systems:
   =======================

| Use | Device | Size (GB) | Device Model |
|---|---|---|---|
| root+swap | /dev/dsk/c0t2d0s2 | 18 | SEAGATE ST118202LC |
| file system | /dev/dsk/c0t0d0s2 | 18 | HP 18.2GMAN3184MC |
| file system | /dev/dsk/c3t0d0 | 18 | SEAGATE ST318404LC |
| file system | /dev/dsk/c3t2d0 | 18 | SEAGATE ST318203LC |
| swap | /dev/dsk/c62t0d5 | 98304 | HP StorageWorks Virtual Array 7110 |

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

We used 9 arrays. Every array is attached to the system via a Fibre Channel link.

Each array contains 30 36GB or 15 73GB disks drives, allocated to 3 to 7 RAID1 LUNs. After formatting and mirroring, the available capacity of the arrays with 30 36GB drives is 491.14GB. The capacity of the array with 73-GB drives is 491.14GB.

One array contains the redo log, and swap space. It has enough space for 8 hours of redo logs exclusive of the swap space (see above).

On the 8 data arrays, LUN 0 is 92GB and is allocated to volume group vgtpcc_raw. Some of the Oracle files are striped across the 8 arrays in this volume groups. The other 46 LUNS are mapped 1-to-1 to files that contain the customer table, customer index icust2, the stock table.

The 9 arrays and their luns are accessed via the following paths:

```
/dev/dsk/c44t0[d0|d2|d3|d4|d5|d6|d7]
/dev/dsk/c45t0[d0|d2|d3|d4|d5|d6|d7]
/dev/dsk/c48t0[d0|d2|d3|d4|d5|d6|d7]
/dev/dsk/c49t0[d0|d2|d3|d4|d5|d6|d7]
/dev/dsk/c52t0[d0|d2|d3|d4|d5|d6|d7]
/dev/dsk/c53t0[d0|d2|d3|d4|d5|d6]
/dev/dsk/c56t0[d0|d2|d3|d4|d5|d6]
/dev/dsk/c57t0[d0|d2|d3|d4|d5|d6|d7]
/dev/dsk/c62t0[d1|d2]
```

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

| DATAFILE | FILE_ID | SIZE(MB) | TABLESPACE | ACTUAL DISK/LV PATH |
|---|---|---|---|---|
| aux.df | 2 | 120 | SYSAUX | /dev/vgtpcc_raw/r10i_lvm_aux.df |
| control_001 | 5.02 | | | File System file |
| cust_0_0 | 5 | 15900 | CUST_0 | /dev/rdsk/c52t0d4 |
| cust_0_1 | 6 | 15900 | CUST_0 | /dev/rdsk/c52t0d7 |
| cust_0_10 | 15 | 15900 | CUST_0 | /dev/rdsk/c48t0d2 |
| cust_0_11 | 16 | 15900 | CUST_0 | /dev/rdsk/c57t0d2 |
| cust_0_12 | 17 | 15900 | CUST_0 | /dev/rdsk/c52t0d6 |
| cust_0_13 | 18 | 15900 | CUST_0 | /dev/rdsk/c45t0d3 |
| cust_0_14 | 19 | 15900 | CUST_0 | /dev/rdsk/c48t0d7 |
| cust_0_15 | 20 | 15900 | CUST_0 | /dev/rdsk/c45t0d6 |
| cust_0_16 | 21 | 15900 | CUST_0 | /dev/rdsk/c48t0d4 |

| | | | | |
|---|---|---|---|---|
| cust_0_17 | 22 | 15900 | CUST_0 | /dev/rdsk/c45t0d7 |
| cust_0_18 | 23 | 15900 | CUST_0 | /dev/rdsk/c48t0d6 |
| cust_0_19 | 24 | 15900 | CUST_0 | /dev/rdsk/c44t0d7 |
| cust_0_2 | 7 | 15900 | CUST_0 | /dev/rdsk/c57t0d6 |
| cust_0_20 | 25 | 15900 | CUST_0 | /dev/rdsk/c56t0d6 |
| cust_0_3 | 8 | 15900 | CUST_0 | /dev/vgtpcc_raw/r10i_lvm_cust_0_3 |
| cust_0_4 | 9 | 15900 | CUST_0 | /dev/vgtpcc_raw/r10i_lvm_cust_0_4 |
| cust_0_5 | 10 | 15900 | CUST_0 | /dev/vgtpcc_raw/r10i_lvm_cust_0_5 |
| cust_0_6 | 11 | 15900 | CUST_0 | /dev/vgtpcc_raw/r10i_lvm_cust_0_6 |
| cust_0_7 | 12 | 15900 | CUST_0 | /dev/vgtpcc_raw/r10i_lvm_cust_0_7 |
| cust_0_8 | 13 | 15900 | CUST_0 | /dev/vgtpcc_raw/r10i_lvm_cust_0_8 |
| cust_0_9 | 14 | 15900 | CUST_0 | /dev/rdsk/c49t0d3 |
| dist_0_0 | 26 | 537 | DIST_0 | /dev/vgtpcc_raw/r10i_lvm_dist_0_0 |
| hist_0_0 | 27 | 12200 | HIST_0 | /dev/vgtpcc_raw/r10i_lvm_hist_0_0 |
| hist_0_1 | 28 | 12200 | HIST_0 | /dev/vgtpcc_raw/r10i_lvm_hist_0_1 |
| hist_0_2 | 29 | 12200 | HIST_0 | /dev/vgtpcc_raw/r10i_lvm_hist_0_2 |
| icust1_0_0 | 82 | 9880 | ICUST1_0 | /dev/vgtpcc_raw/r10i_lvm_icust1_0_0 |
| icust2_0_0 | 83 | 9880 | ICUST2_0 | /dev/rdsk/c57t0d7 |
| icust2_0_1 | 84 | 9880 | ICUST2_0 | /dev/rdsk/c49t0d7 |
| idist_0_0 | 85 | 54 | IDIST_0 | /dev/vgtpcc_raw/r10i_lvm_idist_0_0 |
| iordr2_0_0 | 88 | 9880 | IORDR2_0 | /dev/vgtpcc_raw/r10i_lvm_iordr2_0_0 |
| iordr2_0_1 | 89 | 9880 | IORDR2_0 | /dev/vgtpcc_raw/r10i_lvm_iordr2_0_1 |
| istok_0_0 | 86 | 14000 | ISTOK_0 | /dev/vgtpcc_raw/r10i_lvm_istok_0_0 |
| istok_0_1 | 87 | 14000 | ISTOK_0 | /dev/vgtpcc_raw/r10i_lvm_istok_0_1 |
| item_0_0 | 59 | 60 | ITEM_0 | /dev/vgtpcc_raw/r10i_lvm_item_0_0 |
| iware_0_0 | 81 | 14 | IWARE_0 | /dev/vgtpcc_raw/r10i_lvm_iware_0_0 |
| log_3 | | 21610 | | /dev/rdsk/c62t0d1 |
| log_4 | | 21610 | | /dev/rdsk/c62t0d2 |
| nord_0_0 | 80 | 3490 | NORD_0 | /dev/vgtpcc_raw/r10i_lvm_nord_0_0 |
| ordr_0_0 | 60 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_0 |
| ordr_0_1 | 61 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_1 |
| ordr_0_10 | 70 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_10 |
| ordr_0_11 | 71 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_11 |
| ordr_0_12 | 72 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_12 |
| ordr_0_13 | 73 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_13 |
| ordr_0_14 | 74 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_14 |
| ordr_0_15 | 75 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_15 |
| ordr_0_16 | 76 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_16 |
| ordr_0_17 | 77 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_17 |
| ordr_0_18 | 78 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_18 |
| ordr_0_19 | 79 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_19 |
| ordr_0_2 | 62 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_2 |
| ordr_0_20 | 91 | 15901 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_20 |
| ordr_0_3 | 63 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_3 |
| ordr_0_4 | 64 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_4 |
| ordr_0_5 | 65 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_5 |
| ordr_0_6 | 66 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_6 |
| ordr_0_7 | 67 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_7 |
| ordr_0_8 | 68 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_8 |
| ordr_0_9 | 69 | 15900 | ORDR_0 | /dev/vgtpcc_raw/r10i_lvm_ordr_0_9 |
| roll01 | 3 | 15900 | UNDO_TS | /dev/vgtpcc_raw/r10i_lvm_roll01 |
| sp_0 | 90 | 2000 | SP_0 | /dev/vgtpcc_raw/r10i_lvm_sp_0 |
| stok_0_0 | 30 | 15900 | STOK_0 | /dev/rdsk/c56t0d5 |
| stok_0_1 | 31 | 15900 | STOK_0 | /dev/rdsk/c56t0d3 |
| stok_0_10 | 40 | 15900 | STOK_0 | /dev/rdsk/c49t0d5 |
| stok_0_11 | 41 | 15900 | STOK_0 | /dev/rdsk/c53t0d6 |

| | | | | |
|---|---|---|---|---|
| stok_0_12 | 42 | 15900 | STOK_0 | /dev/rdsk/c45t0d5 |
| stok_0_13 | 43 | 15900 | STOK_0 | /dev/rdsk/c44t0d2 |
| stok_0_14 | 44 | 15900 | STOK_0 | /dev/rdsk/c52t0d2 |
| stok_0_15 | 45 | 15900 | STOK_0 | /dev/rdsk/c48t0d3 |
| stok_0_16 | 46 | 15900 | STOK_0 | /dev/rdsk/c45t0d2 |
| stok_0_17 | 47 | 15900 | STOK_0 | /dev/rdsk/c44t0d4 |
| stok_0_18 | 48 | 15900 | STOK_0 | /dev/rdsk/c53t0d3 |
| stok_0_19 | 49 | 15900 | STOK_0 | /dev/rdsk/c53t0d5 |
| stok_0_2 | 32 | 15900 | STOK_0 | /dev/rdsk/c57t0d4 |
| stok_0_20 | 50 | 15900 | STOK_0 | /dev/rdsk/c53t0d4 |
| stok_0_21 | 51 | 15900 | STOK_0 | /dev/rdsk/c53t0d2 |
| stok_0_22 | 52 | 15900 | STOK_0 | /dev/rdsk/c52t0d5 |
| stok_0_23 | 53 | 15900 | STOK_0 | /dev/rdsk/c44t0d3 |
| stok_0_24 | 54 | 15900 | STOK_0 | /dev/rdsk/c45t0d4 |
| stok_0_25 | 55 | 15900 | STOK_0 | /dev/rdsk/c49t0d4 |
| stok_0_26 | 56 | 15900 | STOK_0 | /dev/rdsk/c56t0d4 |
| stok_0_27 | 57 | 15900 | STOK_0 | /dev/rdsk/c57t0d3 |
| stok_0_28 | 58 | 15900 | STOK_0 | /dev/rdsk/c49t0d2 |
| stok_0_3 | 33 | 15900 | STOK_0 | /dev/rdsk/c44t0d5 |
| stok_0_4 | 34 | 15900 | STOK_0 | /dev/rdsk/c52t0d3 |
| stok_0_5 | 35 | 15900 | STOK_0 | /dev/rdsk/c44t0d6 |
| stok_0_6 | 36 | 15900 | STOK_0 | /dev/rdsk/c57t0d5 |
| stok_0_7 | 37 | 15900 | STOK_0 | /dev/rdsk/c48t0d5 |
| stok_0_8 | 38 | 15900 | STOK_0 | /dev/rdsk/c49t0d6 |
| stok_0_9 | 39 | 15900 | STOK_0 | /dev/rdsk/c56t0d2 |
| system_001 | 1 | 400 | SYSTEM | /dev/vgtpcc_raw/r10i_lvm_system_001 |
| ware_0_ 0 | 4 | 54 | WARE_0 | /dev/vgtpcc_raw/r10i_lvm_ware_0_0 |

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

## 5.3 Data Model & Interfaces

*A statement must be provided that describes:*

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

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

Oracle Database 10G Standard Edition is a relational DBMS. SQL stored procedures were used, invoked through the Oracle Call Interface (OCI); the application code appears in Appendix A.

## 5.4 Partitions/Replications

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

No partitioning or replication was used.

## 5.5 Growth Requirements

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

# 6 Clause 5 Related Items

## 6.1 Throughput

*Measured tpmC must be reported.*

**Table 6.1: Measured tpmC**

| | |
|---|---|
| tpmC® | 131,639.80 |

## 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 | 0.30s | 0.55s | 12.45s |
| Payment | 0.29s | 0.54s | 12.07s |
| Order-Status | 0.31s | 0.56s | 9.99s |
| Delivery (interactive portion) | 0.12s | 0.21s | 9.30s |
| Delivery (deferred portion) | 0.30s | 0.55s | 9.93s |
| Stock-Level | 0.31s | 0.56s | 10.48s |
| Menu | 0.000039s | 0.1000s | 0.00271s |

## 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 | 18.31s |
| Payment | 3.01s | 3.02s | 3.31s |
| Order Status | 2.01s | 2.02s | 2.31s |
| Interactive Delivery | 2.01s | 2.02s | 2.31s |
| Stock Level | 2.01s | 2.02s | 2.31s |

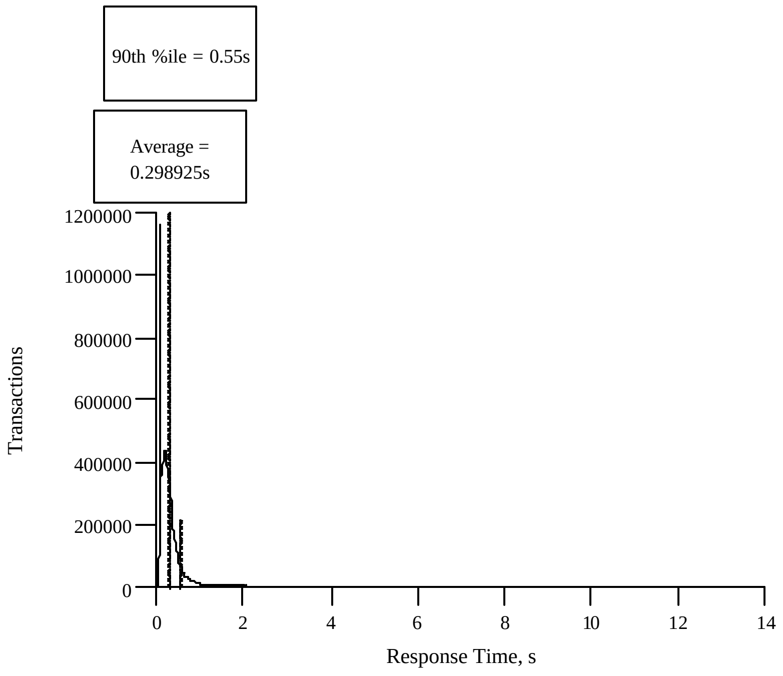
**Table 6.4: Think Times**

| Think Times | Minimum | Average | Maximum |
|---|---|---|---|
| New Order | 0.01s | 12.14s | 189.93s |
| Payment | 0.01s | 12.06s | 205.69s |
| Order Status | 0.01s | 10.1s | 144.01s |
| Interactive Delivery | 0.01s | 5.06s | 82.9s |
| Stock Level | 0.01s | 5.07s | 75.11s |

## 6.4 Response Time Frequency Distribution Curves and Other Graphs

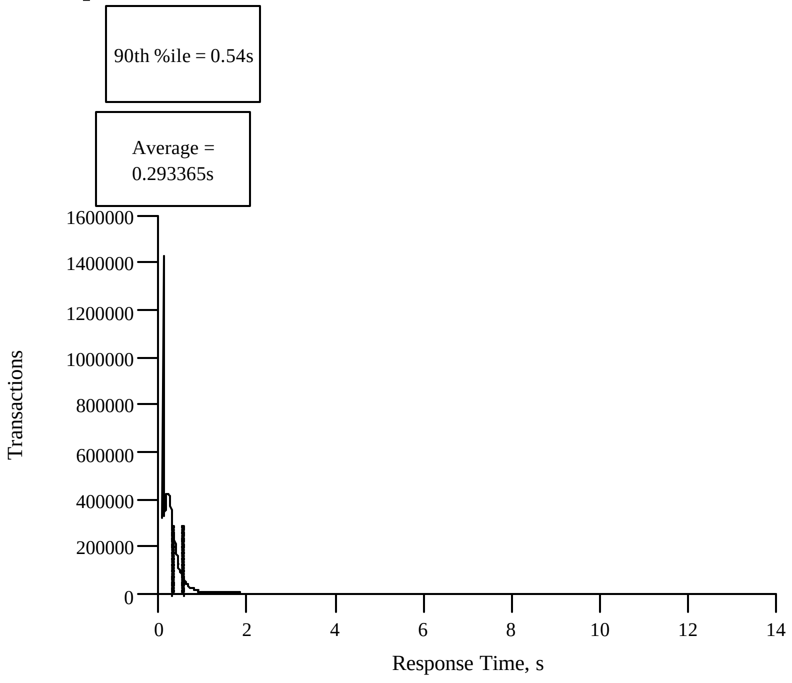
*Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type. The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction. The Think Time frequency distribution curve (see Clause 5.6.3) must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

Figure 6.1: New Order Response Time Distribution



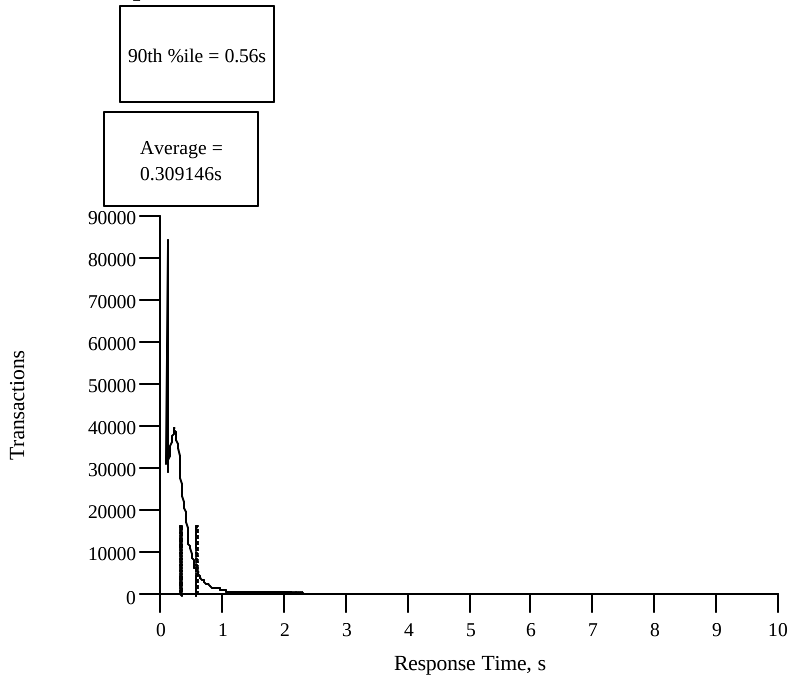
Response time frequency distribution for New Order transaction

Figure 6.2: Payment Response Time Distribution



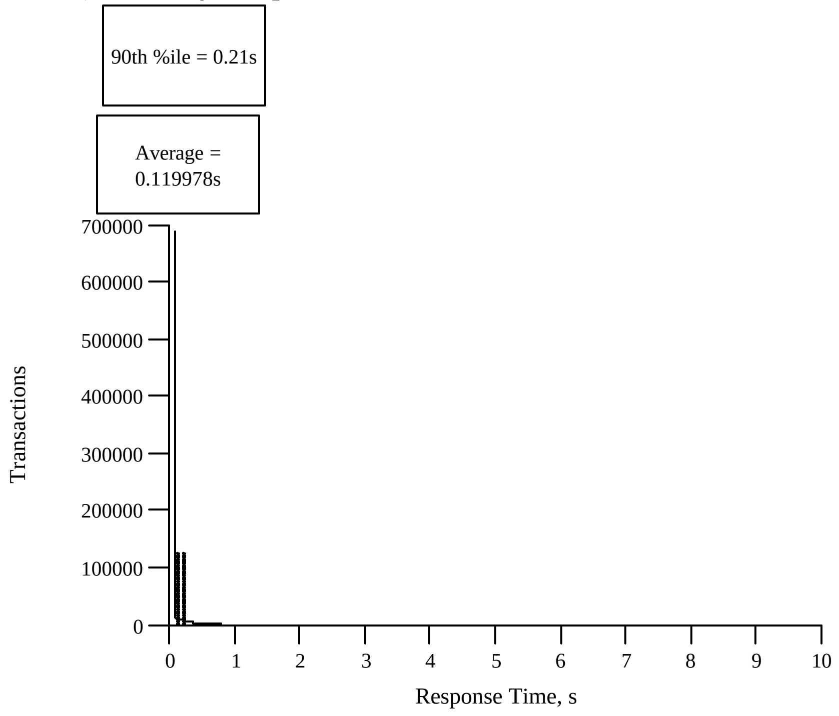
Response time frequency distribution for Payment transaction

## Figure 6.3: Order Status Response Time Distribution



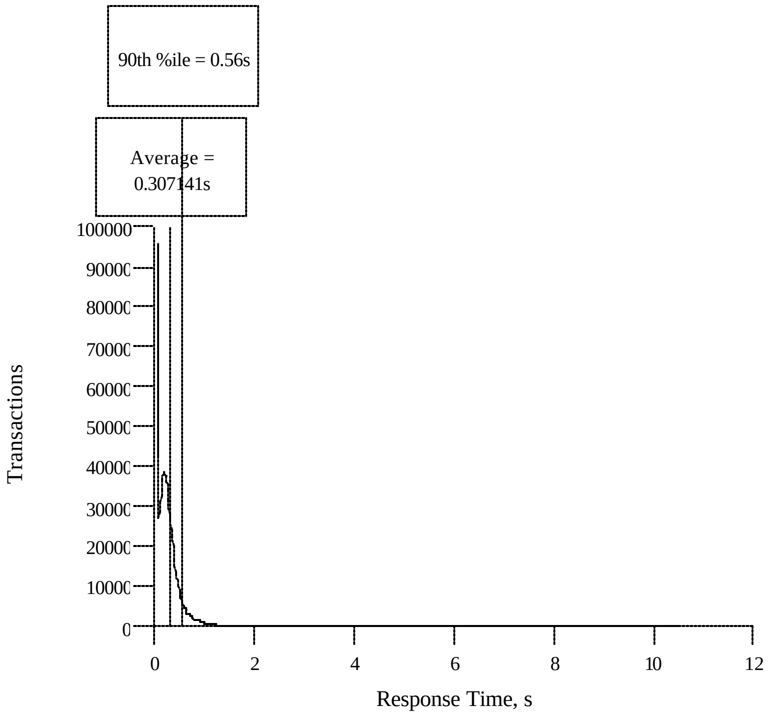
Response time frequency distribution for Order Status transaction

## Figure 6.4: (Interactive) Delivery Response Time Distribution



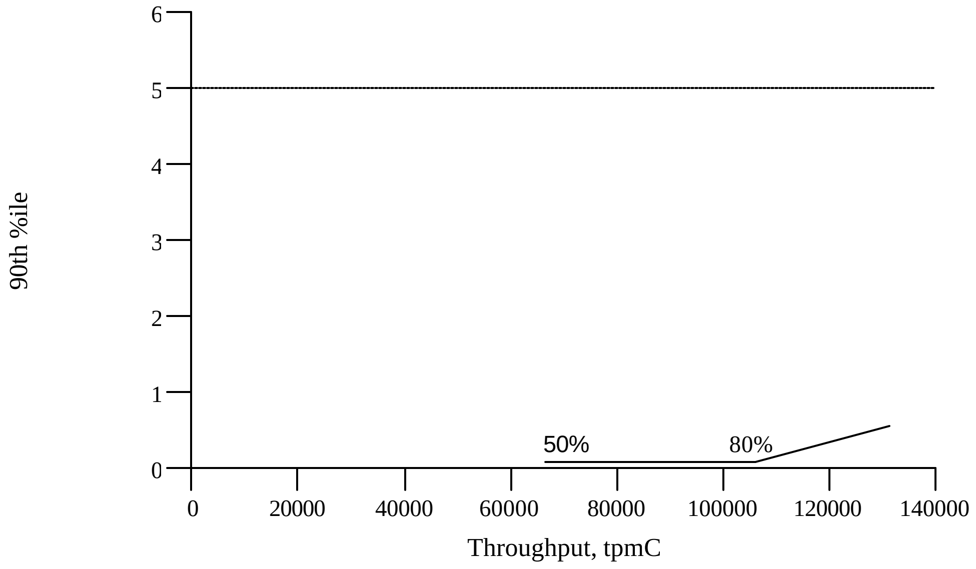
Response time frequency distribution for Delivery transaction

Figure 6.5: Stock Level Response Time Distribution



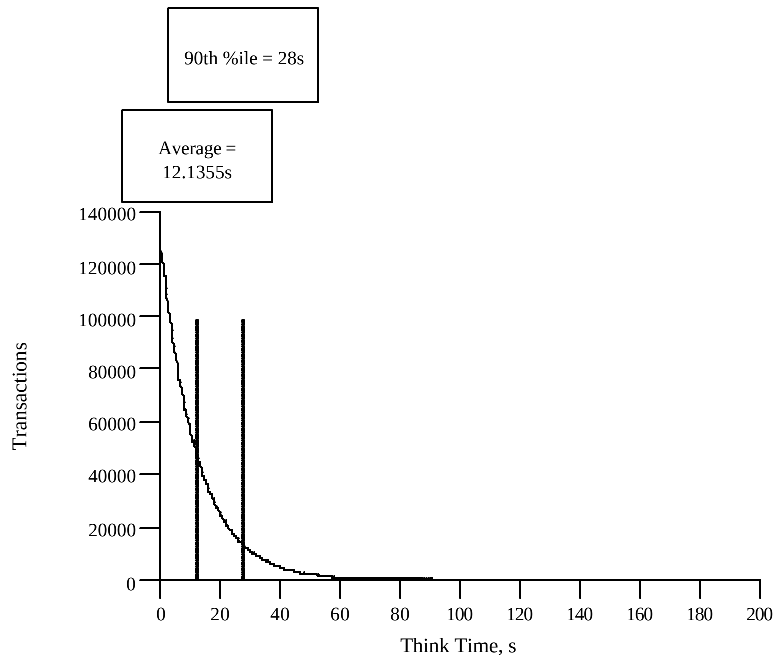
Response time frequency distribution for Stock Level transaction

Figure 6.6: Response Time Versus Throughput



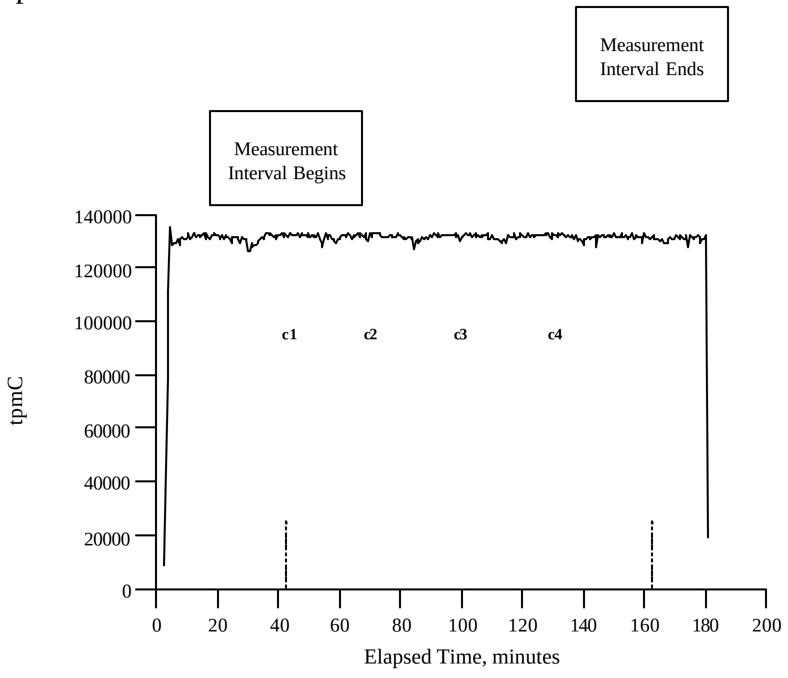
New Order response time versus Throughput

## Figure 6.7: New Order Think Time Distribution



Think time frequency distribution for New Order transaction

## Figure 6.8: Throughput Versus 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 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 Standard Edition checkpoint, all modified blocks in the shared buffer cache which had not been written to disk since the last checkpoint are written to disk.

### 6.6.2 Checkpoint Conditions

Oracle Database 10G Standard Edition performs a checkpoint for the following conditions:

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

### 6.6.3 Checkpoint Implementation

The first method listed above, i.e., a log switch when the redo log file filled up, was used to cause checkpoints. After the initial checkpoint, a log switch was performed every 29.7 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 | 44.97% |
| Payment | 43.00% |
| Order Status | 4.01% |
| Delivery | 4.020% |
| Stock Level | 4.01% |

## 6.10 Transaction Statistics

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

See Table 3.1

## 6.11 Checkpoint Count and Location

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

A checkpoint was completed before the measurement interval began. There were four checkpoints completed within the measurement interval. The first checkpoint starts approximately 1.4 minutes into the measurement interval.
The average checkpoint interval is 29.7 minutes and a checkpoint lasts approximately 26.9 minutes

The run started at 02:22:00. The Measurement Interval was 03:04:30 to 05:04:30.

The checkpoints during this run were:

```
                Start           End
Checkpoint      time            time            Duration
----------      --------        --------        --------
                02:22:00                        run starts
#0              02:36:46        03:03:02        26:16
                03:04:30                        measurement starts
#1              03:05:51        03:32:44        26:53
#2              03:35:35        04:02:24        26:49
#3              04:05:14        04:32:03        26:49
#4              04:34:54        05:01:54        27:00
                                05:04:30        measurement ends
```

# 7 Clause 6 Related Items

## 7.1 RTE Description

*If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of what inputs (e.g., scripts) to the RTE had been used. The RTE input parameters, code fragments, functions, et cetera used to generate each transaction input field must be disclosed. Comment: The intent is to demonstrate the RTE was configured to generate transaction input data as specified in Clause 2.*

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

For this instance of the TPC-C benchmark, 9 drivers and 9 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.

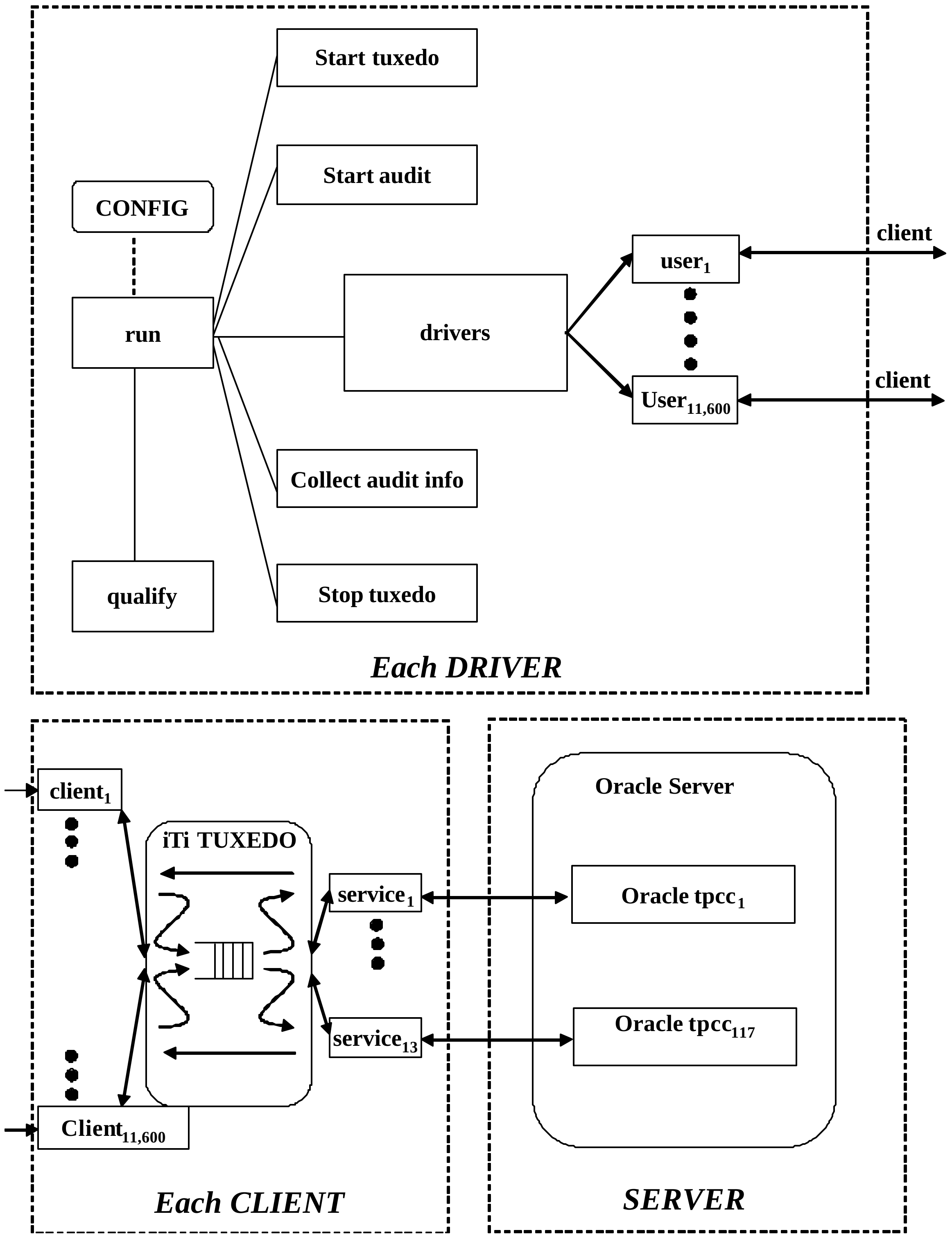


**Figure 7.1: Benchmark Software**

## 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 104,400 simulated workstations connected to the clients over 9 100BT lans through a single hp procurve switch In the priced configuration, the 104,400 worksations would connect to the clients via a combination of hubs and switches which eventually mutipexed down to 9 100BT lans.

## 7.4 Functional Diagrams

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

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

## 7.5 Networks

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

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

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

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

## 7.6 Client Substitution

No client substitution was used.

# 8 Clause 7 Related Items

## 8.1 System Pricing

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

*The total 3-year price of the entire configuration must be reported including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.*

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

## 8.2 Support Pricing

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

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

### 8.2.1 HP Hardware Support

HP's on-site support for hardware provides service 24 hour, seven day support, 4-hour response time.

### 8.2.2 HP Software Support

HP Software Support provides the following:

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

## 8.3 Oracle Corporation Standard Technical Support

Oracle Corporation Standard Technical Support includes:

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

## 8.4 Availability

*The committed delivery date for general availability (availability date) of products used in the price calculation must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available.*

see below

## 8.5 Priced System Configuration

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

## 8.6 Throughput, Price/Performance, and Availability Date

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

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

All hardware components in this test of the hp server rx5670 system is available now. HP-UX 11.i, v2, 64-bit Base incorporating is available Now. Oracle Database 10G Standard Edition will be available January 30, 2004.

# 9 Clause 9 Related Items

## 9.1 Auditor's Report

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

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

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

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

The attestation letter is shown on the following pages.

**Jun 28, 2003**

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

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

| | |
|---|---|
| **Platform:** | **hp server rx5670** |
| **Database Manager:** | **Oracle10i Database Standard Edition** |
| **Operating System:** | **HP-UX 11i v2 64-bit Base OS** |
| **Transaction Monitor:** | **BEA TUXEDO 8** |

| **Server: hp server rx5670** | | | | |
|---|---|---|---|---|
| **CPU's** | **Memory** | **Disks (total)** | **90% Response** | **TpmC** |
| **4 Intel Itanium2 @ 1.5 GHz** | **Main: 96 GB Cache: 6MB (level 3)** | **4 @ 18GB 240 @ 36GB 15 @ 73GB** | **0.55** | **131,639.80** |
| **9 Clients: hp server rp2470** | | | | |
| **1 PA-RISC 8700 @ 750 MHz** | **Main: 3 MB I-cache: 750 KB D-cache: 1.5MB** | **1 @ 36GB** | **Na** | **Na** |

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

**The following attributes of the benchmark were given special attention:**

- **The transactions were correctly implemented.**
- **The database files were properly sized and populated – see notes.**
- **The database was properly scaled with 11,000 warehouses, 10,440 of which were used. I verified that d_next_o_id and w_ytd had initial values.**
- **The ACID properties were successfully demonstrated.**
- **Data loss durability was demonstrated on the SUT with 104,400 active users.**

- **Input data was generated according to the specified percentages.**
- **Eight hours of mirrored log space was present on the tested system.**
- **Eight hours of growth space for the dynamic tables was present on the tested system.**
- **The data for the 60-day space calculation was verified – see notes.**
- **The CNUM for load was "1"; the CNUM for the RTE was "86".**
- **The steady state portion of the test was 120 minutes.**
- **One checkpoint was taken before the measured interval.**
- **Four checkpoints were taken during the measured interval.**
- **Pricing for maintenance and component counts was verified.**

**Auditor Notes:**

**Data compression was used on the city and street columns in the Customer table, and on the s_data and s_dist columns in the Stock table. As these columns are loaded with random a_strings, I believe this does not conflict with Clause 4.3.3.2.**

The measured system had 4 internal 18GB disks for "root" and the Oracle DBMS. SAR data showed little or no usage for these functions, therefore, they were allocated space on one 73GB disk. Two 73GB disks were substituted for the other 3 internal disks to meet the 60-day space requirement.

There were extra disk arrays attached to the SUT; I verified that they had no activity during the measurement.

**Sincerely,**

Lorna Livingtree

**Lorna Livingtree**
**Auditor**

# 10 Report Availability

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

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

or your local Hewlett-Packard sales office.

# Appendix A Client/Server Source

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

## A.1 Client Front-End

### client/client.c

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 2003/06/18 15:02:53 $

 (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/in.h>
#include <netdb.h>
#include <fcntl.h>
#include <errno.h>
#include <pthread.h>

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

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


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


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

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

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

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


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

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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

   /* done */
   cleanup();

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

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


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

Neworder form processing

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


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


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

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

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

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

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

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

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

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

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

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

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

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

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

```
                                                all_local = 0;
                                    }

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

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

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

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

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

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Payment form processing

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



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


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

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

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

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

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

    out_buf = old;
}


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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


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


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


static void
payment_write(payment_trans *t)
{

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

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

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

    trigger2();
}




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

ORDSTAT form processing

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




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

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

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

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

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

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

    /* done */
    out_buf = old;
}




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

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

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

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

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

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

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

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

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

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

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

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

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


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


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

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

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

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

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




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

delivery form processing

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




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


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

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

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

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

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

    /* done */
    out_buf = old;
}

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

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

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

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

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

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

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



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

stocklev form processing

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


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


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

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

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

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

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

    /* done */
    out_buf = old;
}



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

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

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

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


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

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

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

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


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

login form processing

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

static int
login(void)
{
    int field;
    int key;
   char auditstr[21];
    int w_id, d_id;
    /* assume the default values */
    w_id = warehouse;
    d_id = district;
    auditstr[0] = ' \0';

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

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

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

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

    if (key != ENTER)
        return EOF;

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

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

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

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




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

menu form processing

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

static void
menu_setup(void)
{

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

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

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

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

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

static int
setup(int fd)
{
    int key;

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

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

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

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

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

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

    return key;
}

static void
cleanup(void)
{

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

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


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

Screen Output Routines

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



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



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

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

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

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


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

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

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

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

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


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

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


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

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

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

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


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

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

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

ASCII terminal control

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


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

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

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

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

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

Screen Input Routines

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

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


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

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

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

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

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

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


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

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

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

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

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

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

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

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

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

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

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

   return key;
}
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

        default     : valid = NO;
                                   break;
        }

    return valid;
}

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

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

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

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

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

   return fd;
}

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

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

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

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

   return fd;
}

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

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

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


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

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

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

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

   GetArgs(argc, argv);

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

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

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

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

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

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

   return 0;
}

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

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

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


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

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

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

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

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

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

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

    return popc(in_buf);
}
```

## client/tux_transaction.c

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

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

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

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

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

__thread void *data_ptr;

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

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

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

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

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

}


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

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

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

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

void
thread_transaction_begin(TPCONTEXT_T ctx)
{
    unsigned long alloc_size;

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

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

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

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

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

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


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

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


void
delivery_init(int u)
{
}

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

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

void
delivery_done(void)
{
}
```

## client/Makefile

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

include ../buildenv.mk

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

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

all: tpc_lib.a  server_default.o

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

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

clobber: clean

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

## A.2 Tpc_lib Source

## lib/tpcc.h

```
/********************************************************************************
 @(#) Version: A.10.10 $Date: 2002/12/10 14:38:15 $

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

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

********************************************************************************/
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED
#include <values.h>


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

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

#include <stdio.h>

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

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

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


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

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

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

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

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

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



typedef struct {
    int status;
    LOGICAL byname;
    ID W_ID;
    ID D_ID;
    ID C_ID;
```

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


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


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

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


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


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

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

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

typedef struct
   {
   struct timeval start_time;
   } success_header_t;


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

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


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

typedef struct
```

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

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

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

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

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

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


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

/* Status conditions */
#define OK 0
#define E 1
#define E_INVALID_ITEM 2
#define E_NOT_ENOUGH_ORDERS 3
#define E_DB_ERROR 4
#define E_INVALID_INPUT 5
#define E_DB_IRRECERR 6

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


#define YES 1
#define NO 0



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

#define TPC_MSG_QUE  150


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

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

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

#endif /* TPCC_INCLUDED */
```

## lib/key_chars.h

```
#ifndef __TPCC_KEY_CHARS__
#define __TPCC_KEY_CHARS__

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

## lib/errlog.c

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

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

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

static msg_buf();

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

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

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

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

   /* done */
   va_end(argptr);

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


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

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

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

   /* done */
   va_end(argptr);

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

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



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

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

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

   /* done */
   va_end(argptr);
   }




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

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

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

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



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

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

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

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



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

## lib/fmt.c

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

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

/* formatting routines.  */

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

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

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

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

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


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

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

    /* remove leading $ */
    if (*temp == '$')   t = temp + 1;
    else               t = temp;
   /* start scan at current character */
   s = t;

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

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

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

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

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


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

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

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

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



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



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

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

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

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

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

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

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

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



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

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

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

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

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

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

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

      else
         return INVALID_FLT;
      }

   if (fract)
      value /= div;

   if (negative)
      value = -value;

   return value;
   }



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

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

   /* otherwise, blank fill it */
   else
      {

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

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

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



cvt_text(s, text)
   char *s;
```

```
    char *text;
    {
    char *lastnb;

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

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

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



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

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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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



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

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


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

## lib/iobuf.h

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

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

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

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


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

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

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

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


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

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

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


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

## lib/iobuf.c

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

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



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


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


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


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

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

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

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

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



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

    return popc();
    }
```

## lib/random.c

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

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

double drand48();

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

static long RandySeedIter = 7;

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

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

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

   length = RandomNumber(min, max);

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

   return length;
   }

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

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

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


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

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

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

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

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

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

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

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

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

   length = RandomNumber(min, max);

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

   return length;
   }

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

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

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


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

   secs = exponential(mean);

   delay(secs+adjust);
   }

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

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

   return x;
   }

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

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

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

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

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

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

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

## lib/Makefile

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

include ../buildenv.mk

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

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

all: tpc_lib.a  server_default.o

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

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

clobber: clean

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

## A.3 Transaction Source

## client/service.c

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

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

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

extern int userid;
char *cmd          = NULL;

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

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

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

   return 0;
}


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


void PMT_SVC(svcinfo)
TPSVCINFO *svcinfo;
{

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

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

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

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

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

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

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

## client/oracle/transaction.c

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

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

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

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

int     numtrans = 0;

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

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

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

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

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

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

   str->O_ID = ora_str.newout.o_id;
   str->O_OL_CNT = ora_str.newout.o_ol_cnt;
   strncpy (str->C_LAST, ora_str.newout.c_last, 17);
   strncpy (str->C_CREDIT, ora_str.newout.c_credit, 3);
   str->C_DISCOUNT = (REAL) ora_str.newout.c_discount;
   str->W_TAX = (REAL) ora_str.newout.w_tax;
   str->D_TAX = (REAL) ora_str.newout.d_tax;
   strncpy (str->O_ENTRY_D, ora_str.newout.o_entry_d, 20);
   for (i = 0; i < ora_str.newout.o_ol_cnt; i++) {
      strncpy (str->item[i].I_NAME, ora_str.newout.i_name[i], 25);
      str->item[i].S_QUANTITY = ora_str.newout.s_quantity[i];
      str->item[i].brand_generic = ora_str.newout.brand_generic[i];
      str->item[i].I_PRICE = (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(str)
payment_trans *str;
{
   int i;

   struct paystruct ora_str;

   ora_str.payin.w_id = str->W_ID;
   ora_str.payin.d_id = str->D_ID;
   ora_str.payin.c_w_id = str->C_W_ID;
   ora_str.payin.c_d_id = str->C_D_ID;
   ora_str.payin.h_amount = (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(str)
ordstat_trans *str;
{
   int i;

   struct ordstruct ora_str;

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

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

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

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

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

   struct delstruct ora_str;

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

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

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

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

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

void stocklev_transaction(str)
stocklev_trans *str;
{

   struct stostruct ora_str;
   ora_str.stoin.w_id = str->W_ID;
   ora_str.stoin.d_id = str->D_ID;
   ora_str.stoin.threshold = str->threshold;
   retries = 0;

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

## client/oracle/tpccpl.c

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


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

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

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

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

int execstatus;
int errcode;

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


/* for stock-level transaction */

int w_id;
int d_id;
int c_id;
float 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];
float ol_quantity[15];
float 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;
float 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];
float nol_quantity[15];
int nol_quanti10[15];
int nol_quanti91[15];
int nol_ytdqty[15];
float nol_amount[15];
int o_all_local;
float w_tax;
float d_tax;
float total_amount;
char i_name[15][25];
float s_quantity[15];
char brand_gen[15];
float 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 charcentury;
 unsigned charyear;
 unsigned charmonth;
 unsigned charday;
 unsigned charhour;
 unsigned charminute;
 unsigned charsecond;
 } Date;
```

```
int century;
int cnvrtOK;

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

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

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

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

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


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

 return;

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

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


        int day,month,year;

        memcpy(&Date,oradt,7);

        year = (Date.century -100)*100 + Date.year-100;
        month = Date.month;
        day = Date.day;
        sprintf(outdate,"%02d-%02d-%4d\0",day,month,year);

        return;
}

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

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


        int day,month,year;
        int hour,min,sec;

        memcpy(&Date,oradt,7);

        year = (Date.century -100)*100 + Date.year-100;
        month = Date.month;
        day = Date.day;
        hour = Date.hour - 1;
        min = Date.minute - 1;
        sec = Date.second - 1;

        sprintf(outdate,"%02d-%02d-%4d %02d:%02d:%02d\0",
                 day,month,year,hour,min,sec);

        return;
}
#endif


void TPCexit (void)

{

   if (new_init) {
      tkvcndone();
      new_init = 0;
   }
   if (pay_init) {
      tkvcpdone();
      pay_init = 0;
   }
   if (ord_init) {
      tkvcodone();
      ord_init = 0;
   }
   if (del_init_oci) {
      tkvcddone(0);
      del_init_oci = 0;
   }
   if (del_init_plsql) {
      tkvcddone(1);
      del_init_plsql = 0;
   }
   if (sto_init) {
      tkvcsdone();
      sto_init = 0;
   }


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

   if (lfp) {
      fclose (lfp);
      lfp = NULL;
   }

}



TPCinit (id, uid, pwd)

int id;
char *uid;
char *pwd;

{

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

   proc_no = id;
   sprintf (filename, "tpcc_%d.del", proc_no);
   if ((lfp = fopen (filename, "w")) == NULL) {
#ifdef TUX
      userlog ("Error in TPC-C server %d: Failed to open %s\n",
            proc_no, filename);
#else
      fprintf (stderr, "Error in TPC-C server %d: Failed to open %s\n",
            proc_no, filename);
#endif
      return (-1);
   }

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

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

   /* run all transaction in serializable mode */
  OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
  sprintf ((char *) stmbuf, SQLTXT);
  OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT);
  OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
  OCIHandleFree(curi, OCI_HTYPE_STMT);

/*
This is done in cvdrv.c
  if (tracelevel == 2) {
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
    memset(stmbuf,0,100);
    sprintf ((char *) stmbuf, SQLTXTTRC);
    OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                                              OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
    OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
  }
*/
  if (tracelevel == 3) {
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
    memset(stmbuf,0,100);
    sprintf ((char *) stmbuf, SQLTXTTIM);
    OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                                              OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
    OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
  }

  logon = 1;

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

  if (tkvcninit ()) {  /* new order */
    TPCexit ();
    return (-1);
  }
  else
    new_init = 1;

  if (tkvcpinit ()) {   /* payment */
    TPCexit ();
    return (-1);
  }
  else
    pay_init = 1;

  if (tkvcoinit ()) {   /* order status */
    TPCexit ();
    return (-1);
  }
  else
    ord_init = 1;


  if (tkvcdinit (0)) {    /* delivery */
    TPCexit ();
    return (-1);
  }
  else
    del_init_oci = 1;

  if (tkvcdinit (1)) {    /* delivery */
    TPCexit ();
    return (-1);
  }
  else
    del_init_plsql = 1;

  if (tkvcsinit ()) {   /* stock level */
    TPCexit ();
    return (-1);
  }
  else
    sto_init = 1;

  return (0);

}



TPCnew (str)

struct newstruct *str;

{

  int i;

  w_id = str->newin.w_id;
  d_id = str->newin.d_id;
  c_id = str->newin.c_id;
  for (i = 0; i < 15; i++) {
    nol_i_id[i] = str->newin.ol_i_id[i];
    nol_supply_w_id[i] = str->newin.ol_supply_w_id[i];
    nol_quantity[i] = str->newin.ol_quantity[i];
  }
  retries = 0;

#ifndef AVOID_DEADLOCK

  for (i = NITEMS; i > 0; i--) {
    if (nol_i_id[i-1] > 0) {
      ordl_cnt = i;
      break;
    }
  }

  for (i = 0; i < NITEMS; i++) indx[i] = i;

  q_sort(nol_i_id, str, 0, ordl_cnt-1);

#endif

/*
  vgetdate(cr_date); */

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

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

  /* fill in date for o_entry_d from time in beginning of txn*/
/*
  cvtdmyhms(cr_date,o_entry_d);
*/
   datelen = sizeof(o_entry_d);
   OCIERROR(errhp,
       OCIDateToText(errhp,&cr_date,(text*)FULLDATE,SIZ(FULLDATE),(text*)0,0,
                     &datelen,o_entry_d));

  str->newout.terror = NOERR;
  str->newout.o_id = o_id;
  str->newout.o_ol_cnt = o_ol_cnt;
  strncpy (str->newout.c_last, c_last, 17);
  strncpy (str->newout.c_credit, c_credit, 3);
  str->newout.c_discount = c_discount;
  str->newout.w_tax = (float)(w_tax);
  str->newout.d_tax = (float)(d_tax);
  strncpy (str->newout.o_entry_d, (char*)o_entry_d, 20);
  str->newout.total_amount = total_amount;
  for (i = 0; i < o_ol_cnt; i++) {
    strncpy (str->newout.i_name[i], i_name[i], 25);
    str->newout.s_quantity[i] = (int) s_quantity[i];
    str->newout.brand_generic[i] = brand_generic[i][0];
    str->newout.i_price[i] = i_price[i]/100;
    str->newout.ol_amount[i] = 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] = (int) ol_quantity[i];
     str->ordout.ol_amount[i] = ol_amount[i]/100;
     strncpy (str->ordout.ol_delivery_d[i], (char*)ol_delivery_d[i], 11);
   }
   str->ordout.retry = retries;
#if defined(TOP)  || defined(TUX)
   return(1);
#else
   return (0);
#endif

}



TPCdel (str)

struct delstruct *str;

{

   double tr_end;
   int i;

   w_id = str->delin.w_id;
   o_carrier_id = str->delin.o_carrier_id;
   retries = 0;
/*
   vgetdate(cr_date);  */
   OCIERROR(errhp,OCIDateSysDate(errhp,&cr_date));

   if (str->delout.terror = tkvcd (str->delin.plsqlflag)) {
     if(str->delout.terror == DEL_ERROR)
       return DEL_ERROR;
     if (str->delout.terror != RECOVERR)
        str->delout.terror = IRRECERR;
     return (-1);
   }

   /* Comment out for the HP kit.
   tr_end = gettime ();
   fprintf (lfp, "%d %d %f %f %d %d", str->delin.in_timing_int,
          (tr_end - str->delin.qtime) <= DELRT ? 1 : 0,
          str->delin.qtime, tr_end, w_id, o_carrier_id);
   for (i = 0; i < 10; i++) {
     fprintf (lfp, " %d %d", i + 1, del_o_id[i]);
     if (del_o_id[i] <= 0) {
#ifdef TUX
        userlog ("DELIVERY: no new order for w_id: %d, d_id %d\n",
              w_id, i + 1);
#else
        fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
              w_id, i + 1);
#endif
     }
   }
   fprintf (lfp, " %d\n", retries);
   */

   str->delout.terror = NOERR;
   str->delout.retry = retries;
#if defined(TOP) || defined(TUX)   /* changed mjb 17 feb */
   return(1);
#else
   return (0);
#endif

}
```

```
TPCsto (str)

struct stostruct *str;

{

   w_id = str->stoin.w_id;
   d_id = str->stoin.d_id;
   threshold = (float) 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)
{
  float temp_float;
  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_float = nol_quantity[i];
  nol_quantity[i] = nol_quantity[j];
  nol_quantity[j] = temp_float;

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

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

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

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

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

}

#endif
```

## client/oracle/plnew.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: tkvcnew.c 21-apr-98.18:32:59 rdecker Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*==================================================================+
 |        Copyright (c) 1996 , 1997, 1998 Oracle Corp, Redwood Shores, CA       |
 |                   OPEN SYSTEMS PERFORMANCE GROUP                |
 |                        All Rights Reserved                        |
 +==================================================================+
 | FILENAME
 |    plnew.c
 | DESCRIPTION
 |    OCI version (using PL/SQL stored procedure) of
 |    NEW ORDER transaction in TPC-C benchmark.
 +==================================================================*/

#ifndef ORA_TPCC
# define ORA_TPCC
# include "ora_tpcc.h"
#endif

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

#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];

  float 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(float),SQLT_BFLOAT,nctx->nol_quantity_len,
          NITEMS,&nctx->nol_q_count);
  OCIBNDPLA(nctx->curn1, nctx->i_price_bp,errhp,":i_price",i_price,SIZ(float),
          SQLT_BFLOAT, 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(float), SQLT_BFLOAT,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(float),SQLT_BFLOAT, nctx->nol_amount_len,NITEMS,&nctx->nol_am_count);
  OCIBNDPLA(nctx->curn1, nctx->s_remote_bp,errhp,":s_remote",nctx->s_remote,
          SIZ(float),SQLT_BFLOAT, 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] = (float)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 7030100.1 95/07/19 14:44:59 plai Generic<base> $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*==================================================================+
 |       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA       |
 |                  OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                        All Rights Reserved                      |
 +==================================================================+
 | FILENAME
 |    plpay.c
 | DESCRIPTION
 |    OCI version (using PL/SQL stored procedure) of
 |    PAYMENT transa ction 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(float),SQLT_BFLOAT,  &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_BDOUBLE, &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(float),SQLT_BFLOAT, &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_BDOUBLE, &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 *) stm buf, 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_BDOUBLE);
  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_BDOUBLE);
  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(float),
         SQLT_BFLOAT, NULL,octx->ol_quantity_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_amount_dp,errhp,4,ol_amount, SIZ(float),
         SQLT_BFLOAT,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(FULLDATE
),(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 7010000.3 95/02/14 12:48:03 plai Generic<base> $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*==================================================================+
 |       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
 |                  OPEN SYSTEMS PERFORMANCE GROUP                  |
 |                       All Rights Reserved                        |
 +==================================================================+
 | FILENAME
 |    plsto.c
 | DESCRIPTION
 |    OCI version of STOCK LEVEL transaction in TPC-C benchmark.
 +==================================================================*/

#include "ora_tpcc.h"

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

  return (0);

}



tkvcs ()

{

retry:
   execstatus= OCIStmtExecute(tpcsvc,sctx->curs,errhp,1,0,0,0,
                        OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
   if (execstatus != OCI_SUCCESS)
   {
     errcode=OCIERROR(errhp,execstatus);
     OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
     if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
        || (errcode == SNAPSHOT_TOO_OLD))
     {
       retries++;
       goto retry;
     } else {
       return -1;
     }
    }

  return (0);
}



void tkvcsdone ()

{
  if(sctx) free(sctx);
}
```

## client/oracle/pldel.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: pldel.c 7030100.5 96/06/24 16:26:06 plai Generic<base> $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*==================================================================+
 |       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
 |                  OPEN SYSTEMS PERFORMANCE GROUP                  |
 |                       All Rights Reserved                        |
 +==================================================================+
 | FILENAME
 |    pldel.c
 | DESCRIPTION
 |    OCI version of DELIVERY transaction in TPC-C benchmark.
 +==================================================================*/

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

/*
extern int userlog();
*/
```

```
#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];
  float 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(float),SQLT_BFLOAT, pldctx->sum s_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 (size of(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_SY NTAX, 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_rc ode);

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

#ifndef TPCC_H
#define TPCC_H

#ifndef FALSE
# define FALSE 0
#endif

#ifndef TRUE
# define TRUE 1
#endif

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

#ifndef boolean
#define boolean int
#endif
```

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

typedef struct cda_def csrdef;
typedef struct cda_def ldadef;


/* TPC-C transaction functions */

extern int TPCinit ();
extern int TPCnew ();
extern int TPCpay ();
extern int TPCord ();
extern int TPCdel ();
extern int TPCsto ();
extern void TPCexit ();
extern int TPCdumpinit ();
extern void TPCdumpnew ();
extern void TPCdumppay ();
extern void TPCdumpord ();
extern void TPCdumpdel ();
extern void TPCdumpsto ();
extern void TPCdumpexit ();
extern void userlog(char* fmtp, ...);


/* Error codes */

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

#define FULLDATE "dd-mon-yy.hh24:mi:ss"
#define SHORTDATE "dd-mm-yyyy"



#define DELRT 80.0

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

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

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

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


extern void errrpt ();
extern int ocierror(char *fname, int lineno,OCIError *errhp, sword status);
extern int sqlfile(char *fname, text *linebuf);

extern FILE *lfp;
extern FILE *fopen ();
extern int proc_no;
extern int doid[];

extern int execstatus;
extern int errcode;

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

```
/* for stock-level transaction */

extern int w_id;
extern int d_id;
extern int c_id;
extern float 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 float ol_quantity[15];
extern float 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 float 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 float nol_quantity[15];
extern int nol_quanti10[15];
extern int nol_quanti91[15];
extern int nol_ytdqty[15];
extern float 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 float 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 float 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

## 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;
```

## 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;
```

## 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: 2001/08/24 17:21:52 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*******************************************************************************/
#include <sys/time.h>
#include <errno.h>
#ifndef HPUX9
#include <time.h>
#endif
#include "tpcc.h"
#include "shm.h"

delay(sec)
/**************************************************************************
delay sleeps for the specified number of seconds.  (to closest 1/100'th second)
**************************************************************************/
    double sec;
    {
#ifdef HPUX9
    struct timeval delay;
#else
    struct timespec delay;
#endif

    /* if no delay,  done */
    if (sec <= 0.0) return;

    /* add a portion of a clock tick to keep averages correct */
    sec += 1.0 / CLK_TCK;

    /* convert the delay to seconds and nanoseconds */
    delay.tv_sec = sec;
#ifdef HPUX9
    delay.tv_usec = (sec - delay.tv_sec) * 1000000;
#else
    delay.tv_nsec = (sec - delay.tv_sec) * 1000000000;
#endif

    /* sleep on a select call */
#ifdef HPUX9
    if (select(0, NULL, NULL, NULL, &delay) < 0) {
       sysrror("delay:  select() call failed\n");
    }
#else
    if (nanosleep(&delay,NULL) == -1) {
                    if (errno != EINTR) {
          syserror("delay:  nanosleep() call failed, errno = %d\n",errno);
                    }
    }
#endif
}



struct timeval start_time;

initclock()
   {
   gettimeofday(&start_time, NULL);
   }


TIME getclock()
/*********************************************************************
getclock returns the current time, expressed in seconds from start of run
*********************************************************************/
   {
   struct timeval current;
   gettimeofday(&current, NULL);

   return elapsed_time(&current);
   }
```

## random.h

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

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

#ifdef USE_DRAND48
double drand48();
```

```
#else
double randy();
#endif

extern int    MakeNumberString();
extern ID     RandomWarehouse();
extern int    MakeAlphaString();
extern void   RandomPermutation();
extern void   RandomDelay();
extern double exponential();
extern void   Randomize();
extern void   SetRandomSeed();

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

/**************************************************************************
RandomNumber selects a uniform random number from min to max inclusive
**************************************************************************/
#ifdef USE_DRAND48
#define RandomNumber(min,max)  \
                    ((int)(drand48() * ((int)(max) - (int)(min) + 1)) + (int)(min))
#else
#define RandomNumber(min,max)  \
                    ((int)(randy() * ((int)(max) - (int)(min) + 1)) + (int)(min))
#endif

/**************************************************************************
NURandomNumber selects a non-uniform random number
**************************************************************************/
#define NURandomNumber(a, min, max, c) \
   ((RandomNumber(0, a) | RandomNumber(min, max)) + (c)) % \
                                        ((max) - (min) + 1) + (min)


#define SelectCityStreetData(data) \
{ \
    strcpy(data,citystreetData1[RandomNumber(0,9)]); \
    strcat(data,citystreetData2[RandomNumber(0,9)]); \
}

#define SelectFirstName(data) \
{ \
    strcpy(data,firstNameData1[RandomNumber(0,9)]); \
    strcat(data,firstNameData2[RandomNumber(0,9)]); \
}

#define SelectHistoryData(data) \
{ \
    strcpy(data,historyData1[RandomNumber(0,9)]); \
    strcat(data,historyData2[RandomNumber(0,9)]); \
}

#define SelectStockData(data) \
{ \
    strcpy(data,stockData1[RandomNumber(0,9)]); \
    strcat(data,stockData2[RandomNumber(0,9)]); \
}

#define SelectClientData(data) \
{ \
    strcpy(data,customerData1[RandomNumber(0,9)]); \
    strcat(data,customerData2[RandomNumber(0,9)]); \
}

#define SelectPhoneData(data) strcpy(data,phoneData[RandomNumber(0,9)])
#define SelectStockDistrict(data) strcpy(data,StockDistrict[RandomNumber(0,9)])

#define MakeZip(zip) \
{ \
   MakeNumberString(4, 4, zip); \
   zip[4] = '1'; \
   zip[5] = '1'; \
   zip[6] = '1'; \
   zip[7] = '1'; \
   zip[8] = '1'; \
   zip[9] = '\0'; \
}

#define MakeAddress(str1, str2, city, state, zip) \
{ \
    SelectCityStreetData(str1); \
    SelectCityStreetData(str2); \
    SelectCityStreetData(city); \
    MakeAlphaString(2,2,state); \
    MakeZip(zip); \
}

#define LastName(num, name) strcpy(name, lastNames[(num)])

#define Original(str) \
{ \
    int len = strlen(str); \
    if (len >= 8) { \
        int pos = RandomNumber(0,(len-8)); \
        str[pos+0] = 'O'; \
        str[pos+1] = 'R'; \
        str[pos+2] = 'I'; \
        str[pos+3] = 'G'; \
        str[pos+4] = 'I'; \
        str[pos+5] = 'N'; \
        str[pos+6] = 'A'; \
        str[pos+7] = 'L'; \
    } \
}


#endif
```

## results_file.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2002/07/18 22:07:44 $

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



static FILE *rfile;

results_open(id)
    int id;
    {
    char fullname[128];
    char *basename;

    /* get the base file name for the deferred results */
    /*
     * Make it a directory under /tmp so at least we can set it to a
     * symbolic link in case /tmp doesn't have enough room.
     */
    basename = getenv("TPCC_RESULTS_FILE");
    if (basename == NULL)
        basename = "/tmp/TPCC_RESULTS_FILE";

    /* create the full file name */
    sprintf(fullname, "%s.%d", basename, id);

    /* open the file */
    unlink(fullname);
    rfile = fopen(fullname, "wb");
    if (rfile == NULL)
        syserror("Delivery server %d can't open file %s\n", id, fullname);

    /* allocate a larger buffer */
    }



results(t)
    delivery_trans *t;
    {
    if (fwrite(t, sizeof(*t), 1, rfile) != 1)
        syserror("Delivery server: Can't post results\n");
    }


results_close()
    {
    if (fclose(rfile) < 0)
        syserror("Delivery server can't close file\n");
    }
```

# Appendix B Database Design

The source code for the process to define, create and populate the Oracle10i Database Standard Edition TPC-C database is included in this appendix.

## B.1 Scripts

### createtable_cust.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreatetable.sh Tue Apr 8
16:32:31 PDT 2003 */
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 330000000
  hash is ( (c_id * ( 11000 * 10 ) + c_w_id * 10 + c_d_id) )
  size 180
  pctfree 0  initrans 3
  storage ( buffer_pool recycle )
  tablespace cust_0;
#change size 205 to size 180

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 binary_double
, c_ytd_payment binary_double
, c_payment_cnt binary_float
, c_delivery_cnt binary_float
, c_street_1 varchar2(20)
, c_street_2 varchar2(20)
, c_city varchar2(20)
, c_state char(2)
, c_zip char(9)
, c_phone char(16)
, c_since date
, c_middle char(2)
, c_data varchar2(500)
)
cluster custcluster (
  c_id
, c_d_id
, c_w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

### createtable_hist.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreatetable.sh Tue Apr 8
16:32:41 PDT 2003 */
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 binary_float
, h_data varchar2(24)
)
  pctfree 5  initrans 4
  storage ( buffer_pool default )
  tablespace hist_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

### createtable_nord.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreatetable.sh Tue Apr 8
16:32:54 PDT 2003 */
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 110000
    hash is ( (no_w_id - 1) * 10 + no_d_id - 1 )
    size 190
    tablespace nord_0;

  create table nord (
    no_w_id number
, no_d_id number
, no_o_id number sort
    , constraint nord_uk primary key ( no_w_id
, no_d_id
, no_o_id )
  )
  cluster nordcluster_queue (
    no_w_id
, no_d_id
, no_o_id
  );
    set echo off
    spool off
    exit sql.sqlcode;
```

### createtable_ordr.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-
11k/scripts/buildcreatetable.sh Tue Apr 8 16:32:50 PDT 2003 */
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 110000
    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_ordl.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreatetable.sh Tue Apr 8
16:32:52 PDT 2003 */
set timing on
   set sqlblanklines on
   spool createtable_ordl.log
   set echo on
      create table ordl (
   ol_w_id number
, ol_d_id number
, ol_o_id number sort
, ol_number number sort
, ol_i_id number
, ol_delivery_d date
, ol_amount binary_float
, ol_supply_w_id number
, ol_quantity binary_float
, ol_dist_info char(24)
  , constraint ordl_uk primary key (ol_w_id, ol_d_id, ol_o_id, ol_number  )) CLUSTER
ordrcluster_queue(ol_w_id, ol_d_id, ol_o_id, ol_number) ;
   set echo off
   spool off
   exit sql.sqlcode;
```

## createtable_stok.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreatetable.sh Tue Apr 8
16:32:42 PDT 2003 */
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 1100000000
  hash is ( (s_i_id * 11000 + s_w_id) )
  size 295
  pctfree 0  initrans 3
  storage ( buffer_pool keep )
  tablespace stok_0;

create table stok (
  s_i_id number
, s_w_id number
, s_quantity binary_float
, s_ytd binary_double
, s_order_cnt binary_float
, s_remote_cnt binary_float
, 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_dist.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreatetable.sh Tue Apr 8
16:32:37 PDT 2003 */
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 110000
  hash is ( ((d_w_id * 10) + d_id) )

    initrans 4
  storage ( buffer_pool default )
  tablespace dist_0;

create table dist (
  d_id number
, d_w_id number
, d_ytd binary_double
, 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_item.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreatetable.sh Tue Apr 8
16:32:48 PDT 2003 */
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 binary_float
, i_data varchar2(50)
, i_im_id number
)
cluster itemcluster (
  i_id
);
   set echo off
   spool off
   exit sql.sqlcode;
```

## Createtable_ware.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreatetable.sh Tue Apr 8
16:32:27 PDT 2003 */
set timing on
   set sqlblanklines on
   spool createtable_ware.log
   set echo on
     drop cluster warecluster including tables ;

create cluster warecluster (
  w_id number(5,0)
  )
  single table
  hashkeys 11000
  hash is ( (w_id) )
  initrans 2
  storage ( buffer_pool default )
  tablespace ware_0;

create table ware (
  w_id number(5,0)
, w_ytd binary_double
, 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;
```

## Createinde x_iware.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:01 PDT 2003 */
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
  tablespace iware_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

## createindex_iitem.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:06 PDT 2003 */
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 )
  tablespace item_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

## createindex_idist.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:04 PDT 2003 */
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
  tablespace idist_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

## createindex_iordl.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:07 PDT 2003 */
set timing on
  exit 0;
```

## createindex_icust1.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:02 PDT 2003 */
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
  tablespace icust1_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

## createindex_icust2.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:03 PDT 2003 */
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
  tablespace icust2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

## createindex_inord.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:10 PDT 2003 */
set timing on
  exit 0;
```

## createindex_iordr1.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:07 PDT 2003 */
set timing on
  exit 0;
```

## createindex_iordr2.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:08 PDT 2003 */
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
  tablespace iordr2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

## createindex_istok.sql

```
/* created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/buildcreateindex.sh Tue Apr 8
16:33:05 PDT 2003 */
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
  tablespace istok_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

## 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 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
    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 *.log `find $tpcc_bench/scripts -name '*.log'` $tpcc_bench/log/
    else
        mv *.log $tpcc_bench/log/
    fi

    if expr $lasterror != 0 > /dev/null; then
      if expr $lasterror != 99 > /dev/null; then
        echo Step $step failed.  Stopping driver.
        exit 1
      else
       echo Step $step has completed and requested stop.  Stopping driver.
        exit 0
      fi
    fi
    if test "x$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
```

## 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_args=
tpcc_user_pass=
tpcc_sqlplus=sqlplus
tpcc_user_pass='tpcc/tpcc'

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

#8gb oracle filesize limit (in k)
tpcc_fsize_limit_k=8388608
#2gb - 1k oracle extent limit (in k)
tpcc_extent_limit_k=2097151

# Runlen calculations 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=200M
tpcc_logfile_size=`$tpcc_bcexpr 20 + \( $tpcc_scale \)`M

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

tpcc_undo_bs=8K

tpcc_statspack_size=`$tpcc_bcexpr 1 \* $tpcc_scale`
if test $tpcc_statspack_size -gt 2048; then
  tpcc_statspack_size=2048
fi
tpcc_statspack_size="${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 createtablespaces.sh)
#and add that much to the base tablespace size.
tpcc_hist_growth=51
tpcc_ordr_growth=35
tpcc_nord_growth=13
#tpcc_ordl_growth=660
tpcc_ordl_growth=900

#i started indices at 1/10th... need an exact figure
tpcc_iordr1_growth=20
tpcc_iordr2_growth=20
tpcc_iordl_growth=66
tpcc_inord_growth=2

tpcc_item_growth=0
tpcc_iitem_growth=0
tpcc_temp_growth=0

tpcc_cust_growth=regular
tpcc_icust1_growth=regular
tpcc_icust2_growth=regular

tpcc_stok_growth=regular
tpcc_istok_growth=regular

tpcc_ware_growth=regular
tpcc_iware_growth=regular

tpcc_dist_growth=regular
tpcc_idist_growth=regular

# minimum size of temp tablespace
tpcc_tempts_min=10240

# for Linux, set appropriate tablespace heuristics
# to set high io tables to have 64 files, and minimize
# others.
if expr $tpcc_os = linux > /dev/null; then
  for table in $tpcc_table_list $tpcc_index_list temp; do
    eval "tpcc_${table}_tsfileinc=1"
  done
  tpcc_os=unix

  tpcc_stok_tsfileinc=64
  tpcc_cust_tsfileinc=64
  tpcc_iordl2_tsfileinc=16
  tpcc_icust2_tsfileinc=16
  tpcc_iordl_tsfileinc=16
else
#in case someone changes out of linux, and the shell is stuck
  for table in $tpcc_table_list $tpcc_index_list temp; do
    eval "tpcc_${table}_tsfileinc="
  done
  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="buildcreatets buildcreatedb \
buildcreatetable-ware buildcreatetable -cust buildcreatetable-dist buildcreatetable-hist
buildcreatetable-stok buildcreatetable-item buildcreatetable-ordr buildcreatetable -ordl
buildcreatetable-nord \
buildloadware buildloaddist buildloaditem buildloadhist buildloadnord buildloadordrordl
buildloadcust buildloadstok buildfixoo \
buildcreateindex-iware buildcreateindex-icust1 buildcreateindex-icust2 buildcreateindex-idist
buildcreateindex-istok buildcreateindex-iitem buildcreateindex-iordr1 buildcreateindex-iordr2
buildcreateindex-iordl buildcreateindex-inord \
listfiles
"

tpcc_steps="runsqllocal-createdb shutdowndb startupdb-p_build createuser runscript-createts
assigntemp ddview \
  runsql-createtable_ware runsql-createtable_cust runsql-createtable_dist runsql-createtable_hist
runsql-createtable_stok runsql-createtable_item runsql-createtable_ordr runsql-createtable_ordl
runsql-createtable_nord \
runscript-loadware runscript-loaddist runscript-loaditem runscript-loadhist runscript-loadnord
runscript-loadordrordl runscript-loadcust runscript-loadstok \
 runsql-createindex_iware runsql-createindex_icust1 runsql-createindex_icust2 runsql-
createindex_idist runsql-createindex_istok runsql-createindex_iitem runsql-createindex_iordr1
runsql-createindex_iordr2 runsql-createindex_iordl runsql-createindex_inord \
analyze runscript-loadfixordrordl createstats createstoredprocs createspacestats createmisc"

# no longer automatically exports env variables
set +a

# check for problems with configuration
badconf=
for table in $tpcc_table_list; do
  if expr `tp $table imp` = queue > /dev/null; then
    if expr $tpcc_queue_avail = f > /dev/null; then
                        echo Table $table may not be a queue, since queues are
                        echo are unavailable in the selected Oracle version.
                        badconf=t
    fi
  fi
  if expr $tpcc_autospace_avail = f \& `tp $table autospace` = t > /dev/null; then
    echo Table $table may not use bitmapped space management
    echo since it is not available in the selected Oracle version.
    badconf=t
  fi
done

if test -n "$badconf"; then
  exit 1
fi

# make sure we have everything
if $tpcc_require ORACLE_SID \
 tpcc_tokilobytes tpcc_createts tpcc_lcm \
 tpcc_sqlplus tpcc_internal_connect\
 tpcc_np tpcc_cpu tpcc_os tpcc_runlen tpcc_ldrive tpcc_scale tpcc_disks_location
tpcc_auto_undo tpcc_tempts_min\
 tpcc_system_size tpcc_logfile_size \
 tpcc_undo_size tpcc_undo_bs\
 oracle_dba oracle_dba_password tpcc_dba_user_pass
then exit 1; fi
```

## options.sh

```
tpcc_os='unix'
tpcc_version='ttt'
tpcc_ldrive='1'
tpcc_scale='11000'
tpcc_np='1'
tpcc_cpu='4'
tpcc_memsize='88000'
tpcc_runlen='60'
tpcc_compress='f'
tpcc_overflow='t'
tpcc_defbs='4'

tpcc_cust_imp='cluster'
tpcc_cust_size='calc'
tpcc_cust_ext='calc'
tpcc_cust_nf='calc'
tpcc_cust_bs='auto'
tpcc_cust_used='-1'
tpcc_cust_free='0'
tpcc_cust_trans='3'
tpcc_cust_autospace='t'
tpcc_cust_flg='43'
tpcc_cust_fl='22'
tpcc_cust_rsize='auto'
tpcc_cust_hkey='auto'
tpcc_cust_hash='auto'
tpcc_cust_bpool=''
tpcc_cust_indices=3-2-1-

tpcc_dist_imp='cluster'
tpcc_dist_size='calc'
tpcc_dist_ext='calc'
tpcc_dist_nf='calc'
tpcc_dist_bs='2K'
tpcc_dist_used='-1'
tpcc_dist_free='-1'
tpcc_dist_trans='4'
tpcc_dist_autospace='t'
tpcc_dist_flg='43'
tpcc_dist_fl='22'
tpcc_dist_rsize='auto'
tpcc_dist_hkey='auto'
tpcc_dist_hash='auto'
tpcc_dist_bpool='default'
tpcc_dist_indices=2-1-

tpcc_hist_imp='table'
tpcc_hist_size='1791'
tpcc_hist_ext='calc'
tpcc_hist_nf='calc'
tpcc_hist_bs='auto'
tpcc_hist_used='-1'
tpcc_hist_free='5'
tpcc_hist_trans='4'
tpcc_hist_autospace='t'
tpcc_hist_flg='43'
tpcc_hist_fl='22'
tpcc_hist_rsize='auto'
tpcc_hist_hkey='auto'
tpcc_hist_hash='auto'
tpcc_hist_bpool=''
tpcc_hist_indices=no

tpcc_item_imp='cluster'
tpcc_item_size='calc'
tpcc_item_ext='calc'
tpcc_item_nf='calc'
tpcc_item_bs='auto'
tpcc_item_used='-1'
tpcc_item_free='0'
tpcc_item_trans='3'
tpcc_item_autospace='t'
tpcc_item_flg='43'
tpcc_item_fl='22'
tpcc_item_rsize='auto'
tpcc_item_hkey='auto'
tpcc_item_hash='auto'
tpcc_item_bpool='keep'
tpcc_item_indices=1-

tpcc_nord_imp='queue'
tpcc_nord_size='178'
tpcc_nord_ext='calc'
tpcc_nord_nf='calc'
tpcc_nord_bs='auto'
tpcc_nord_used='-1'
tpcc_nord_free='5'
tpcc_nord_trans='4'
tpcc_nord_autospace='t'
tpcc_nord_flg='43'
tpcc_nord_fl='22'
tpcc_nord_rsize='auto'
tpcc_nord_hkey='auto'
tpcc_nord_hash='auto'
tpcc_nord_bpool='default'
tpcc_nord_indices=1-2-3-

tpcc_ordl_imp='queue'
tpcc_ordl_size='21775'
tpcc_ordl_ext='calc'
tpcc_ordl_nf='calc'
tpcc_ordl_bs='16K'
tpcc_ordl_used='-1'
tpcc_ordl_free='5'
tpcc_ordl_trans='4'
```

```
tpcc_ordl_autospace='t'
tpcc_ordl_flg='43'
tpcc_ordl_fl='22'
tpcc_ordl_rsize='auto'
tpcc_ordl_hkey='auto'
tpcc_ordl_hash='auto'
tpcc_ordl_bpool='default'
tpcc_ordl_indices=1-2-3-4-

tpcc_ordr_imp='queue'
tpcc_ordr_size='1206'
tpcc_ordr_ext='calc'
tpcc_ordr_nf='calc'
tpcc_ordr_bs='16K'
tpcc_ordr_used='-1'
tpcc_ordr_free='5'
tpcc_ordr_trans='4'
tpcc_ordr_autospace='t'
tpcc_ordr_flg='43'
tpcc_ordr_fl='22'
tpcc_ordr_rsize='auto'
tpcc_ordr_hkey='auto'
tpcc_ordr_hash='auto'
tpcc_ordr_bpool='default'
tpcc_ordr_indices=2-3-1-

tpcc_stok_imp='cluster'
tpcc_stok_size='calc'
tpcc_stok_ext='calc'
tpcc_stok_nf='calc'
tpcc_stok_bs='auto'
tpcc_stok_used='-1'
tpcc_stok_free='0'
tpcc_stok_trans='3'
tpcc_stok_autospace='t'
tpcc_stok_flg='43'
tpcc_stok_fl='22'
tpcc_stok_rsize='auto'
tpcc_stok_hkey='auto'
tpcc_stok_hash='auto'
tpcc_stok_bpool='keep'
tpcc_stok_indices=1-2-

tpcc_ware_imp='cluster'
tpcc_ware_size='calc'
tpcc_ware_ext='calc'
tpcc_ware_nf='calc'
tpcc_ware_bs='2K'
tpcc_ware_used='-1'
tpcc_ware_free='-1'
tpcc_ware_trans='2'
tpcc_ware_autospace='t'
tpcc_ware_flg='43'
tpcc_ware_fl='22'
tpcc_ware_rsize='auto'
tpcc_ware_hkey='auto'
tpcc_ware_hash='auto'
tpcc_ware_bpool='default'
tpcc_ware_indices=1-

tpcc_icust1_imp='index'
tpcc_icust1_size='736'
tpcc_icust1_ext='calc'
tpcc_icust1_nf='calc'
tpcc_icust1_bs='16K'
tpcc_icust1_used='-1'
tpcc_icust1_free='1'
tpcc_icust1_trans='3'
tpcc_icust1_autospace='t'
tpcc_icust1_flg='43'
tpcc_icust1_fl='22'
tpcc_icust1_rsize='auto'
tpcc_icust1_hkey='auto'
tpcc_icust1_hash='auto'
tpcc_icust1_bpool='default'
tpcc_icust1_indices=3-2-1-

tpcc_icust2_imp='index'
tpcc_icust2_size='4591'
tpcc_icust2_ext='calc'
tpcc_icust2_nf='calc'
tpcc_icust2_bs='16K'
tpcc_icust2_used='-1'
tpcc_icust2_free='1'
tpcc_icust2_trans='3'
tpcc_icust2_autospace='t'
tpcc_icust2_flg='43'
tpcc_icust2_fl='22'
tpcc_icust2_rsize='auto'
tpcc_icust2_hkey='auto'
tpcc_icust2_hash='auto'
tpcc_icust2_bpool='default'
tpcc_icust2_indices=6-3-2-7-1-

tpcc_idist_imp='index'
tpcc_idist_size='4'
tpcc_idist_ext='calc'
tpcc_idist_nf='calc'
tpcc_idist_bs='2K'
tpcc_idist_used='-1'
tpcc_idist_free='5'
tpcc_idist_trans='3'
tpcc_idist_autospace='t'
tpcc_idist_flg='43'
tpcc_idist_fl='22'
tpcc_idist_rsize='auto'
tpcc_idist_hkey='auto'
tpcc_idist_hash='auto'
tpcc_idist_bpool='default'
tpcc_idist_indices=2-1-

tpcc_iitem_imp='index'
tpcc_iitem_size='2048'
tpcc_iitem_ext='calc'
tpcc_iitem_nf='calc'
tpcc_iitem_bs='auto'
tpcc_iitem_used='-1'
tpcc_iitem_free='5'
tpcc_iitem_trans='4'
tpcc_iitem_autospace='t'
tpcc_iitem_flg='43'
tpcc_iitem_fl='22'
tpcc_iitem_rsize='auto'
tpcc_iitem_hkey='auto'
tpcc_iitem_hash='auto'
tpcc_iitem_bpool='default'
tpcc_iitem_indices=1-

tpcc_inord_imp='none'
tpcc_inord_size='229'
tpcc_inord_ext='calc'
tpcc_inord_nf='calc'
tpcc_inord_bs='auto'
tpcc_inord_used='-1'
tpcc_inord_free='5'
tpcc_inord_trans='4'
tpcc_inord_autospace='t'
tpcc_inord_flg='43'
tpcc_inord_fl='22'
tpcc_inord_rsize='auto'
tpcc_inord_hkey='auto'
tpcc_inord_hash='auto'
tpcc_inord_bpool='default'
tpcc_inord_indices=1-2-3-

tpcc_iordl_imp='none'
tpcc_iordl_size='8072'
tpcc_iordl_ext='calc'
tpcc_iordl_nf='calc'
tpcc_iordl_bs='auto'
tpcc_iordl_used='-1'
tpcc_iordl_free='5'
tpcc_iordl_trans='4'
tpcc_iordl_autospace='t'
tpcc_iordl_flg='43'
tpcc_iordl_fl='22'
tpcc_iordl_rsize='auto'
tpcc_iordl_hkey='auto'
tpcc_iordl_hash='auto'
tpcc_iordl_bpool='default'
tpcc_iordl_indices=1-2-3-4-

tpcc_iordr1_imp='none'
tpcc_iordr1_size='703'
tpcc_iordr1_ext='calc'
tpcc_iordr1_nf='calc'
tpcc_iordr1_bs='auto'
tpcc_iordr1_used='-1'
tpcc_iordr1_free='1'
tpcc_iordr1_trans='3'
tpcc_iordr1_autospace='t'
tpcc_iordr1_flg='43'
tpcc_iordr1_fl='22'
tpcc_iordr1_rsize='auto'
tpcc_iordr1_hkey='auto'
tpcc_iordr1_hash='auto'
tpcc_iordr1_bpool='default'
tpcc_iordr1_indices=2-3-1-

tpcc_iordr2_imp='index'
tpcc_iordr2_size='1135'
tpcc_iordr2_ext='calc'
tpcc_iordr2_nf='calc'
tpcc_iordr2_bs='16K'
tpcc_iordr2_used='-1'
tpcc_iordr2_free='25'
tpcc_iordr2_trans='4'
tpcc_iordr2_autospace='t'
tpcc_iordr2_flg='43'
tpcc_iordr2_fl='22'
```

```
tpcc_iordr2_rsize='auto'
tpcc_iordr2_hkey='auto'
tpcc_iordr2_hash='auto'
tpcc_iordr2_bpool='default'
tpcc_iordr2_indices=4-3-2-1-

tpcc_istok_imp='index'
tpcc_istok_size='2090'
tpcc_istok_ext='calc'
tpcc_istok_nf='calc'
tpcc_istok_bs='16K'
tpcc_istok_used='-1'
tpcc_istok_free='1'
tpcc_istok_trans='3'
tpcc_istok_autospace='t'
tpcc_istok_flg='43'
tpcc_istok_fl='22'
tpcc_istok_rsize='auto'
tpcc_istok_hkey='auto'
tpcc_istok_hash='auto'
tpcc_istok_bpool='default'
tpcc_istok_indices=1-2-

tpcc_iware_imp='index'
tpcc_iware_size='1'
tpcc_iware_ext='calc'
tpcc_iware_nf='calc'
tpcc_iware_bs='auto'
tpcc_iware_used='-1'
tpcc_iware_free='1'
tpcc_iware_trans='3'
tpcc_iware_autospace='t'
tpcc_iware_flg='43'
tpcc_iware_fl='22'
tpcc_iware_rsize='auto'
tpcc_iware_hkey='auto'
tpcc_iware_hash='auto'
tpcc_iware_bpool='default'
tpcc_iware_indices=1-

tpcc_temp_imp='temp'
tpcc_temp_size='16145'
tpcc_temp_ext='calc'
tpcc_temp_nf='calc'
tpcc_temp_bs='auto'
tpcc_temp_used='-1'
tpcc_temp_free='0'
tpcc_temp_trans='3'
tpcc_temp_autospace='t'
tpcc_temp_flg='43'
tpcc_temp_fl='22'
tpcc_temp_rsize='auto'
tpcc_temp_hkey='auto'
tpcc_temp_hash='auto'
tpcc_temp_bpool='default'
tpcc_temp_indices=no
```

## 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=/ORACLE2/oracle/dbs/tpcc_disks

#locations of various files used in the generation scripts.
#(you can usually leave these alone.)
tpcc_sql_dir=${tpcc_bench}/scripts/sql
tpcc_log_dir=${tpcc_bench}/log
tpcc_genscripts_dir=${tpcc_bench}/scripts/generated


#Once you have filled all the options, comment
#out or delete this line.
#tpcc_no_options=t
```

## p_build.ora

```
compatible = 10.1.0.0.0
db_name = tpcc
control_files = $tpcc_disks_location/control_001
parallel_max_servers = 100
recovery_parallelism = 40
db_files = 524
db_block_size = 4096
db_cache_size = 30000M
db_2k_cache_size = 100M
db_8k_cache_size = 100M
db_16k_cache_size = 30000M
dml_locks = 500
log_buffer = 10485760
processes = 200
sessions = 200
transactions = 100
shared_pool_size = 1000M
cursor_space_for_time = TRUE
undo_management = auto
undo_retention = 2
UNDO_TABLESPACE = undo_ts
_ksmg_granule_size = 33554432
_column_compression_factor = 175  #added 04/24/03 req. Xumin
```

## p_create.ora

```
compatible = 10.1.0.0.0
db_name = tpcc
control_files = $tpcc_disks_location/control_001
db_block_size = 4096
db_cache_size = 100M
db_2k_cache_size = 50M
db_8k_cache_size = 50M
db_16k_cache_size = 50M
log_buffer = 1048576
undo_management = manual
```

## 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)
# global variable $tpcc_listfiles, does not execute sql

if expr x$tpcc_listfiles = xt > /dev/null; then
  echo $2 $3 >> $tpcc_bench/files.dat
  exit 0
fi


if expr $5 = auto > /dev/null; then
  bssql=
else
  bssql="blocksize $5"
fi

if expr $6 = 1 > /dev/null; then
  createsql="create temporary tablespace $1 tempfile '$2' size $3 reuse extent management local
uniform size $4;"
else
  if expr x$7 = xt > /dev/null; then
    autospace=auto
  else
    autospace=manual
  fi
    createsql="create tablespace $1 datafile '$2' size $3 reuse extent management local uniform size
$4 segment space management $autospace $bssql nologging ;"
fi

$tpcc_sqlplus $tpcc_user_pass <<!
   spool createts_$1.log
   set echo on
   drop tablespace $1 including contents;
 $createsql
   set echo off
   spool off
   exit ;
!
```

## loadware.sh

```
cd $tpcc_bench
```

```
$tpcc_load -M $tpcc_scale -w > loadware.log 2>&1
```

## loaddist.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -d > loaddist.log 2>&1
```

## loaditem.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -i > loaditem.log 2>&1
```

## loadhist.sh

```
#created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/evenload.sh Tue Apr 8 16:32:56
PDT 2003
rm loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 11000 -h -b 1 -e 687 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 688 -e 1374 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 1375 -e 2061 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 2062 -e 2748 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 2749 -e 3435 >> loadhist4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 3436 -e 4122 >> loadhist5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 4123 -e 4809 >> loadhist6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 4810 -e 5496 >> loadhist7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 5497 -e 6184 >> loadhist8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 6185 -e 6872 >> loadhist9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 6873 -e 7560 >> loadhist10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 7561 -e 8248 >> loadhist11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 8249 -e 8936 >> loadhist12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 8937 -e 9624 >> loadhist13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 9625 -e 10312 >> loadhist14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -h -b 10313 -e 11000 >> loadhist15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

## loadnord.sh

```
#created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/evenload.sh Tue Apr 8 16:32:56
PDT 2003
rm loadnord*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 11000 -n -b 1 -e 687 >> loadnord0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 688 -e 1374 >> loadnord1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 1375 -e 2061 >> loadnord2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 2062 -e 2748 >> loadnord3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 2749 -e 3435 >> loadnord4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 3436 -e 4122 >> loadnord5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 4123 -e 4809 >> loadnord6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 4810 -e 5496 >> loadnord7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 5497 -e 6184 >> loadnord8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 6185 -e 6872 >> loadnord9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 6873 -e 7560 >> loadnord10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 7561 -e 8248 >> loadnord11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 8249 -e 8936 >> loadnord12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 8937 -e 9624 >> loadnord13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 9625 -e 10312 >> loadnord14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -n -b 10313 -e 11000 >> loadnord15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

## loadordrordl.sh

```
#created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/evenload.sh Tue Apr 8 16:32:57
PDT 2003
rm loadordrordl*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy0.dat -b 1 -e 687 >> loadordrordl0.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy1.dat -b 688 -e 1374 >> loadordrordl1.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy2.dat -b 1375 -e 2061 >> loadordrordl2.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy3.dat -b 2062 -e 2748 >> loadordrordl3.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy4.dat -b 2749 -e 3435 >> loadordrordl4.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy5.dat -b 3436 -e 4122 >> loadordrordl5.log
2>&1 &
allprocs="$allproc s ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy6.dat -b 4123 -e 4809 >> loadordrordl6.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy7.dat -b 4810 -e 5496 >> loadordrordl7.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy8.dat -b 5497 -e 6184 >> loadordrordl8.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy9.dat -b 6185 -e 6872 >> loadordrordl9.log
2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy10.dat -b 6873 -e 7560 >>
loadordrordl10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy11.dat -b 7561 -e 8248 >>
loadordrordl11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy12.dat -b 8249 -e 8936 >>
loadordrordl12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy13.dat -b 8937 -e 9624 >>
loadordrordl13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy14.dat -b 9625 -e 10312 >>
loadordrordl14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -o $tpcc_disks_location/dummy15.dat -b 10313 -e 11000 >>
loadordrordl15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

## loadcust.sh

```
#created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/evenload.sh Tue Apr 8 16:32:58
PDT 2003
rm loadcust*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 11000 -c  -b 1 -e 687 >> loadcust0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 688 -e 1374 >> loadcust1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 1375 -e 2061 >> loadcust2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 2062 -e 2748 >> loadcust3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 2749 -e 3435 >> loadcust4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 3436 -e 4122 >> loadcust5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 4123 -e 4809 >> loadcust6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 4810 -e 5496 >> loadcust7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 5497 -e 6184 >> loadcust8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 6185 -e 6872 >> loadcust9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 6873 -e 7560 >> loadcust10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 7561 -e 8248 >> loadcust11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 8249 -e 8936 >> loadcust12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 8937 -e 9624 >> loadcust13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 9625 -e 10312 >> loadcust14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -c  -b 10313 -e 11000 >> loadcust15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

## loadstok.sh

```
#created automatically by /mnt9/tpcc-kits/tpcc-kit-11k/scripts/evenload.sh Tue Apr 8 16:32:59
PDT 2003
rm loadstok*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 11000 -S  -j 1 -k 6250 >> loadstok0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 6251 -k 12500 >> loadstok1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 12501 -k 18750 >> loadstok2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 18751 -k 25000 >> loadstok3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 25001 -k 31250 >> loadstok4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 31251 -k 37500 >> loadstok5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 37501 -k 43750 >> loadstok6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 43751 -k 50000 >> loadstok7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 50001 -k 56250 >> loadstok8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 56251 -k 62500 >> loadstok9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 62501 -k 68750 >> loadstok10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 68751 -k 75000 >> loadstok11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 75001 -k 81250 >> loadstok12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 81251 -k 87500 >> loadstok13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 87501 -k 93750 >> loadstok14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 11000 -S  -j 93751 -k 100000 >> loadstok15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

## Createts.ksh

```
#!/bin/ksh
#NOTE - ANY CHANGES MUST BE MADE TO CREATETS.SH AS WELL.

# createts.sh [name] [no. of file] [no. of partition] [filesize] [ext_size]
#            [unix/nt] [1: temporary ts / 0: others] [filecount] [no of cpu]
#                          [blocksize]

name=$1
fileno=$2
noofts=$3
filesize=$4
extsize=$5
ver=$6
isTemp=$7
filecount=$8
para=`expr $9 \* 2`
blocksize=${10}
autospace=${11}

addts=$tpcc_scripts/addts.sh
addfile=$tpcc_scripts/addfile.sh

if [ $ver = unix ];
then
  fileaddr=$tpcc_disks_location/;
elif [ $ver = nt ];
then
  fileaddr=\\\\\\\\\\\\\\\.\\\\\\;
fi

i=0
while [ $i -lt $noofts ] ;do

  filecount=`expr $filecount + 1`;

  $addts $name\_$i $fileaddr$name\_$i\_0 $filesize $extsize $blocksize $isTemp $autospace \>
junk$filecount 2\>\&1 \&;
  typeset proc$filecount=$!

  p=`expr $filecount % $para`;
  if [ $p = 0 ]
  then
    k=`expr $filecount - $para + 1`;
    if [ $k -le $8 ];
    then
      k=`expr $8 + 1`;
    fi
    while [ $k -le $filecount ] ;do
      wait proc$k
      typeset proc$k=$?
      k=`expr $k + 1`;
    done
  fi

  i=`expr $i + 1`;

done

p=`expr $filecount % $para`
if [ $p != 0 ]
then
  k=`expr $filecount - $p + 1`;
  if [ $k -le $8 ];
  then
    k=`expr $8 + 1`;
  fi
  while [ $k -le $filecount ] ;do
    wait proc$k
    typeset proc$k=$?
    k=`expr $k + 1`
  done
fi

fileperts=`expr $fileno / $noofts - 1`
if [ $fileperts -gt 0 ];
then
  i=0
  while [ $i -lt $noofts ] ;do
    j=0;
    while [ $j-lt $fileperts ] ;do
```

```
      filecount=`expr $filecount + 1`;
      $addfile $name\_$i $fileaddr$name\_$i\_`expr $j + 1` $filesize $isTemp \> junk$filecount
2\>\&1 \&;
      typeset proc$filecount=$!

      p=`expr $filecount % $para`;
      if [ $p = 0 ]
      then
        k=`expr $filecount - $para + 1`;
        if [ $k  -le $8 ];
        then
          k=`expr $8 + 1`;
        fi
       while [ $k  -le $filecount ] ;do
          wait proc$k
          typeset proc$k=$?
          k=`expr $k + 1`;
        done
      fi

      j=`expr $j + 1`
    done

    i=`expr $i + 1`
  done

  p=`expr $filecount % $para`
  if [ $p != 0 ]
  then
    k=`expr $filecount - $p + 1`;
    if [ $k  -le $8 ];
    then
      k=`expr $8 + 1`;
    fi
    while [ $k  -le $filecount ] ;do
      wait proc$k
      typeset proc$k=$?
      k=`expr $k + 1`
    done
  fi
fi

i=`expr $8 + 1`
proc=0
while [ $i -le $filecount ] ;do
  typeset  process=$((proc$i))
  proc=`expr $proc + $process`
  i=`expr $i + 1`
done

out=`expr $proc % 127`

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

## 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;
```

## 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;
```

## 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;
```

## 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.  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_ty pe = '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;
```

## Createdb.sql

```
/* created automatically by  /ORACLE2/build11K/tpcc-kit-11k/scripts/buildcreatedb.sh Thu Apr
10 14:52:02 PDT 2003 */
/* manually modified 04/23/03 */
spool createdb.log

set echo on

shutdown abort

startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxinstances 1
  datafile '$tpcc_disks_location/system_001' size 400M reuse
  logfile '$tpcc_disks_location/log_1' size 9880M reuse,
          '$tpcc_disks_location/log_2' size 9880M reuse
  sysaux datafile '$tpcc_disks_location/aux.df' size 120M reuse ;

create undo tablespace undo_ts datafile
  '$tpcc_disks_location/roll01' size 15900M reuse blocksize 8K;

set echo off
exit sql.sqlcode
```

## Analyze.sql

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

# this one tends to fail if indices aren't made, which is legal, so
# always exit without error.

exit 0
```

## Createstoredprocs.sql

```
spool createstoredprocs.log
@$tpcc_sql_dir/tkvcinin.sql
spool off
exit sql.sqlcode;
```

## Createspacestats.sql

```
 spool createspacestats.log
@$tpcc_sql_dir/space_init
@$tpcc_sql_dir/space_get 132000 10440
@$tpcc_sql_dir/space_rpt
spool off
exit sql.sqlcode;
```

## Createuser.sql

```
spool createusertpcc.log;

set echo on;

create user tpcc identified by tpcc;

grant dba to tpcc;

set echo off;
spool off;

exit ;
```

## Shutdowndb.sh

```
#!/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
!
```

## startupdb.sh

```
#!/bin/sh

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

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool startdb.log

set echo on

startup pfile=${1}.ora open

spool off
set echo off
exit sql.sqlcode
!
```

## 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
/
```

## 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;
```

## pay.sql

```
rem
rem
===================================================================+
rem       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
rem                  OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                       All Rights Reserved                       |
rem
===================================================================+
rem  FILENAME
rem       pay.sql
rem  DESCRIPTION
rem       SQL script to create a stored procedure for payment
rem       transactions.
rem ===================================================================
rem

CREATE OR REPLACE PACKAGE payment
IS
   PROCEDURE dopayment_z
   (
      ware_id                INTEGER,
      dist_id                INTEGER,
      cust_w_id              INTEGER,
      cust_d_id              INTEGER,
      cust_id          IN OUT INTEGER,
      bylastname             INTEGER,
      hist_amount            INTEGER,
      cust_last        IN OUT VARCHAR2,
      ware_street_1       OUT VARCHAR2,
      ware_street_2       OUT VARCHAR2,
      ware_city           OUT VARCHAR2,
      ware_state          OUT VARCHAR2,
      ware_zip            OUT VARCHAR2,
      dist_street_1       OUT VARCHAR2,
      dist_street_2       OUT VARCHAR2,
      dist_city           OUT VARCHAR2,
      dist_state          OUT VARCHAR2,
      dist_zip            OUT VARCHAR2,
      cust_first          OUT VARCHAR2,
      cust_middle         OUT VARCHAR2,
      cust_street_1       OUT VARCHAR2,
      cust_street_2       OUT VARCHAR2,
      cust_city           OUT VARCHAR2,
      cust_state          OUT VARCHAR2,
      cust_zip            OUT VARCHAR2,
      cust_phone          OUT VARCHAR2,
      cust_since          OUT VARCHAR2,
      cust_credit      IN OUT VARCHAR2,
      cust_credit_lim     OUT NUMBER,
      cust_discount       OUT NUMBER,
      cust_balance     IN OUT NUMBER,
      cust_data           OUT VARCHAR2,
      hist_date           OUT VARCHAR2,
      retry            IN OUT INTEGER,
      cur_date               IN DATE
   );

PROCEDURE dopayment_nz
   (
      ware_id              INTEGER,
      dist_id              INTEGER,
      cust_w_id            INTEGER,
      cust_d_id            INTEGER,
      cust_id          IN OUT INTEGER,
      bylastname           INTEGER,
      hist_amount          INTEGER,
      cust_last        IN OUT VARCHAR2,
      ware_street_1       OUT VARCHAR2,
      ware_street_2       OUT VARCHAR2,
      ware_city           OUT VARCHAR2,
      ware_state          OUT VARCHAR2,
      ware_zip            OUT VARCHAR2,
      dist_street_1       OUT VARCHAR2,
      dist_street_2       OUT VARCHAR2,
      dist_city           OUT VARCHAR2,
      dist_state          OUT VARCHAR2,
      dist_zip            OUT VARCHAR2,
      cust_first          OUT VARCHAR2,
      cust_middle         OUT VARCHAR2,
      cust_street_1       OUT VARCHAR2,
      cust_street_2       OUT VARCHAR2,
      cust_city           OUT VARCHAR2,
      cust_state          OUT VARCHAR2,
      cust_zip            OUT VARCHAR2,
      cust_phone          OUT VARCHAR2,
      cust_since          OUT VARCHAR2,
      cust_credit      IN OUT VARCHAR2,
      cust_credit_lim     OUT NUMBER,
      cust_discount       OUT NUMBER,
      cust_balance     IN OUT NUMBER,
      cust_data           OUT VARCHAR2,
      hist_date           OUT VARCHAR2,
      retry            IN OUT INTEGER,
      cur_date               IN DATE
   );
END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY payment
IS
   PROCEDURE dopayment_z
   (
      ware_id              INTEGER,
      dist_id              INTEGER,
      cust_w_id            INTEGER,
      cust_d_id            INTEGER,
      cust_id          IN OUT INTEGER,
      bylastname           INTEGER,
      hist_amount          INTEGER,
      cust_last        IN OUT VARCHAR2,
      ware_street_1       OUT VARCHAR2,
      ware_street_2       OUT VARCHAR2,
      ware_city           OUT VARCHAR2,
      ware_state          OUT VARCHAR2,
      ware_zip            OUT VARCHAR2,
      dist_street_1       OUT VARCHAR2,
      dist_street_2       OUT VARCHAR2,
      dist_city           OUT VARCHAR2,
      dist_state          OUT VARCHAR2,
      dist_zip            OUT VARCHAR2,
      cust_first          OUT VARCHAR2,
      cust_middle         OUT V ARCHAR2,
      cust_street_1       OUT VARCHAR2,
      cust_street_2       OUT VARCHAR2,
      cust_city           OUT VARCHAR2,
      cust_state          OUT VARCHAR2,
      cust_zip            OUT VARCHAR2,
      cust_phone          OUT VARCHAR2,
      cust_since          OUT VARCHAR2,
      cust_credit      IN OUT VARCHAR2,
      cust_credit_lim     OUT NUMBER,
      cust_discount       OUT NUMBER,
      cust_balance     IN OUT NUMBER,
      cust_data           OUT VARCHAR2,
      hist_date           OUT VARCHAR2,
      retry            IN OUT INTEGER,
      cur_date               IN DATE
   )
   IS
      TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
      cust_rowid          ROWID;
      ware_rowid          ROWID;
      dist_ytd                        NUMBER(12);
      dist_name           VARCHAR2(11);
      ware_ytd                        NUMBER(12);
      ware_name           VARCHAR2(11);
      history_date        DATE;
      c_num               BINARY_INTEGER;
      row_id              rowidarray;
      cust_payments                   PLS_INTEGER;
      cust_ytd                        NUMBER(12);
      cust_data_temp                  VARCHAR2(500);
      node_num                        VARCHAR2(512);
      not_serializable    EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
      deadlock            EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old    EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
      CURSOR c_cur IS
               SELECT rowid
         FROM cust
        WHERE c_d_id = cust_d_id AND c_w_id = cust_w_id AND c_last = cust_last
         ORDER BY c_w_id, c_d_id, c_last, c_first;
   BEGIN
     SELECT substr(value,1,5)
        INTO node_num
        FROM v$parameter
       WHERE name = 'instance_number';

     plsql_mon_pack.print ('Payment started at ' ||
                    to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                    node_num);
      LOOP BEGIN
        SELECT rowid, c_first, c_middle, c_last, c_street_1, c_street_2,
             c_city, c_state, c_zip, c_phone,
             to_char (c_since, 'DD-MM-YYYY'), c_credit, c_credit_lim,
             c_discount, c_balance - hist_amount, c_payment_cnt ,
                          c_ytd_payment + hist_amount, decode (c_credit, 'BC', c_data , ' ')
          INTO cust_rowid, cust_first, cust_middle, cust_last,
               cust_street_1, cust_street_2, cust_city, cust_state,
               cust_zip, cust_phone, cust_since, cust_credit,
               cust_credit_lim, cust_discount, cust_balance, cust_payments,
                          cust_ytd, cust_data_temp
          FROM cust
         WHERE c_id = cust_id AND c_d_id = cust_d_id AND
               c_w_id = cust_w_id;
        cust_payments := cust_payments + 1;
        IF cust_credit = 'BC' THEN
                    cust_data_temp := substr ((to_char (cust_id) || ' ' ||
                              to_char (cust_d_id) || ' ' ||
                              to_char (cust_w_id) || ' ' ||
                              to_char (dist_id) || ' ' ||
                              to_char (ware_id) || ' ' ||
                              to_char (hist_amount, '9999.99') || ' | ')
                              || cust_data_temp, 1, 500);

          UPDATE cust
             SET c_balance = cust_balance,
                 c_ytd_payment = cust_ytd,
                 c_payment_cnt = cust_payments,
                           c_data = cust_data_temp
           WHERE rowid = cust_rowid;

                    cust_data := substr(cust_data_temp,1, 200);

        ELSE

          UPDATE cust
             SET c_balance = cust_balance,
                 c_ytd_payment = cust_ytd,
                 c_payment_cnt = cust_payments
           WHERE rowid = cust_rowid;

                    cust_data := cust_data_temp;
        END IF;

        SELECT dist.rowid, d_name, d_street_1, d_street_2, d_city,
                     d_state, d_zip, d_ytd + hist_amount,
              ware.rowid, w_name, w_street_1, w_street_2, w_city,
              w_state, w_zip, w_ytd + hist_amount
          INTO cust_rowid, dist_name, dist_street_1, dist_street_2, dist_city,
               dist_state, dist_zip, dist_ytd,
               ware_rowid, ware_name, ware_street_1, ware_street_2, ware_city,
               ware_state, ware_zip, ware_ytd
          FROM dist, ware
```

```
    WHERE d_id = dist_id
      AND d_w_id = ware_id
      AND w_id = ware_id;

  UPDATE dist
    SET d_ytd = dist_ytd
    WHERE rowid = cust_rowid;

  UPDATE ware
    SET w_ytd = ware_ytd
    WHERE rowid = ware_rowid;

  history_date := cur_date;

  INSERT INTO hist(h_c_id,h_c_d_id, h_c_w_id,h_d_id,h_w_id,h_date,
        h_amount, h_data) VALUES
    (cust_id, cust_d_id, cust_w_id, dist_id, ware_id, history_date,
     hist_amount, ware_name || '    ' || dist_name);
  COMMIT;
  hist_date := to_char (history_date, 'DD-MM-YYYY.HH24:MI:SS');
  EXIT;

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

 END LOOP;
END;


PROCEDURE dopayment_nz
(
 ware_id          INTEGER,
 dist_id          INTEGER,
 cust_w_id        INTEGER,
 cust_d_id        INTEGER,
 cust_id          IN OUT INTEGER,
 bylastname       INTEGER,
 hist_amount      INTEGER,
 cust_last        IN OUT VARCHAR2,
 ware_street_1    OUT VARCHAR2,
 ware_street_2    OUT VARCHAR2,
 ware_city        OUT VARCHAR2,
 ware_state       OUT VARCHAR2,
 ware_zip         OUT VARCHAR2,
 dist_street_1    OUT VARCHAR2,
 dist_street_2    OUT VARCHAR2,
 dist_city        OUT VARCHAR2,
 dist_state       OUT VARCHAR2,
 dist_zip         OUT VARCHAR2,
 cust_first       OUT VARCHAR2,
 cust_middle      OUT VARCHAR2,
 cust_street_1    OUT VARCHAR2,
 cust_street_2    OUT VARCHAR2,
 cust_city        OUT VARCHAR2,
 cust_state       OUT VARCHAR2,
 cust_zip         OUT VARCHAR2,
 cust_phone       OUT VARCHAR2,
 cust_since       OUT VARCHAR2,
 cust_credit      IN OUT VARCHAR2,
 cust_credit_lim  OUT NUMBER,
 cust_discount    OUT NUMBER,
 cust_balance     IN OUT NUMBER,
 cust_data        OUT VARCHAR2,
 hist_date        OUT VARCHAR2,
 retry            IN OUT INTEGER,
 cur_date                   IN DATE
)
IS
 TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
 cust_rowid       ROWID;
 ware_rowid       ROWID;
 dist_ytd                   NUMBER(12);
 dist_name        VARCHAR2(11);
 ware_ytd                   NUMBER(12);
 ware_name        VARCHAR2(11);
 history_date     DATE;
 c_num            BINARY_INTEGER;
 row_id           rowidarray;
 cust_payments              PLS_INTEGER;
 cust_ytd                   NUMBER(12);
 cust_data_temp             VARCHAR2(500);
 node_num                   VARCHAR2(512);
 not_serializable   EXCEPTION;
 PRAGMA EXCEPTION_INIT(not_serializable,-8177);
 deadlock           EXCEPTION;
 PRAGMA EXCEPTION_INIT(deadlock,-60);
 snapshot_too_old   EXCEPTION;
 PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
 CURSOR c_cur IS
   SELECT rowid
   FROM cust
  WHERE c_d_id = cust_d_id AND c_w_id = cust_w_id AND c_last = cust_last
    ORDER BY c_w_id, c_d_id, c_last, c_first;
BEGIN
 SELECT substr(value,1,5)
    INTO node_num
    FROM v$parameter
   WHERE name = 'instance_number';

 plsql_mon_pack.print ('Payment started at ' ||
                 to_char (sysdate, 'HH24:MI:SS') || ' on node ' ||
                 node_num);
 LOOP BEGIN

   c_num := 0;
   FOR c_id_rec IN c_cur LOOP
     c_num := c_num + 1;
     row_id(c_num) := c_id_rec.rowid;
   END LOOP;
   cust_rowid := row_id ((c_num + 1) / 2);    -- use row_id.count ?

   SELECT c_id, c_first, c_middle, c_last, c_street_1, c_street_2,
        c_city, c_state, c_zip, c_phone,
         to_char (c_since,'DD-MM-YYYY'), c_credit, c_credit_lim,
         c_discount, c_balance - hist_amount, c_payment_cnt,
                          c_ytd_payment + hist_amount, decode (c_credit, 'BC', c_data, ' ')
     INTO cust_id, cust_first, cust_middle, cust_last,
         cust_street_1, cust_street_2, cust_city, cust_state,
         cust_zip, cust_phone, cust_since, cust_credit,
         cust_credit_lim, cust_discount, cust_balance, cust_payments,
                          cust_ytd, cust_data_temp
     FROM cust
     WHERE rowid = cust_rowid;
   cust_payments := cust_payments + 1;
   IF cust_credit = 'BC' THEN
                cust_data_temp := substr ((to_char (cust_id) || ' ' ||
                      to_char (cust_d_id) || ' ' ||
                      to_char (cust_w_id) || ' ' ||
                      to_char (dist_id) || ' ' ||
                      to_char (ware_id) || ' ' ||
                      to_char (hist_amount/100, '9999.99') || ' | ')
                      || cust_data_temp, 1, 500);

     UPDATE cust
       SET c_balance = cust_balance,
           c_ytd_payment = cust_ytd,
           c_payment_cnt = cust_payments,
                          c_data = cust_data_temp
       WHERE rowid = cust_rowid;

                cust_data := substr(cust_data_temp,1, 200);

   ELSE

     UPDATE cust
       SET c_balance = cust_balance,
           c_ytd_payment = cust_ytd,
           c_payment_cnt = cust_payments
       WHERE rowid = cust_rowid;

                cust_data := cust_data_temp;

   END IF;

   SELECT dist.rowid, d_name, d_street_1, d_street_2, d_city,
                 d_state, d_zip, d_ytd + hist_amount,
         ware.rowid, w_name, w_street_1, w_street_2, w_city,
          w_state, w_zip, w_ytd + hist_amount
     INTO cust_rowid, dist_name, dist_street_1, dist_street_2, dist_city,
          dist_state, dist_zip, dist_ytd,
          ware_rowid, ware_name, ware_street_1, ware_street_2, ware_city,
          ware_state, ware_zip, ware_ytd
     FROM dist, ware
     WHERE d_id = dist_id
       AND d_w_id = ware_id
       AND w_id = ware_id;

   UPDATE dist
     SET d_ytd = dist_ytd
     WHERE rowid = cust_rowid;

   UPDATE ware
     SET w_ytd = ware_ytd
     WHERE rowid = ware_rowid;

   history_date := cur_date;

   INSERT INTO hist VALUES
     (cust_id, cust_d_id, cust_w_id, dist_id, ware_id, history_date,
      hist_amount, ware_name || '    ' || dist_name);
   COMMIT;
   hist_date := to_char (history_date, 'DD-MM-YYYY.HH24:MI:SS');
   EXIT;

   EXCEPTION
     WHEN not_serializable OR deadlock OR snapshot_too_old THEN
       ROLLBACK;
```

```
            retry := retry + 1;
         END;

      END LOOP;
   END;
END;
/
show errors;

quit;
```

## 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 ;
!
```

## analyze.sh

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

# this one tends to fail if indices aren't made, which is legal, so
# always exit without error.

exit 0
```

## ddviews.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

@$ORACLE_HOME/rdbms/admin/catparr

REM fi

spool off
!

sh $tpcc_scripts/queue.sh
```

## 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
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_sql_dir/createspacestats > junk 2>&1

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

## createstats.sh

```
#!/bin/sh

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

REM
REM create tablespace for statspack user sp begin
REM

spool createstats.log

set echo on
   drop tablespace sp including contents;
   create tablespace sp_0 datafile '$tpcc_disks_location/sp_0' size $tpcc_statspack_size reuse
autoextend on extent management local uniform size 1M 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

@$tpcc_sql_dir/c_stat
@$tpcc_sql_dir/pst_c

!
```

## createstoredprocs.sh

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

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

## createts.sh

```
#!/bin/sh
#NOTE - ANY CHANGES MUST BE MADE TO CREATETS.KSH AS WELL.

# createts.sh [name] [no. of file] [no. of partition] [filesize] [ext_size]
#            [unix/nt] [1: temporary ts / 0: others] [filecount] [no of cpu]
#                          [blocksize] [t: bitmapped / f: dict manage ]

name=$1
fileno=$2
noofts=$3
filesize=$4
extsize=$5
ver=$6
isTemp=$7
filecount=$8
para=`expr $9 \* 2`
#blocksize=${10}  sh bug workaround
blocksize=`echo $@ | cut -d' ' -f10`
#autospace=${11} sh bug workaround
autospace=`echo $@ | cut -d' ' -f11`

addts=$tpcc_scripts/addts.sh
addfile=$tpcc_scripts/addfile.sh

if test $ver = unix;
then
  fileaddr=$tpcc_disks_location/;
elif test $ver = nt;
then
  fileaddr=\\\\\\\\\\\\\\\.\\\\\\;
fi

i=0
while test $i -lt $noofts; do

  filecount=`expr $filecount + 1`;

  $addts $name\_$i $fileaddr$name\_$i\_0 $filesize $extsize $blocksize $isTemp $autospace \>
junk$filecount 2\>\&1 \&;
  eval "proc$filecount=$!"

  p=`expr $filecount % $para`;
  if test $p = 0;
  then
    k=`expr $filecount - $para + 1`;
    if test $k -le $8;
    then
      k=`expr $8 + 1`;
    fi
    while test $k -le $filecount ; do
      wait `eval echo '$proc'$k`
      eval "proc$k=$?"
      k=`expr $k + 1`;
    done
  fi

  i=`expr $i + 1`;

done

p=`expr $filecount % $para`
if test $p != 0;
then
  k=`expr $filecount - $p + 1`;
  if test $k -le $8;
  then
    k=`expr $8 + 1`;
  fi
  while test $k -le $filecount; do
    wait `eval echo '$proc'$k`
    eval "proc$k=$?"
    k=`expr $k + 1`
  done
fi

fileperts=`expr $fileno / $noofts - 1`
if test $fileperts -gt 0 ;
then
  i=0
  while test $i -lt $noofts ; do
    j=0;
    while test $j -lt $fileperts ;do

      filecount=`expr $filecount + 1`;
      $addfile $name\_$i $fileaddr$name\_$i\_`expr $j + 1` $filesize $isTemp \> junk$filecount
2\>\&1 \&;
      eval "proc$filecount=$!"

      p=`expr $filecount % $para`;
      if test $p = 0;
      then
        k=`expr $filecount - $para + 1`;
        if test $k -le $8;
        then
          k=`expr $8 + 1`;
        fi
       while test $k -le $filecount ; do
          wait `eval echo '$proc'$k`
          eval "proc$k=$?"
          k=`expr $k + 1`;
        done
      fi

      j=`expr $j + 1`
    done

    i=`expr $i + 1`
  done

  p=`expr $filecount % $para`
  if test $p != 0;
  then
    k=`expr $filecount - $p + 1`;
    if test $k -le $8;
    then
      k=`expr $8 + 1`;
    fi
    while test $k -le $filecount; do
      wait `eval echo '$proc'$k`
      eval "proc$k=$?"
      k=`expr $k + 1`
    done
  fi
fi

i=`expr $8 + 1`
proc=0
while test $i -le $filecount ;do
  eval 'process=$proc'"$i"
  proc=`expr $proc + $process`
  i=`expr $i + 1`
done

out=`expr $proc % 127`

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

## 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;

set echo off;
```

## 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;
```

# Appendix C Tunable Parameters

The HP-UX operating system tunable parameters employed to generate the kernel for the hp server rx5670 and the 9 HP 9000 Model hp server rp2470 clients are listed below. Included as well are the Oracle10i Database Standard Edition and TUXEDO 8.0 parameters.

## C.1 HP-UX Configuration - Clients

### Config/Client2/ostune.ver

```
****************************************
* Source: /ux/core/kern/filesets.info/CORE-KRN/generic
* @(#)B.11.11_LR
*
****************************************
* Additional drivers required in every machine-type to create a complete
* system file during cold install.  This list is every driver that the
* master.d/ files do not force on the system or is not identifiable by
* ioscan.
* Other CPU-type specific files can exist for their special cases.
* see create_sysfile (1m).
****************************************
*
* Drivers/Subsystems
sba
lba
btlan
c720
sctl
sdisk
asio0
cdfs
cxperf
diag0
diag1
diag2
dmem
dev_config
iomem
nfs_core
nfs_client
nfs_server
maclan
dlpi
token_arp
inet
uipc
tun
telm
tels
netdiag1
nms
hpstreams
clone
strlog
sad
echo
sc
timod
tirdwr
pipedev
pipemod
ffs
ldterm
ptem
pts
ptm
pckt
vxfs
vxportal
lvm
lv
nfsm
rpcmod
autofsc
cachefsc
cifs
fddi4
gelan
GSCtoPCI
iop_drv
bs_osm
ipmi
ipmi_psm
func0
td
iether
igelan
vxvm
vxdmp
vol
vols
stape
tape2
dev_olar
olar_psm
olar_psm_if

*************
* Tunables
*************
STRMSGSZ        65535
bufpages        8192
create_fastlinks    1
dbc_min_pct         0
dbc_max_pct         0
default_disk_ir     1
fs_async            1
maxfiles        2048
maxfiles_lim     2048
maxdsiz         0x80000000
maxssiz         0X10000000
maxswapchunks                    4096
maxuprc         200
nproc           (100+MAXUPRC)
max_thread_proc 1030
nkthread        14000
msgmni          NKTHREAD
msgtql          NKTHREAD
msgseg          (MSGMNI*2)
msgssz          512
msgmap          (MSGSEG)
msgmax          32768
msgmnb          (MSGMAX*2)
nfile           (NKTHREAD+NPROC*5)
ninode          (NKTHREAD+NPROC*5)
npty            128
nstrpty         200
semmni          32
semmns          NKTHREAD
semmnu          (SEMMNS)
semume          4
semvmx          40960
shmmax          0X40000000
shmmni          16
shmseg          16
swapmem_on      0
unlockable_mem  1
nhtbl_scale     1
vps_ceiling     4
```

### Config/Server/ostune.ver

```
*
* Created on Tue Jun 24 15:36:25 2003
*
version 1
configuration nextboot "" [3ef79f48]
*
* Module entries
*
module pfil auto 0.1.0
module gelan best [3EE76E6A]
module asyncdsk best [3EE4C9D8]
module pty0 best [3EE4C9CE]
module pty1 best [3EE4C9CE]
module ipmi best [3EE4C9F8]
module LCentIf best [3EE4C366]
module acpi_node best [3EE4CB45]
module sac best [3EE4C366]
module wxb_hp best [3EE4C366]
module ia64_psm best [3EE4C9FA]
module pdh best [3EE4C9FA]
module c8xx best [3EE4C5A4]
module diag2 best [3EE4C5A4]
module dmem best [3EE4C9C5]
module dev_config best [3EE4C9FB]
module cdfs best 0.1.0
module rng best 0.1.0
module inet best [3EE4CB37]
module uipc best [3EE4C9CD]
module tun best [3EE4C9CE]
module telm best [3EE4C9D0]
module tels best [3EE4C9D0]
module intl100 best [3EE4C9FA]
module dlpi best [3EE4C9F3]
module token_arp best [3EE4C9F3]
module netdiag1 best [3EE4C9D4]
module nms best [3EE4C9EC]
module nfs_core best [3EE4C9EF]
module nfsm best [3EE4C9EE]
module rpcmod best [3EE4C9DF]
module autofsc best [3EE4CB41]
module cachefsc best [3EE4CB3F]
module hpstreams best [3EE4C9D8]
module clone best [3EE4C9D8]
module strlog best [3EE4C9D8]
module sad best [3EE4C9D8]
module echo best [3EE4C9D8]
module sc best [3EE4C9D8]
module timod best [3EE4C9D8]
module tirdwr best [3EE4C9D8]
module pipedev best [3EE4C9D8]
module pipemod best [3EE4C9D8]
module ffs best [3EE4C9D8]
module ldterm best [3EE4C9D5]
module ptem best [3EE4C9D5]
module pts best [3EE4C9D5]
module ptm best [3EE4C9D5]
module pckt best [3EE4C9D5]
module td best [3EE76E36]
module fddi4 best [3EE4CB39]
module iether best [3EE76E42]
module igelan best [3EE76E3C]
module vxfs best [3EE4C9C3]
module lvm best [3EE4C9F2]
module lv best [3EE4C9F2]
module vxdmp best [3EE76E5F]
module vol best [3EE76E60]
module vols best [3EE76E60]
module mpt best [3EE76E51]
module root best [3EE4C9FA]
module sba best [3EE4C9FB]
module lba best [3EE4C9EC]
module asio0 best [3EE4C366]
module tgt best [3EE4C5A4]
module sdisk best [3EE4C5A4]
module sctl best [3EE4C5A4]
module PCItoPCI best [3EE4C9EC]
module fcp best [3EE4CB3B]
module fcparray best [3EE4CB3B]
module fcpdev best [3EE4CB3B]
*
* Swap entries
*
*
* Dump entries
*
*
* Driver binding entries
*
*
* Tunables entries
*
tunable timezone 420
tunable vxfs_ifree_timelag 3600000
tunable bufpages 2048
tunable dbc_max_pct 1
tunable dbc_min_pct 1
tunable ksi_alloc_max 8192
tunable disksort_seconds 1
tunable max_async_ports 400
tunable maxfiles_lim 2048
tunable maxuprc 1000
tunable maxvgs 32
tunable nclist 512
tunable nfile 16384
tunable nkthread 2064
tunable nproc 1024
tunable scsi_max_qdepth 36
tunable secure_sid_scripts 0
tunable shmmax 0x1700000000
tunable shmmni 512
tunable swapmem_on 0
tunable swchunk 65536
tunable timeslice 1
tunable unlockable_mem 1
tunable nstrpty 60
tunable vxfs_bc_bufhwm 6144
```

## C.2 Oracle10i Database Standard Edition Parameters

## Config/Server/dbtune.ver

```
compatible = 10.1.0.0.0
db_name = tpcc
control_files = $tpcc_disks_location/control_001
#parallel_max_servers = 100
parallel_max_servers = 30
recovery_parallelism = 4
db_files = 524
db_block_size = 4096
db_cache_size = 2112M
db_2k_cache_size = 384M
db_8k_cache_size = 128M
db_16k_cache_size = 20800M
db_keep_cache_size = 54464M
db_recycle_cache_size  = 8768M
dml_locks = 500
log_buffer = 10485760
processes = 300
sessions = 300
transactions = 200
shared_pool_size = 384M
cursor_space_for_time = TRUE
_imu_pools = 200
undo_management = auto
undo_retention = 0
UNDO_TABLESPACE = undo_ts
_ksmg_granule_size = 67108864
_column_compression_factor = 175
fast_start_mttr_target  = 0
cursor_space_for_time = TRUE
disk_asynch_io = TRUE
log_checkpoint_interval  =    0
log_checkpoint_timeout   =    0
log_checkpoints_to_alert =    TRUE
parallel_execution_message_size = 4096
plsql_optimize_level  =       2
statistics_level = basic
timed_statistics = FALSE
_lgwr_async_io = FALSE
_cursor_cache_frame_bind_memory = TRUE
db_block_checksum = false
db_block_checking = false
lock_sga = true
```

## 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          client2

  MAXACCESSERS          12050           # 1024 or more
  MAXGTT   1024
  MAXSERVERS            18
  MAXSERVICES           75   # MAXSERVERS * #-of-services-each-server + 10( for BBL)
  MODEL      SHM
  LDBAL      Y

# During benchmark, don't want to scan too often.  In particular, while
# the client1s are stabilizing in virtual memory, we don't want to sanity
# scan; and if we do sanity scan, we want large timeouts, since the BRIDGE
# the BBL, the DBBL, and the client1s aren't getting much CPU time during that
# period.  Current settings:
#                    *  scan servers every 5 minutes (maximum  allowed by TUXEDO);
#                    *  wait 1 minute for sanity  responses (maximum allowed by TUXEDO);
#                    *  scan all the BBLs from DBBL every 30 minutes (want one scan in the
#                       audited results);
#                    *  timeout a blocking call after 5 minutes (the maximum).
                     SCANUNIT   60
                     SANITYSCAN                    5
                     DBBLWAIT   1
                     BBLQUERY   30
                     BLOCKTIME 5
#
#-----------------------------------------------------------------------
*MACHINES
#-----------------------------------------------------------------------
DEFAULT:
              TUXCONFIG="/project/tuxedo/confs/TUXconfig.client2"
                                   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?
client2            LMID=client2
       TUXCONFIG="/project/tuxedo/confs/TUXconfig.client2"
#-----------------------------------------------------------------
*GROUPS
#-----------------------------------------------------------------
group1          LMID=client2
                GRPNO=1
group2          LMID=client2
                GRPNO=2
group3          LMID=client2
                GRPNO=3

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

#-----------------------------------------------------------------------
*SERVERS
#-----------------------------------------------------------------------
#
# "--" is application-specific arguments to be passed to server
# "-n" is designed to specify server-id

service SRVGRP=group1
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n1"
       RQADDR=tpcc_1 SRVID=1

service SRVGRP=group1
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n2"
       RQADDR=tpcc_2 SRVID=2

service SRVGRP=group1
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n3"
       RQADDR=tpcc_3 SRVID=3

service SRVGRP=group1
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n4"
       RQADDR=tpcc_4 SRVID=4

service SRVGRP=group1
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n5"
       RQADDR=tpcc_5 SRVID=5

service SRVGRP=group2
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n6"
       RQADDR=tpcc_6 SRVID=6

service SRVGRP=group2
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n7"
       RQADDR=tpcc_7 SRVID=7

service SRVGRP=group2
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n8"
       RQADDR=tpcc_8 SRVID=8

service SRVGRP=group2
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n9"
       RQADDR=tpcc_9 SRVID=9

service SRVGRP=group2
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n10"
       RQADDR=tpcc_10 SRVID=10

service SRVGRP=group3
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n11"
       RQADDR=tpcc_11 SRVID=11

service SRVGRP=group3
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n12"
       RQADDR=tpcc_12 SRVID=12

service SRVGRP=group3
                     CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n13"
       RQADDR=tpcc_13 SRVID=13
#-----------------------------------------------------------------------
*SERVICES
#-----------------------------------------------------------------------
*ROUTING
#-----------------------------------------------------------------------

# This is a UBBconfig for a client1-server configuration.
#
# This UBBconfig requires settings for:
# SERVER_NAME CLIENT_NAME MASTER_NAME SERVER_ADDR CLIENT_ADDR
NODE_NAMES
# TLISTEN_PORT TBRIDGE_PORT
# In addition, it requires setting the things all UBBconfig.gens need:
#                    IPCKEY                            some decent IPCKEY, should be different for each
config
#                    ROOTDIR
#                    TUXCONFIG
#                    APPDIR
#                    ULOGDIR
#

#-----------------------------------------------------------------------
*RESOURCES
#-----------------------------------------------------------------------
                     IPCKEY        40001
                     PERM          0666
                     MASTER        client42

   MAXACCESSERS            12750            # 1024 or more
   MAXGTT    1024
   MAXSERVERS              35
   MAXSERVICES             160   # MAXSERVERS * #-of-services-each-server + 10( for BBL)
   MODEL       SHM
   LDBAL       Y

# During benchmark, don't want to scan too often.  In particular, while
# the client1s are stabilizing in virtual memory, we don't want to sanity
# scan; and if we do sanity scan, we want large timeouts, since the BRIDGE
# the BBL, the DBBL, and the client1s aren't getting much CPU time during that
# period.  Current settings:
#                    *  scan servers every 5 minutes (maximum allowed by TUXEDO);
#                    *  wait 1 minute for sanity  responses (maximum allowed by TUXEDO);
#                    *  scan all the BBLs from DBBL every 30 minutes (want one scan in the
#                       audited results);
#                    *  timeout a blocking call after 5 minutes (the maximum).
                     SCANUNIT   60
                     SANITYSCAN                    5
                     DBBLWAIT   1
                     BBLQUERY   30
                     BLOCKTIME 5
#
#-----------------------------------------------------------------------
*MACHINES
#-----------------------------------------------------------------------
DEFAULT:
              TUXCONFIG="/project/tuxedo/confs/TUXconfig.client42"
                                   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?
client42           LMID=client42
       TUXCONFIG="/project/tuxedo/confs/TUXconfig.client42"
#-----------------------------------------------------------------
*GROUPS
#-----------------------------------------------------------------
group1          LMID=client42
                GRPNO=1
group2          LMID=client42
                GRPNO=2
```

```
group3          LMID=client42
           GRPNO=3
group4          LMID=client42
           GRPNO=4
group5          LMID=client42
           GRPNO=5
group6          LMID=client42
           GRPNO=6

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

#----------------------------------------------------------------------
*SERVERS
#----------------------------------------------------------------------
#
# "--" is application-specific arguments to be passed to server
# "-n" is designed to specify server-id

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n1"
     RQADDR=tpcc_1 SRVID=1

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n2"
     RQADDR=tpcc_2 SRVID=2

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n3"
     RQADDR=tpcc_3 SRVID=3

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n4"
     RQADDR=tpcc_4 SRVID=4

service SRVGRP=group1
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n5"
     RQADDR=tpcc_5 SRVID=5

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n6"
     RQADDR=tpcc_6 SRVID=6

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n7"
     RQADDR=tpcc_7 SRVID=7

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n8"
     RQADDR=tpcc_8 SRVID=8

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n9"
     RQADDR=tpcc_9 SRVID=9

service SRVGRP=group2
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n10"
     RQADDR=tpcc_10 SRVID=10

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n11"
     RQADDR=tpcc_11 SRVID=11

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n12"
     RQADDR=tpcc_12 SRVID=12

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n13"
     RQADDR=tpcc_13 SRVID=13

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n14"
     RQADDR=tpcc_14 SRVID=14

service SRVGRP=group3
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n15"
     RQADDR=tpcc_15 SRVID=15

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n16"
     RQADDR=tpcc_16 SRVID=16

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n17"
     RQADDR=tpcc_17 SRVID=17

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n18"
     RQADDR=tpcc_18 SRVID=18

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n19"
     RQADDR=tpcc_19 SRVID=19

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n20"
     RQADDR=tpcc_20 SRVID=20

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n21"
     RQADDR=tpcc_21 SRVID=21

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n22"
     RQADDR=tpcc_22 SRVID=22

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n23"
     RQADDR=tpcc_23 SRVID=23

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n24"
     RQADDR=tpcc_24 SRVID=24

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n25"
     RQADDR=tpcc_25 SRVID=25

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n26"
     RQADDR=tpcc_26 SRVID=26

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n27"
     RQADDR=tpcc_27 SRVID=27

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n28"
     RQADDR=tpcc_28 SRVID=28

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n29"
     RQADDR=tpcc_29 SRVID=29

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC -- -n30"
     RQADDR=tpcc_30 SRVID=30
#----------------------------------------------------------------------
*SERVICES
#----------------------------------------------------------------------
*ROUTING
#----------------------------------------------------------------------
```

# Appendix D RTE Configuration

This appendix lists RTE input parameters and code fragments used to generate each transaction input file, to demonstrate the RTE was configured to generate transaction input data as specified in *Clause 2* of the specification.

## D.1 Field Value Generation

## Source/src/driver/generate.c

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 2002/12/10 14:35:33 $

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

#include <stdio.h>
#include <values.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <fcntl.h>
#include <signal.h>
#include <math.h>

#include "shm_lookup.h"
#include "random.h"

#include <time.h>

int CLAST_CONST_C = 208;
int CID_CONST_C =  37;
int IID_CONST_C = 75;

int trans_type = 0;   /* type of transaction 0 == all */

extern ID warehouse;
extern ID district;

neworder_gen(t)
   neworder_trans *t;
   {
   int i;

   t->W_ID = warehouse;

   t->D_ID = RandomNumber(1, no_dist_pw);
   t->C_ID = NURandomNumber( 1023, 1, no_cust_pd, CID_CONST_C);

   t->O_OL_CNT = RandomNumber(5, 15);

   for (i=0; i<t->O_OL_CNT; i++)
      {
      t->item[i].OL_I_ID = NURandomNumber(8191, 1, no_item, IID_CONST_C);
      t->item[i].OL_SUPPLY_W_ID = RandomWarehouse(warehouse, scale, 1);
      t->item[i].OL_QUANTITY = RandomNumber(1, 10);
      }
   /* Zero out the non-used items as the oracle driver does. */
   for (; i< 15; i++)
      {
      t->item[i].OL_I_ID = 0;
      t->item[i].OL_SUPPLY_W_ID = 0;
      t->item[i].OL_QUANTITY = 0;
      }


   /* 1% of transactions roll back. Give the last order line a bad item */
   if (RandomNumber(1, 100) == 1)
      t->item[t->O_OL_CNT - 1].OL_I_ID = -1;
   }

payment_gen(t)
   payment_trans *t;
   {

   /* home warehouse is fixed */
   t->W_ID = warehouse;

   /* Random district */
   t->D_ID = RandomNumber(1, no_dist_pw);

   /* Customer is from remote warehouse and district 15% of the time */
   t->C_W_ID = RandomWarehouse(warehouse, scale, 15);
   if (t->C_W_ID == t->W_ID)
      t->C_D_ID = t->D_ID;
   else
      t->C_D_ID = RandomNumber(1, no_dist_pw);

   /* by name 60% of the time */
   t->byname = RandomNumber(1, 100) <= 60;
   if (t->byname)
      LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1, CLAST_CONST_C),
                                       t->C_LAST);
   else
      t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);

   /* amount is random from [1.00..5,000.00] */
   t->H_AMOUNT = RandomNumber(100, 500000);

   }

ordstat_gen(t)
   ordstat_trans *t;
   {

   /* home warehouse is fixed */
   t->W_ID = warehouse;

   /* district is randomly selected from warehouse */
   t->D_ID = RandomNumber(1, no_dist_pw);

   /* by name 60% of the time */
   t->byname = RandomNumber(1, 100) <= 60;
   if (t->byname)
      LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1, CLAST_CONST_C),
                                       t->C_LAST);
   else
      t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);
   }

delivery_gen(t)
   delivery_trans *t;
   {
   t->W_ID = warehouse;
   t->O_CARRIER_ID = Random Number(1,10);
   }

stocklev_gen(t)
   stocklev_trans *t;
   {
   t->W_ID = warehouse;
   t->D_ID = district;
   t->threshold = RandomNumber(10, 20);
   }

int get_trans_type()
/***************************************************************************
 * get_trans_type selects a transaction according to the weighted average
 * For TPC-C rev 3.0 and less and TPC-C rev 3.2 this is:
 *       new-order :  ???
 *       payment   : 43.0%
 *       order stat:  4.0%
 *       delivery  :  4.0%
 *       stock     :  4.0%
 ***************************************************************************/
   {
   static double weight[] = { 0.0,  0.0, .4301, .0401, .0402, .0401};
   double drand48();
   int type;
   double r;

   /* choose a random number between 0.0 and 1.0 */
   if (trans_type == 0) {
#ifdef USE_DRAND48
      r = drand48();
#else
      r = randy();
#endif

      /*
       * select one of STOCKLEV, DELIVERY, ORDSTAT and PAYMENT
       * based on weight
       */
      for (type = STOCKLEV;  type > NEWORDER;  type--) {
         r -= weight[type];
         if (r < 0) break;
      }
   } else if (trans_type > 0) {
                 /* user wants only a certain type (say all stocklevel) so do that
      instead */
                 type = trans_type;
} else {
                  /* Trans type is less than zero, so this means exclude only
                     the selected type */
   for (type = STOCKLEV;  type > NEWORDER;  type--) {
       r -= weight[type];
                      if (-trans_type == type) { continue; }
       if (r < 0) break;
   }
                  if (-trans_type == NEWORDER &&
                                      type == NEWORDER) { type = PAYMENT; }
}
/* return the value of the selected card, or NEWORDER if none selected */
return type;
}
```

# Appendix E Disk Storage

**The calculations used to determine the storage requirements for the 8 hours logical log and the 60-day space calculations are contained in this appendix.**

**The calculations 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 29.73 minutes. Each log file is 21610MB.**

**So, to run for 8 hours, we need:**

**( (8 * 60) / 29.73 ) * 21610 / 1024 = 340.72GB (must be mirrored)**

**The log disk array has a capacity of 491.14GB (mirrored), 96GB of which was allocated to swap leaving 395.14GB available for recovery log.**

**The calculation for 60 day space yields 4,057.94GB. The 8 data disk arrays have a capacity of 491.14GB (mirrored) each for a total of 3,929.12GB. The difference, 128.82GB, is provided for by 2 internal SCSI disks with a capacity of 71.13GB each (not mirrored).**

**TPC-C 60-Day Space Requirements**

| | |
|---|---|
| **TPM** | 132000 |
| **Warehouses** | 10440 |

| SEGMENT | TYPE | TSPACE | BLOCKS | FIVE_PCT | DAILY_GROWTH | BLOCK SIZE | TOTAL in MB |
|---|---|---|---|---|---|---|---|
| CUSTCLUSTER | CLUSTER | CUST_0 | 42,725,760 | 2,136,288 | 0 | 4,096 | 175,242 |
| DISTCLUSTER | CLUSTER | DIST_0 | 128,000 | 6,400 | 0 | 2,048 | 263 |
| HIST | TABLE | HIST_0 | 4,916,348 | 0 | 994,572 | 4,096 | 23,090 |
| ICUST1 | INDEX | ICUST1_0 | 445,110 | 22,256 | 0 | 16,384 | 7,303 |
| ICUST2 | INDEX | ICUST2_0 | 981,648 | 49,082 | 0 | 16,384 | 16,105 |
| IDIST | INDEX | IDIST_0 | 1,296 | 65 | 0 | 2,048 | 3 |
| IITEM | INDEX | IITEM_0 | 5,120 | 256 | 0 | 2,048 | 11 |
| IORDR2 | INDEX | IORD2_0 | 700,146 | 35,007 | 0 | 16,384 | 11,487 |
| ISTOK | INDEX | ISTOK_0 | 1,315,080 | 65,754 | 0 | 16,384 | 21,576 |
| ITEMCLUSTER | CLUSTER | ITEM_0 | 10,240 | 512 | 0 | 2,048 | 21 |
| IWARE | INDEX | IWARE_0 | 108 | 5 | 0 | 2,048 | 0 |
| NORDCLUSTER | TABLE | NORD_0 | 691,516 | 34,576 | 0 | 4,096 | 2,836 |
| ORDRCLUSTER | TABLE | ORDR_0 | 17,497,311 | 0 | 3,539,686 | 16,384 | 328,703 |
| STOKCLUSTER | CLUSTER | STOK_0 | 85,451,520 | 4,272,576 | 0 | 4,096 | 350,485 |
| SYSTEM | SYS | SYS | 102,400 | 0 | 0 | 4,096 | 400 |
| WARECLUSTER | CLUSTER | WARE_0 | 12,800 | 640 | 0 | 2,048 | 26 |
| SYSAUX | SYS | SYSAUX | 30,720 | 0 | | 4,096 | 120 |
| Benchmark stats | SYS | SYS | 512,000 | 0 | 0 | 4,096 | 2,000 |
| Undo | SYS | UNDO_TS | 2,035,200 | 0 | 0 | 8,192 | 15,900 |
| **Total** | | | | | | | 955,569 |

| | | |
|---|---|---|
| Dynamic space | 292,600 | Initial MB for (History+Orders) |
| Static space | 603,777 | Initial Blocks + 5% - Dynamic |
| Daily Growth | 59,193 | Total Dynamic [(calc. as (Initial Blocks)*tpmC/(WHS*62.5)] |
| Daily Spread | 0 | Oracle may be configured so that daily spread is 0 |
| 60-day space (MB) | 4,155,335 | Static + 60*(Daily Growth+Daily Spread) |
| 60-day (GB) | 4,057.94 | Excludes OS, Paging and RDBMS Logs |
| 8-hour log (GB) | 340.72 | RDBMS Logs |
| Server swap (GB) | 96.00 | OS: Paging |
| Server OS (GB) | 16.00 | OS: UNIX File System |
| **Total Space Needed** | 4,510.67 | GB |

| **Priced-Sytem Configuration** | **Size in MB after RAID 1 redundency** | **Quantity** | **Total (GB)** |
|---|---|---|---|
| VA7110 with 30 36.4-GB disk drives in RAID 1 mode | 502,927 | 8 | 3,929.12 |
| VA7100 with 15 73-GB disk drives in RAID 1 mode | 502,927 | 1 | 491.14 |
| Internal 73GB drives | 72,837 | 3 | 213.39 |
| **Total Storage in Priced System (GB)** | | | 4,633.65 |

| TSPACE NAME | BLOCKS ALLOCATED | BLOCK SIZE | INITIAL STATIC | INITIAL DYNAMIC | 8-HOUR REQUIRED | OVERSIZE (BLOCKS) |
|---|---|---|---|---|---|---|
| CUSTCLUSTER | 85478400 | 4096 | 44862048 | 44862048 | 0 | 40616352 |
| DISTCLUSTER | 274944 | 2048 | 134400 | 134400 | 0 | 140544 |
| HIST | 9369600 | 4096 | 5910920 | 0 | 4916348 | 3458680 |
| ICUST1 | 632320 | 16384 | 467366 | 467366 | 0 | 164954 |
| ICUST2 | 1264640 | 16384 | 1030730 | 1030730 | 0 | 233910 |
| IDIST | 27648 | 2048 | 1361 | 1361 | 0 | 26287 |
| IITEM | 30720 | 2048 | 5376 | 5376 | 0 | 25344 |
| IORDR2 | 1264640 | 16384 | 735153 | 735153 | 0 | 529487 |
| ISTOK | 1792000 | 16384 | 1380834 | 1380834 | 0 | 411166 |
| ITEMCLUSTER | 30720 | 2048 | 10752 | 10752 | 0 | 19968 |
| IWARE | 7168 | 2048 | 113 | 113 | 0 | 7055 |
| NORDCLUSTER | 893440 | 4096 | 726092 | 726092 | 0 | 167348 |
| ORDRCLUSTER | 21369664 | 16384 | 21036997 | 0 | 17497311 | 332667 |
| STOKCLUSTER | 118041600 | 4096 | 89724096 | 89724096 | 0 | 28317504 |
| SYSTEM | 102400 | 4096 | 102400 | 102400 | 0 | 0 |
| WARECLUSTER | 27648 | 2048 | 13440 | 13440 | 0 | 14208 |
| Benchmark stats | 512000 | 4096 | 512000 | 512000 | 0 | 0 |
| SYSAUX | 30720 | 4096 | 30720 | 30720 | 0 | 0 |
| Undo | 2035200 | 8192 | 2035200 | 2035200 | 0 | 0 |

## Appendix F Price Quotes

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

Andreas Hotea
HP
Cupertino, CA 95014
July 30, 2003

**HP Unix Sales Development**
**19111 Pruneridge Avenue**
**Cupertino, CA 95014**
**(408) 447-2320**

# hp server rx5670

TPC-C Rev 5

Report Date: July 30, 2003

| Description | Part Number | Brand | Price Key | US List Price | Qty | Price | 3Year Main.Price |
|---|---|---|---|---|---|---|---|
| **Server Hardware** | | | | | | | |
| hp server rx5670 with 1.5GHz processor | A6838B | | 1 | 26,494 | 1 | 26,494 | |
| 1.5GHz Processor with 6MB Cache | A9810A | | 1 | 8,250 | 3 | 24,750 | |
| 3 Year Support Price (Hardware and Software) | | | 1 | | | | 15,259 |
| 8GB PC2100 DDR-SDRAM Memory (4x2GB DIMMs | A6835A | | 1 | 16,000 | 12 | 192,000 | |
| Memory Carrier Board for rx5670 | A6747A | | 1 | 1,981 | 2 | 3,962 | |
| 73GB 15K RPM Ultra320 SCSI Internal Drive (incl. 10 | A9760A | | 1 | 1,362 | 5 | 6,810 | |
| DVD ROM | A5557B | | 1 | 450 | 1 | 450 | |
| Sytstem Console | C1099A | | 1 | 550 | 1 | 550 | |
| 2GB Fibre Channel Adapter | A6795A | | 1 | 2,240 | 3 | 6,720 | |
| | | | | **Subtotal** | | **261,736** | **15,259** |
| **Server Software** | | | | | | | |
| HPUX 11i, V2 Foundation Operating Environment | B9429AC | | 1 | 2,370 | 4 | 9,480 | |
| Foundation Operating Environment Media Kit | B9106A, Opt OD1 | | 1 | 199 | 1 | 199 | |
| | | | | **Subtotal** | | **9,679** | **0** |
| **Storage** | | | | | | | |
| Rack System/E R3000 XR UPS | J4367A | | 1 | 1,948 | 4 | 7,792 | |
| Surestore VA 7110 w/dual controllers 512MB cache | A7294AZ | | 1 | 50,880 | 9 | 457,920 | |
| 3 Year Support Price | | | | | | | 38,714 |
| Disk System 2405 with dual 2GB link cards | A6250AZ | | 1 | 6,595 | 8 | 52,760 | |
| 36GB 15K RPM FC HDD. | A6193A, Opt 0D1 | | 1 | 1,349 | 240 | 323,760 | |
| 36GB 15K RPM FC HDD. (10% spare) | A6193A, Opt 0D1 | | 1 | 1,349 | 24 | 32,376 | |
| 73GB 10K RPM FC HDD. | A6194A, Opt 0D1 | | 1 | 2,019 | 15 | 30,285 | |
| 73GB 10K RPM FC HDD. (10% spare) | A6194A, Opt 0D1 | | 1 | 2,019 | 2 | 3,029 | |
| 2 meter LC Fibre Optic Cable | C7524A | | 1 | 215 | 24 | 5,160 | |
| 16 meter LC/SC Fibre Optic Cable | C7525A | | 1 | 260 | 3 | 780 | |
| HP FC 1GB/2GB Entry Switch 8B, Field Rack | A7346A | | 1 | 6,599 | 2 | 13,198 | |
| HP9000 Std. Rack System E41 | A4902A | | 1 | 1,910 | 3 | 5,730 | |
| Modular Power Dist. | A5137AZ | | 1 | 145 | 12 | 1,740 | |
| 200-240 Volts Power Option | A5137AZ, Opt AW4 | | 1 | 94 | 12 | 1,128 | |
| | | | | **Subtotal** | | **935,658** | **38,714** |
| **Client Hardware** | | | | | | | |
| HP server rp2470 | A6890A | | 1 | 1,865 | 9 | 16,785 | |
| 750Mhz PA-RISC 8700 CPU | A6892A | | 1 | 4,500 | 9 | 40,500 | |
| 3 Year Support Price (Hardware and Software) | | | | | | | 39,987 |
| 36GB 15K HotPlug Ultra 160 SCSI Internal Disk | A6948A | | 1 | 1,298 | 9 | 11,682 | |
| 2GB Memory Module | A6114A | | 1 | 3,400 | 9 | 30,600 | |
| 1GB Memory Module | A5841A | | 1 | 1,900 | 9 | 17,100 | |
| HP-UX 11.i Sys Media, CD-ROM | B3920EA, Opt. OD1 | | 1 | 195 | 9 | 1,755 | |
| | | | | **Subtotal** | | **118,422** | **39,987** |
| **Client Software** | | | | | | | |
| HP C/ANSI C Compiler | B3901BA, Option AH0 | | 1 | 1,600 | 1 | 1,600 | 170 |
| | | | | **Subtotal** | | **1,600** | **170** |
| **User Connectivity** | | | | | | | |
| HP ProCurve Switch 4000M | J4121A | | 1 | 2,379 | 1 | 2.379 | 588 |
| HP ProCurve Switch 100/1000Base-T Mod. | J4115B | | 1 | 509 | 1 | 509 | |
| | | | | **Subtotal** | | **2,888** | **588** |
| HP's Large configuration Discount and Support Prepayment* | | | | | | (491,728) | (30,175) |
| *All discounts are based on US list prices and for similar quantities and configurations | | | | **Total** | | **838,254** | **64,543** |

Quote is valid for 90 days

‘**July 28, 2003**

| Product | Price | Quantity | Extended Price |
|---|---|---|---|
| **Oracle Database 10G Standard Edition, Processor 3 year term for 4 processors, Unlimited Users** | **$7,500** | **4** | **$30,000** |
| **Oracle Database Server Support Package for** 3 years | **$2,000** | **1** | **$6,000** |
| **Oracle Mandatory E-Business Discount** | | | **<$1,800>** |
| Oracle TOTAL | | | $34,200 |

Oracle pricing contact: MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com, (650) 506-2118

**July 29, 2003**

**Ms. Lucille Boushey**
**TPC-C Performance Project Manager**
**Hewlett Packard**
**408 447 7364**
**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 and 8.1. Please note that Tuxedo 8.1 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. . Note that a 5% discount will apply to total list price license purchases less than $100,000 (for instance 11 Tier 1 servers; RP2470 Server 1 cpu configuration– 9 * 1,200 = $10,800 - would be eligible for a 5% discount). Support is not discountable and is priced at $252 per server for 24x7 support. This quote is valid for 60 days from the date of this letter.

### A.1.1 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, and 8.1. 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 and support fees.

**Very Truly Yours,**

Robert J. Gieringer

**Rob Gieringer,**
**Worldwide Pricing Manager**

### A.1.1.1 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 | | | | | | |
| **HP/UX 9.X;10.X** | **Uni-processor Workstation<br><br>B Class - 132/180/2000<br><br>C Class (3000/3600 / 3700)** | **9000/E25<br>9000/E35<br>9000/E45<br>9000/E55<br>9000/G30<br>9000/G40<br>9000/A180<br>9000/A180C<br>9000 /A400<br>RP2470 – 1 CPU<br>RP2430** | **9000/G50<br>9000/G60<br>Multi-Processor Workstations<br>J Class (J282/J2240/J5600/J6000/J6700)<br>9000/R380,390<br>9000/D200,210 220/30/50/60/80<br>D310/20/30<br>D350/60/70/80<br>9000 /A500<br>9000 – L1000<br>9000 – R Class<br>RP2470 – 2 CPU** | **9000/H20, 30<br>9000/H40, 50<br>9000/I30, 40<br>9000/K1XX<br>9000 – L2000/L3000<br><br>9000/I50,60<br>9000/H60<br>9000/G70<br>9000/H70<br>9000/I70<br>9000/K2XX<br>9000/K3XX<br>9000/K4XX<br>9000/K5XX<br>N4xxx Series** | **9000/T500, T520,T600<br>1-16 CPUs<br>S-Class** | **9000/V series all models<br>X-Class<br><br>9000 Series - Superdome** |