**Hewlett-Packard Company**
**Network Server Division**

**HP NetServer LH Pro**
**Using Microsoft SQL 6.5 on Microsoft NT 4.0**

# TPC Benchmark$^{TM}$ C
# Full Disclosure Report

**First Edition**
**Submitted for review**
**December 16, 1996**

Printed in U.S.A. December 16, 1996

# Abstract

## Overview

This report documents the methodology and results of the TPC Benchmark $^{tm}$ C test conducted on the Hewlett-Packard Company on the HP NetServer LH Dual Pentium Pro. The tests were run in a client/server configuration using 2 HP Series 9000 E55s as clients. The operating system used for the benchmark was Microsoft NT Server 4.0 for the server and HP-UX 10.01 on the clients. The database was the Microsoft SQL Server v. 6.5.SP3. All tests were done in compliance with Revision 3.2 of the Transaction Processing Council's TPC Benchmark$^{tm}$ C Standard Specification.

Two standard TPC Benchmark$^{tm}$ C metrics, transactions per second (tpmC) and price per tpmC ($/tpmC) are reported and referred to in this document. The results from the tests are summarized below.

| Hardware | Software | Total System Cost | tpmC | $/tpmC | Availability Date |
|---|---|---|---|---|---|
| Hewlett-Packard NetServer LH dual Pentium Pro (2-way) | Microsoft NT Server v4.0 (O/S), Microsoft SQL Server v6.5.213 (database) | $430,835 | 3904.00 tpmC | $110 per tpmC | HW: Feb. 1, 1997<br>SW: Feb. 28, 1997 |

## Auditor

The results of the benchmark and test methodology used to produce the results were audited by Richard Gimarc of Performance Metrics, Inc and have fully met the TPC-C rev 3.2 specifications.

Additional copies of this Full Disclosure Report can be obtained from either the Transaction Processing Performance Council or Hewlett-Packard Company at the following address:

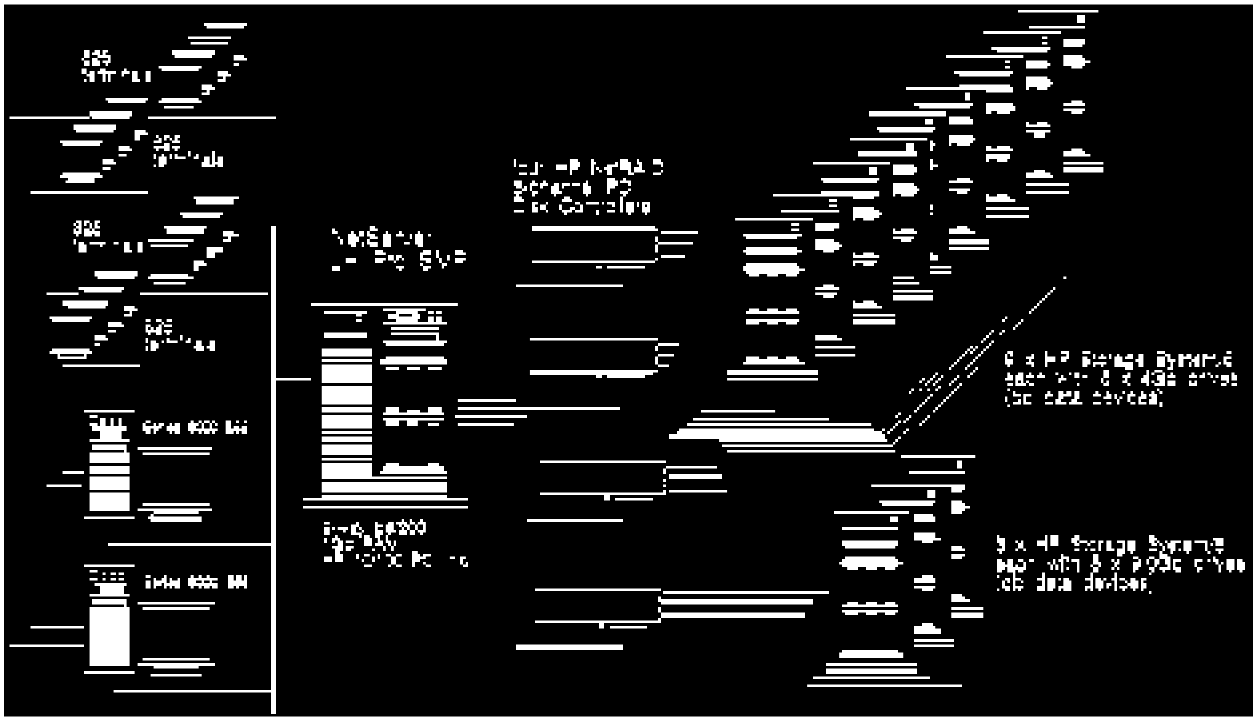
Transaction Processing Performance Council (TPC)
c/o Shanley Public Relations
777 North First Street, Suite 600
San Jose, CA 95112, USA
Phone: (408) 295-8894, (408) 295-9768 fax

or

Hewlett-Packard Company/NetWork Server Division
5301 Stevens Creek Blvd
Santa Clara, CA 95052-8059 USA

c/o Mary Johnson, bldg 53U

| Hewlett Packard | NetServer LH Pro<br>Client/Server Configuration | | | TPC-C Rev 3.2<br>Report Date: 12/16/96 |
|---|---|---|---|---|
| Total System Cost | TPC-C Throughput | | Price/Performance | Availability Date |
| $430,835 | 3904.00 tpmC | | $110 per tpmC | HW: Feb. 1, 1997<br>SW: Feb. 28, 1997 |
| Processors | Database Manager | Operating System(s) | Other Software | # Users |
| 2 Intel Pentium Pro$^{R}$ 200MHz | Microsoft SQL Server v. 6.5.SP3 | Microsoft NT Server 4.0, HPUX 10.01 | Microsoft C++ compiler, Visigenic ODBC Library (HPUX) TUXEDO TP Monitor | 3300 |



## System Components

**Serve**

> 1 HP NetServer LH Pro 6/200 SMP model M1, which includes 2 Intel Pentium Pro$^{R}$ 200MHz processors each with 256KB of 2nd level cache, 64Mb of ECC memory, 2 integrated Fast/Wide PCI SCSI-2 disk controllers, and one CDROM drive

> 4 256Mbyte NetServer Memory Modules (Simms)

> 4 HP NetRAID 3-ch PCI disk array controllers

> 49  HP 4.2GB Hot Swap disk drives (database and log)

> 15 HP 9GB Hot Swap disk drive (database)

> 1 HP 2GB Hot Swap disk drives (OS)

> 12 HP Storage System/6 external disk drive enclosures

**Clients**

> 2 100 MHz HP series 9000 model E55, each including one 1GB SCSI disk drive with HPUX v10.01, a MUX Personality card, 2 LAN/9000 Link, and 650Mb CDROM  drive

> 4 128Mb Series 9000 Memory Modules (2/client)

> 2 Series 9000 System Consoles (1/client)

| Hewlett Packard | NetServer LH Pro<br>Client/Server Configuration | TPC-C Rev 3.2<br>Report Date: 12/16/96 |
|---|---|---|

| HP NetServer LH Pro 6/200 - Microsoft NT 4.0 - Microsoft SQL 6.5 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Purchase Price | | | Support Price | | |
| | Product | Third Party | | Unit | | Extended | Unit | | 5 Year |
| | Number | Brand | Pricing | Price | Qty | Price | $/month | Qty | Support |
| ***Server Hardware*** | | | | | | | | | |
| HP NetServer LH Pro 6/200 2XM Array M1 | D3915A | HP | 3 | $10,870 | 1 | $10,870 | 63 | 1 | [illegible] |
| 3-year on-site, next day warranty | incl | | | | | | | | |
| 2 200MHz Pentium Pro processors | incl | | | | | | | | |
| 128 MB memory | incl | | | | | | | | |
| 8-disk Hot swap disk subsystem | incl | | | | | | | | |
| CD-ROM drive | incl | | | | | | | | |
| Integrated dual PCI Fast SCSI-2 controllers | incl | | | | | | | | |
| Integrated LAN controller | incl | | | | | | | | |
| 3.5-inch 1.44MB floppy drive | incl | | | | | | | | |
| 1 parallel, 2 serial ports | incl | | | | | | | | |
| Keyboard | incl | | | | | | | | |
| HP NetServer Navigator | incl | | | | | | | | |
| HP NetServer Assistant | incl | | | | | | | | |
| 256MByte ECC Module Memory Upgrade | D4020A | HP | 3 | $2,900 | 4 | $[illegible] | | | $0 |
| Support Pack for NetServer (upgrade to same day, 4 | H5518A | HP | | | | | | 1 | $[illegible] |
| 15" VGA Monitor | D2806A | HP | 3 | $[illegible] | 1 | $[illegible] | 7 | 1 | $158 |
| HP 10/100 PCI Network Adapter | J3171A | HP | | $135 | | $135 | | | $0 |
| ***Storage*** | | | | | | | | | |
| HP 3-Channel NetRAID PCI Array Controller + 10% spare | D4318A | HP | 3 | $2,[illegible] | 6 | $13,[illegible] | | | |
| Internal SCSI cable | 5182-[illegible] | HP | 3 | $[illegible] | 8 | $[illegible] | | | |
| External SCSI cable | 5182-6737 | HP | 3 | $[illegible] | 15 | $[illegible] | | | |
| HP Storage System/6 + 10% spare | [illegible] | HP | 3 | $[illegible] | [illegible] | $[illegible] | | | $0 |
| HP 4.2GB SCSI-2 hot swap drive + 10% spare | [illegible] | HP | | $[illegible] | 57 | $[illegible] | | | $0 |
| HP 2GB SCSI-2 hot swap drive + 10% spares | D4289A | HP | 3 | $[illegible] | 17 | $[illegible] | | | $0 |
| HP 2GB SCSI-2 hot swap drive | [illegible] | HP | | $970 | | $970 | 13 | 1 | $384 |
| HP SureStore Tape 6000 [illegible] Internal DAT | [illegible] | HP | 3 | $[illegible] | 1 | $[illegible] | 27 | 1 | $[illegible] |
| ***Server Software*** | | | | | | | | | |
| Microsoft Windows NT Server 4.0 | | Microsoft | 4 | $[illegible] | 1 | $[illegible] | | | |
| Microsoft SQL Server 6.5, unlimited license | | Microsoft | 4 | $2[illegible] | | $2[illegible] | | 1 | $1[illegible] |
| ***Communications*** | | | | | | | | | |
| Allied Telesyn 24-port 10BaseT Hub + 10% spare | [illegible] | Allied Telesys | [illegible] | $[illegible] | [illegible] | $[illegible] | | | |
| HP ANSI Compiler, [illegible] | [illegible] | HP | 3 | $[illegible] | | $[illegible] | 1,[illegible] | 1 | $[illegible] |
| ***Client Hardware*** | | | | | | | | | |
| HP Series 9000 model E55 [illegible] 2-User HP-UX [illegible] | [illegible] | HP | 3 | $[illegible] | 3 | $[illegible] | 62 | 6 | $[illegible] |
| [illegible] | [illegible] | HP | [illegible] | $[illegible] | [illegible] | $[illegible] | | | $0 |
| 1GB SE SCSI2 disk drive [illegible] HP-UX 10.01 | A[illegible] | HP | 3 | $970 | 3 | $[illegible] | 2 | 6 | $[illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | | [illegible] |

**NOTES:**

*** Support pack H5518A upgrades std NetServer 3yr Warranty to same day, 4-hr response, includes monitor and all internal components**
*** NetServer support for yrs 4-5 via HP contract support (order opt 02A for 24 months), includes all HP components**
*** Disk drives have external storage system warranty for yrs. 1-3, user opt 02L for yrs 4-5**
*** Series 9000 model E55 standard warranty is 1yr, next day, opt 0S3 upgrades to next day. Use service contract 02A for yrs 2-5. Includes all component**
*** Pricing key: 1=MicroAge, 2=Core Technology, 3=HP Corporate Price List, 4=Microsoft, 5=Micro Warehouse, 6=Visigenic Software, 7=BEA Systems.**

## Numerical Quantities Summary for HP NetServer LH Pro 6/200

**MQTH Computed Maximum Qualified Throughput:** 3904.30 tpmC

| Response Times (in seconds) | 90th %ile | Maximum | Average |
|---|---|---|---|
| New-Order | [illegible] | [illegible] | [illegible] |
| Payment | [illegible] | [illegible] | [illegible] |
| Order-Status | [illegible] | [illegible] | [illegible] |
| Delivery (interactive portion) | [illegible] | [illegible] | [illegible] |
| Delivery (deferred portion) | [illegible] | [illegible] | [illegible] |
| Stock-Level | [illegible] | [illegible] | [illegible] |
| Menu | - | [illegible] | [illegible] |

| Transaction Mix, in percent of total transactions | |
|---|---|
| New-Order | [illegible] |
| Payment | [illegible] |
| Order-Status | [illegible] |
| Delivery | [illegible] |
| Stock-Level | [illegible] |

| Keying/Think Times (in seconds) | Keying Time | | | Think Time | | |
|---|---|---|---|---|---|---|
| | Min. | Avg | Max | Min. | Avg | Max. |
| New-Order | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| Payment | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| Order-Status | | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| Delivery | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| Stock-Level | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |

| Test Duration | |
|---|---|
| Ramp-up time | [illegible] minutes |
| Measurement interval | [illegible] minutes |
| Transactions during measurement interval | [illegible] |
| Ramp-down time | [illegible] minutes |

| Checkpointing | |
|---|---|
| Number of checkpoints during measurement interval | 1 |
| Checkpoint interval | [illegible] minutes |

| Reproducibility Run | [illegible] tpmC | [illegible]% |
|---|---|---|

# *Preface*

## Document Structure

The contents of this report are determined by the TPC Benchmark C Standard Specification Revision 3.2, written and approved by the Transaction Processing Performance Council (TPC). This report documents the compliance of a system based on the HP NetServer LH Pro 6/200, Microsoft Windows NT v4.0 and Microsoft SQL Server 6.5.SP3 to this TPC Benchmark C specification.

The format of this report is based on this specification. Most sections of this report begins with the revelant specification requirements printed in italic type, immediately followed by the detail in plain type of how HP complied with the specification. Where extensive listings are required (such as listing of code), a note is included which references an appendix containing the listing.

## TPC Benchmark C Overview

The TPC Benchmark™ C (TPC-C) is an 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 on 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 any other environment are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-C should not be used as a substitute for a specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

.

## System Overview

The hardware configuration used in this TPC-C test was based on the Hewlett-Packard NetServer LH Pro 6/200 M1 Array server.  The full configuration was built by adding additional memory, additional disk adaptors and drives, and a network adaptor.  The operating system used was Microsoft's NT 4.0 and the database was Microsoft's SQL 6.5.SP3.

The architecture of the NetServer LH Pro was designed by Hewlett-Packard and based on the Intel Pentium Pro chip and associated chipset.   The LH used in this test was powered by two 200H z Intel Pentium Pro(R) chips, each with 256K bytes of SRAM 2nd level cache.

This configuration used 1Gbyte of HP 60-ns RAM. This was achieved by adding 4 256Mbytes DIMMs. This RAM was attachec to the system bus via a controller.

This configuration also used two SCSI-2 Fast/Wide PCI Disk controllers that were embedded onto the motherboard and 4 HP NetRaid 3-channel PCI Disk Array Controllers (DACs). These cards plugged into PCI slots on the motherboard, which are connected to a 33MHz PCI I/O bus. The PCI bus attaches directly to the P6 bus through a PCI and Memory Controller (PMC). In this way, the PCI bus masters have access to memory through the PMC.

One HP 2Gbyte SCSI-2 Hot Swap disk, one CDROM drive and 4 4G Hot Swap hard disks were attached to one of the embedded PCI SCSI controllers. This 2Gb drive was used exclusively for the Operating System (NT v4.0). Four 4Gbyte HP SCSI-2 Hot Swap hard disks were used exclusively for the database log. 45 HP 4Gbyte Hot Swap drives and 15 9G Hot Swap disk drives were equally distributed across 4 3-Channel HP NetRaid PCI Disk Array Controllers (DACs). Five Hot Swap disks were assigned per SCSI channel. Each channel was striped and the channels spanned using the HP NetRaid Utility. Controller write-back caching and read ahead were specifically disabled.

At the operating system, NT’s disk administrator shows 8 logical hard drives - the 2Gbyte SCSI-2 boot drive, the two mirrored pairs of 4Gbyte Hot Swap drives used for the log, 3 disks that are externalized by the NetRaid controllers as

60Gbyte logical disk drives, and one 135G disk externalized by the controller containing the 15 9G disks. Each of these 3 60Gbyte logical drives represent a hardware stripe set of 15 4Gbyte Hot Swap drives, created at the HP NetRaid level with channel spanning. The 135G disk is hardware striped in the same manner using the 9G disks.

The four logical drives were used to hold all the TPC database tables. This was done for maximum performance. Protection against data loss from a failed drive was achieved by normal database level recovery from the NT mirrored log drives.

This configuration also used one HP J3171A PCI network adaptor card, attached to the LH motherboard via the PCI bus.  This network adaptor supplied a 10BaseT network interface to the two HP-UX clients.  Each of the clients had 256Mbytes of RAM, one 1Gbyte SCSI hard disk, one HP Lan/9000 Link network adaptor, and was running HP-UX 10.01.  HP Monochrome VGA displays were used on the NetServer LH Pro and the HP System Console was used on each of the two clients.

# *Section 1.0 – General Items*

## 1.1 Test Sponsor

*A statement identifying the sponsor of the Benchmark and any other companies who have participated.*

The Network Server Division of the Hewlett-Packard Company was the test sponsor of this TPC Benchmark C.

## 1.2 Application Code and Definition Statements

*The application program must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input/output functions.*

The Section 3.0 entitled Clause 3 Related Items contains a brief discussion of the database design and loading. The database definiton statements, distribution across disk drives, loading scripts, and tables are provided in Appendix A- Database Generation

The program that implements the TPC Benchmark C translation and collects appropriate transaction statistics is referred to as the Remote Terminal Emulator (RTE) or Driver program. The Driver program is discussed in Section 7.0. The source code for this driver program is provided in Apprendix B - Source Code.

## 1.3 Parameter Settings

*Settings must be provided for all customer-tunable parameters and options which have been changed from the default found in actual products; including but not limited to:*

- *Database options*
- *Recover/commit options*
- *Consistency/locking options*
- *System parameter, application parameters, and configuration parameters.*

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

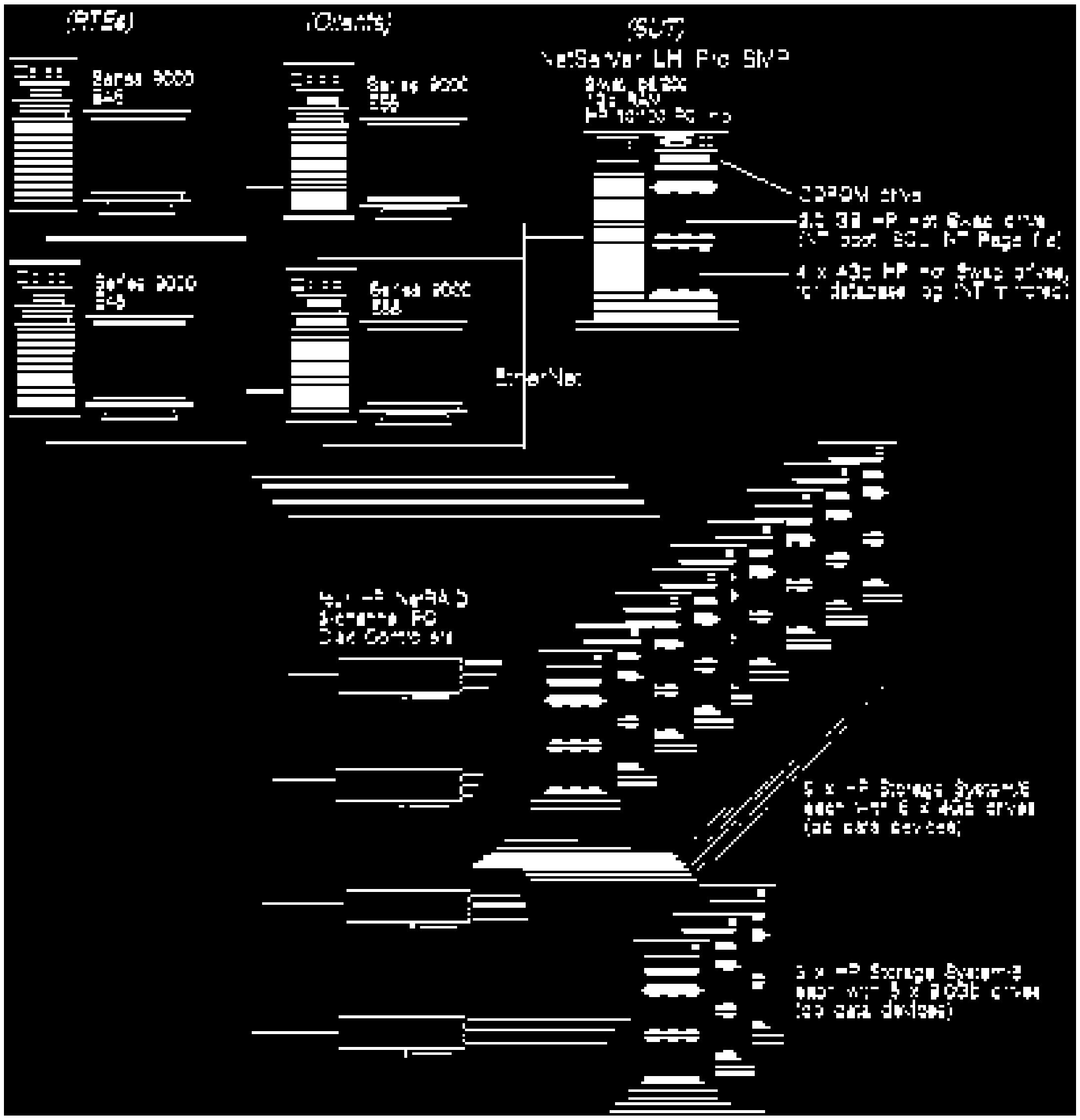
Appendix C contains all the database and operating system parameters used in this benchmark. Appendix D contains all the hardware configuration details.

## 1.4 Configuration Diagrams

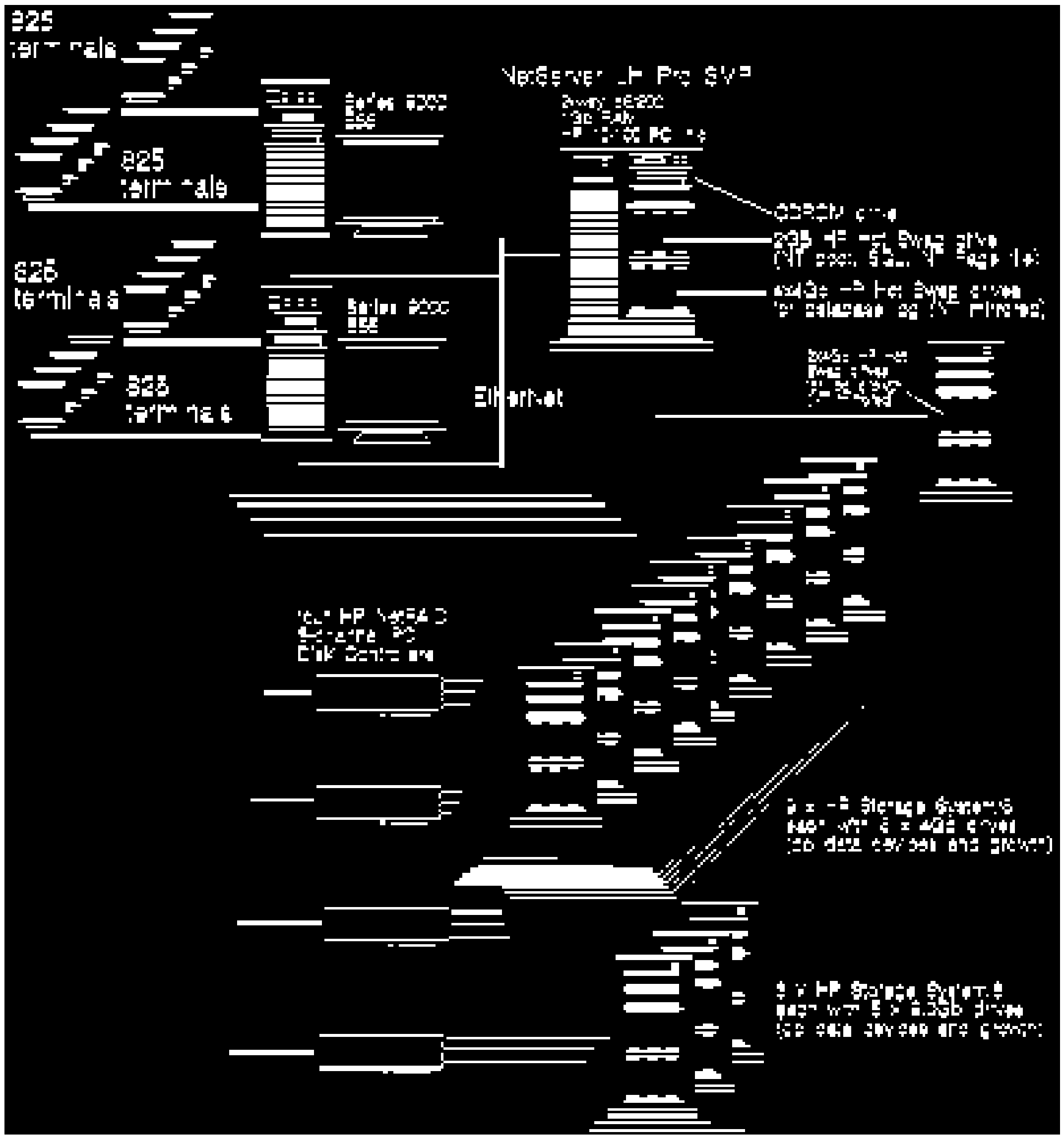
*Diagrams of both the measured priced system must be provided, accompanied by a description of the differences.*

Figure 1-1 and 1-2 respectively show the measured and priced full client/server configurations. The SUT in the measured system was idential to the priced one, except 4 extra 4GB Hot Swap disk were added for the log growth space.

**FIGURE 1-1: NetServer LH Pro - Measured Configuration**



## FIGURE 1-2: NetServer LX Pro - Priced Configuration



# *Section  2.0 – Clause 1 Related Items*

## 2.1 Table Definitions

*A listing must be provided for all table definitions statements and all other statements used to set up the database.*

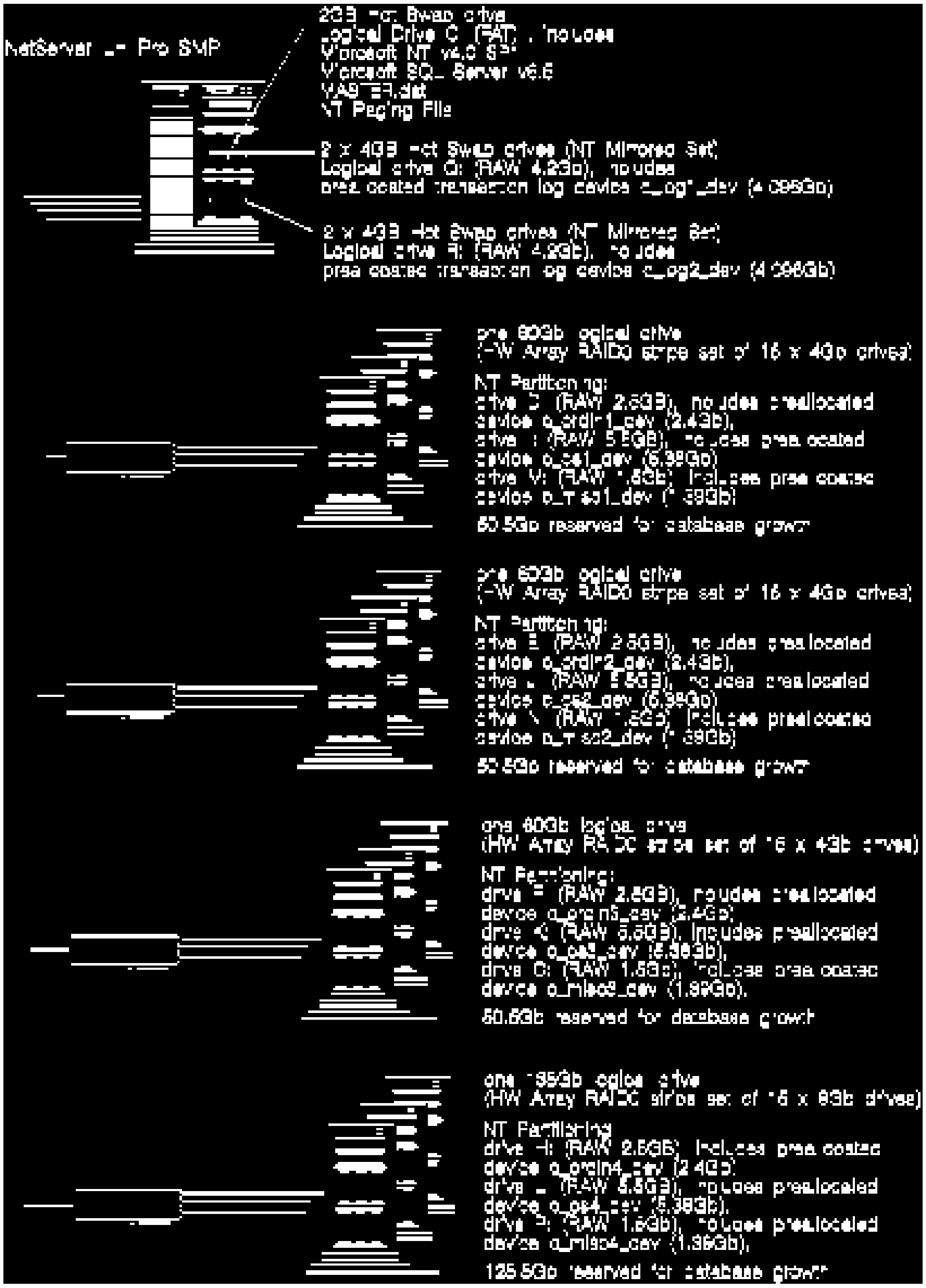
Appendix B contains the code used to define and load the database tables.

## 2.2 Physical Organization of the Database

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

The measured configuration used a total of one 2Gb, 49 4Gb, and 15 9Gb Hot Swap disk drives. Figure 3-1 below depicts the data distribution of the files across the hard drives of the HP NetServer LH Pro.

## FIGURE 3.1: HP NetServer 6/200 LH Pro Database Distribution


NetServer LH Pro SMP
2GB Hot Swap drive
Logical Drive C: (FAT), includes
Microsoft NT v4.0 SP1
Microsoft SQL Server v6.5
MASTER.dat
NT Paging File
2 x 4GB Hot Swap drives (NT Mirrored Set)
Logical drive Q: (RAW 4.23Gb), includes
preallocated transaction log device c_log1_dev (4.096Gb)
2 x 4GB Hot Swap drives (NT Mirrored Set)
Logical drive R: (RAW 4.23Gb), includes
preallocated transaction log device c_log2_dev (4.096Gb)
one 60Gb logical drive
(HW Array RAID0 stripe set of 16 x 4Gb drives)
NT Partitioning:
drive D: (RAW 2.6GB), includes preallocated
device c_ordln1_dev (2.4Gb),
drive I: (RAW 5.5GB), includes preallocated
device c_cs1_dev (6.39Gb)
drive M: (RAW 1.6Gb), includes preallocated
device c_misc1_dev (1.39Gb)
50.5Gb reserved for database growth
one 60Gb logical drive
(HW Array RAID0 stripe set of 16 x 4Gb drives)
NT Partitioning:
drive E: (RAW 2.6GB), includes preallocated
device c_ordln2_dev (2.4Gb),
drive J: (RAW 5.5GB), includes preallocated
device c_cs2_dev (6.39Gb)
drive N: (RAW 1.6Gb), includes preallocated
device c_misc2_dev (1.39Gb)
50.5Gb reserved for database growth
one 60Gb logical drive
(HW Array RAID0 stripe set of 16 x 4Gb drives)
NT Partitioning:
drive F: (RAW 2.6GB), includes preallocated
device c_ordln3_dev (2.4Gb),
drive K: (RAW 5.5GB), includes preallocated
device c_cs3_dev (5.39Gb),
drive O: (RAW 1.6Gb), includes preallocated
device c_misc3_dev (1.39Gb).
50.6Gb reserved for database growth
one 135Gb logical drive
(HW Array RAID0 stripe set of 16 x 9Gb drives)
NT Partitioning
drive H: (RAW 2.6GB), includes preallocated
device c_ordln4_dev (2.4Gb)
drive L: (RAW 5.5GB), includes preallocated
device c_cs4_dev (5.39Gb),
drive P: (RAW 1.6Gb), includes preallocated
device c_misc4_dev (1.39Gb),
125.5Gb reserved for database growth

## 2.3 Insert and Delete Operations

*It must be ascertained that inset and 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 minimum key value for these new rows.*

All insert and delete functions were fully operational and verified during the entire benchmark.

## 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.*

Partitioning was not used on any table.

## 2.5 Replication, Duplication or Additions

*Replication of tables, if used, must be dislosed. Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.*

No replications, duplications or additional attributes were used.

# Section 3.0 – Clause 2 Related Items

## 3.1 Random Number Generation

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

The library routine SRAND48 (3C) was used to seed the library routine DRAND48 (3C) which generated pseudo-random numbers using the well-known linear congruential algorithm and 48-bit integer arithmetic. Further information on SRAND48 (3C) and DRAND48 (3C) can be found in the HP-UX Reference Manual Vol. 3.

## 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-NetServer LC 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.

## 3.5 Transaction Statistics

Table 2.3 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 require.

**Table 3.1: Transaction Statistics**

| Type | Item | Value |
|---|---|---|
| New Order | Home warehouse items | 90.46% |
| | Remote warehouse items | 9.54% |
| | Rolled back transactions | 1.02% |
| | Average items per order | 9.98 |
| Payment | Home warehouse | 85.07% |
| | Remote Warehouse | 14.93 |
| | Non-primary key access | 60.26 |
| Order Status | Non primary key access | 59.74 |
| Delivery | Skipped transactions | 0 |
| Transaction Mix | New Order | 44.88% |
| | Payment | 43.00% |
| | Order Status | 4.04% |
| | Delivery | 4.06% |
| | Stock Level | 4.02% |

## 3.6 Queueing Mechanism

The queueing 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 asychronously.

# Section 4.0 – Clause 3 Related Items

## 4.1 Transaction System Properties (ACID Tests)

*Results of the ACID test must describe how the 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). The following subsections will define each of these properties and describe the series of tests that were performed by HP to demonstrate that the properties were met.

All of the specified ACID tests were performed on the HP NetServer LH Pro. A fully scaled database was used except for the durability tests of durable media failure. The test was performed on a database scaled to 10 warehouses, using the standard driving mechanism. However a fully scaled database under a full load would also pass this durability test.

## 4.2 Atomicity Tests

*The system under test (SUT) 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 have any effects on the data.*

### 4.2.1 COMMIT Transaction

*The following steps were done to demonstrate the COMMIT property of Atomicity:*

1. A row was randomly selected from the Warehouse, District and Customer tables, and the present balances noted

2. The standard payment transaction was started against the above identifiers using a known amount.

3. The transaction was committed and the rows were verified to contain the correct updated balances.

### 4.2.2 ROLLBACK Transaction

*The following steps were done to demonstrate the ROLLBACK property of Atomicity:*

1. A row was randomly selected from the Warehouse, District, Customer tables, and the present balances noted.

2. The standard payment transaction was started against the above identifiers using a known amount.

3. The transaction was rolled back and the rows were verified to contain the original balances.

## 4.3 Consistency Tests

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

To prove consistency, queries were issued to the database. The results of the queries verified that the database was consistent for all conditions as specified in clause 3.3.2.1 to 3.3.2.4.

The consistency tests were run before and after the performance run.

## 4.4 Isolation Tests

*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 mix of arbitrary transactions.*

We ran a total of nine isolation tests. Seven of these tests are detailed in the TPC-C specification (clause 3.4.2.1to 3.4.2.7). The additional two are to fully comply with the isolation requirements that are not directly specified in the TPC-C specification. These two tests are known as Phantom Protection One and Two. They demonstrate that the applications are protected from phantom inserts.

## 4.5 Durability Tests

*The tested system must guarantee the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed in clause 3.5.3.1, 3.5.3.2, and 3.5.3.3.*

There 3 types of failures were tested to ensure the durability of the database: Loss of Data drive, Loss of Log drive, and Loss of Memory test.

A fully scaled database was used for the Loss of Memory and the Loss of Log test while a 10 warehouse database was used for the Loss of Data test. With this exception of scaling, all other aspects of the configurations on the 10 warehouse database were identical to the fully scaled database configuration, including the use of the standard RTE drivers. Given this, the Loss of Data test would pass in a fully scaled database configuration.

### TESTING PROCEDURE AND RESULTS:

The following steps detail the testing procedure and results for all the three durability tests. Each test was done separately.

Step 1: Database was backed up.

Step 2: The total number of new orders was calculated and recorded. Consistency test #3 was run to show that the database was in a consistent state prior to the durability tests.

Step 3: The standard TPC-C benchmark was launched. For the Loss of Data test, the benchmark was run with 100 users. The transaction rate was monitored until the system was in steady state. During this time, the number of users in the benchmark run was verified. After this, a checkpoint was issued. An additional 3-minute run was performed.

Step 4:The failure was initiated. For the Loss of Data drive test, one HP 4Gb Hot Swap drive holding a portion of the database data was pulled out while the benchmark was running. For the Loss of Log drive test, one HP 4Gb Hot Swap drive holding a portion of the mirrored database log was pulled out while the benchmark was running. For the Loss of Memory test, the power switch on the NetServer LH was depressed (turning off the system) while the benchmark was running.

Step 5: The recovery process was performed.

For the loss of Data drive test, we then backed up the transaction log, and restored the combination of the initial back up (step 1) and the just-backed-up transaction log to bring it to the most recent consistent state.

For the loss of log test, as would be expected, NT produced an alert message informing us that one of the members of the mirror set has failed. The SUT slowed down for a brief period as it needs to alter its log-write destination to the primary drives of the mirror. These activities were transparent to the database server as we observed that it continued to run after the aforementioned slow-down period.

For the loss of memory test, we re-powered the system, and started the server. As we would have expected, the server performed the automatic recovery.

Step 6: We computed the total number of order transactions again, and the difference between it and the one measured in step two. We verified that this difference was the same as the total number of new order transactions recorded in the “success” file. This file records committed transactions on the clients. In addition, we reran the consistency test #3 to show the database was in a consistent state after the durability tests.

We sampled the after-failure database with those recorded in the “success” file. We chose the first, last and middle two transactions from the “success” file to sample the database.

# *Section 5.0 – Clause 4 Related Items*

## 5.1 Database Layout

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

The measured (tested) and priced system have identical controller configurations and only differ by 4 additional 4Gb Hot Swap disks added to the priced configuration to supply growth space for the log.

Both configurations used one of the SCSI-2 Fast/Wide PCI Disk controllers that were embedded onto the motherboard and 4 HP NetRAID 3-channel PCI Disk Array Controllers (DACs). These cards plugged into PCI slots on the motherboard.

One HP 2Gb Hot Swap Fast hard disk, four HP SCSI-2 4Gb Hot Swap hard disks and one CDROM drive were attached to the first (A) of the available embedded PCI SCSI controllers. The 2Gb drive was used for the Operating System (NT v4.0). For the measured configuration, four 4Gbyte HP SCSI-2 Hot Swap hard disks were attached to the same embedded PCI SCSI controller (A) and were used exclusively for the log.

For the priced configuration a total of 6 HP SCSI-2 Hot 4GB Hot Swap hard disks were used, to supply growth space for the log. 45 HP 4Gbyte Hot Swap drives are attached to 3 of the HP NetRAID PCI Disk Array controllers and 15 HP 9Gbyte Hot Swap drives are attached to the fourth controller. Five Hot Swap disks were placed in each HP Storage System 6. Each channel was striped using the NetRAID Utility. Controller write-back caching and read ahead were specifically disabled.

At the operating system, NT’s disk administrator shows 10 logical drives - the 2Gbyte SCSI-2 boot drive, the four 4Gbyte Hot Swap drives used for the mirrored log, 3 60 Gbyte logical drives and one 130 Gbyte logical drive. Each of these 60Gbyte logical drives represent a hardware stripe set of fifteen 4Gbyte Hot Swap drives, created at the DAC level spaning the three channels. The 130Gbyte drive is the same DAC configuration, except that the hard disks are 9Gbytes each. Protection against data loss from a failed drive was achieved by normal database level recovery from the log drives, which are mirrored. The preceeding Figure 3-1 depicts the data distribution of the files across the hard drives of the HP NetServer LH Pro.

## 5.2 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 over-scaled 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*

**Table 5.1: Number of Rows**

| Table | Occurrences |
|---|---|
| Warehouse | 360 |
| District | 3600 |
| Customer | 10800000 |
| History | 10800000 |
| Orders | 10800000 |
| New Orders | 3240000 |
| Order Line | 108000000 |
| Stock | 36000000 |
| Item | 100000 |

Rows 331 through 360 were deleted were for the benchmark runs.

## 5.3 180 Day Space

*Details of the 180 day space computations along with proof that the database is configured to sustain 8 hours of growth for the dynmic tables must be disclosed.*

**Transaction Log Space Requirements**

To calculate the space required to sustain the database log for 8 hours of growth at steady state, the following steps were followed:

1. The free space on the logfile was queried using **dbcc checktable(syslogs)**.

2. Transactions were run against the database with a full load of users.

3. The free space was again queried using **dbcc checktable(syslogs**)

4. The space used was calculated as the difference between the first and second query.

5. The number of NEW-ORDERS was verified from an RTE report covering the entire run.

6. The space used was divided by the number of NEW-ORDERS giving a space used per NEW-ORDER transaction.

7. The space used per transaction was multiplied by the measured tpmC rate times 480 minutes.

The result of the above steps yielded a requirement of 10.6GB (including mirror). to sustain the log for 8 hours. Space available on the transaction log was 12GB (including mirror), indicating enough storage was configured to sustain 8 hour growth.

The same methodology was used to calculate the growth requirements for the other dynamic tables Order, Order-Line and History.

The details of the 180 day growth calculation are shown in appendix D.

## 5.4 Type of Database Used

*A statement must be provided that describes 1) the data model implemented by DBMS used and 2) the database interface and access language*

Microsoft SQL Server 6.5 is a relational DBMS.

The interface was SQL Server stored procedures accessed with ODBC library calls embedded in C code.

## 5.5 Database Mapping

*The mapping of database partitions and replications must be described.*

The database was neither partitioned nor replicated.

# *Section 6.0 – Clause 5 Related Items*

## 6.1 Throughput

*Measured tpmC® must be reported.*

**Table 6.1: Throughput**

| | |
|---|---|
| tpmC® | 3904.00 |

## 6.2 ResponseTimes

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

**Table 6.2: Response Times**

| Type | Average | Maximum | 90th Percentile |
|---|---|---|---|
| New Order | 1.67 | 174.07 | 2.20 |
| Payment | 1.31 | 157.20 | 1.96 |
| Order-Status | 2.01 | 146.17 | 2.73 |
| Interactive Delivery | 0.12 | 0.29 | 0.21 |
| Deferred Delivery | 1.65 | 131.79 | 2.18 |
| Stock-Level | 3.83 | 130.58 | 5.26 |
| Menu | 0.01 | 0.13 | na |

## 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**

| Type | Minimum | Average | Maximum |
|---|---|---|---|
| New-Order | 18.01 | 18.02 | 18.08 |
| Payment | 3.01 | 3.02 | 3.07 |
| Order-Status | 2.01 | 2.02 | 2.08 |
| Interactive Delivery | 2.01 | 2.02 | 2.07 |
| Stock Level | 2.01 | 2.02 | 2.08 |

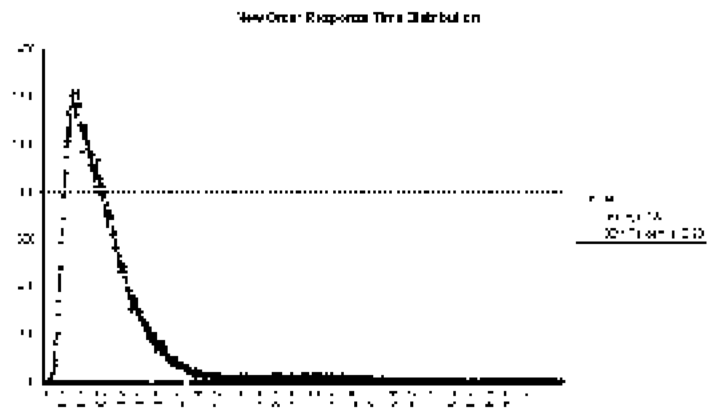
**Table 6.4: Think Times**

| Type | Minimum | Average | Maximum |
|---|---|---|---|
| New-Order | 0.01 | 12.17 | 133.63 |
| Payment | 0.01 | 12.21 | 141.12 |
| Order-Status | 0.01 | 10.34 | 83.30 |
| Interactive Delivery | 0.01 | 5.15 | 59.67 |
| Stock-Level | 0.01 | 5.17 | 49.04 |

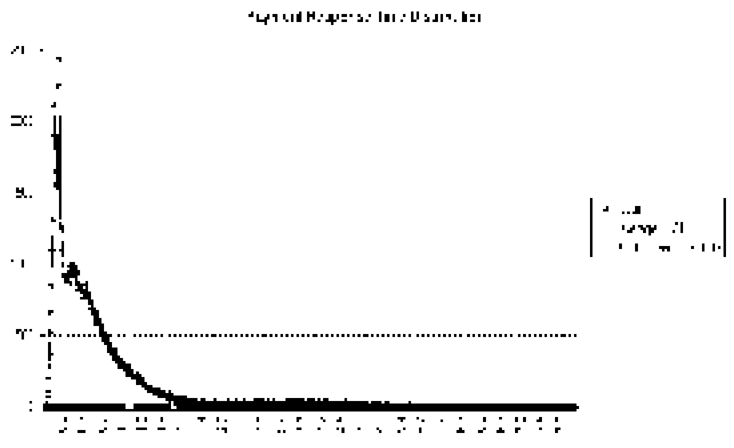
## 6.4 Response Time Frequency and

*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. Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type. Keying Time frequency distribution curves (see Clause 5.6.4) must be reported for each transaction type. A graph of throughput versus elapsed time (see Clause 5.6.5) must be reported for the New-Order transaction.*

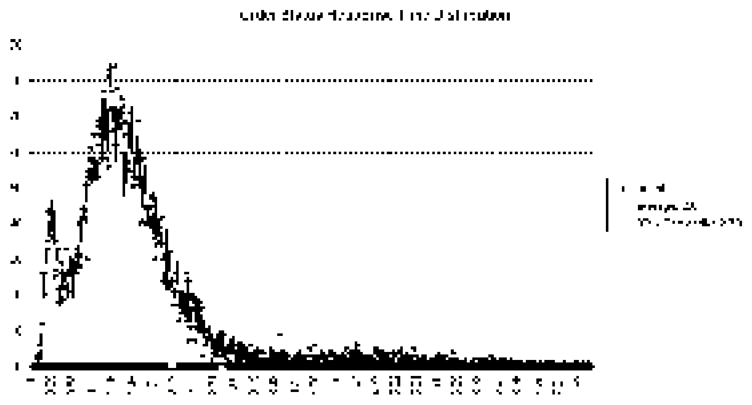
### 6.4.1 New Order Response Time



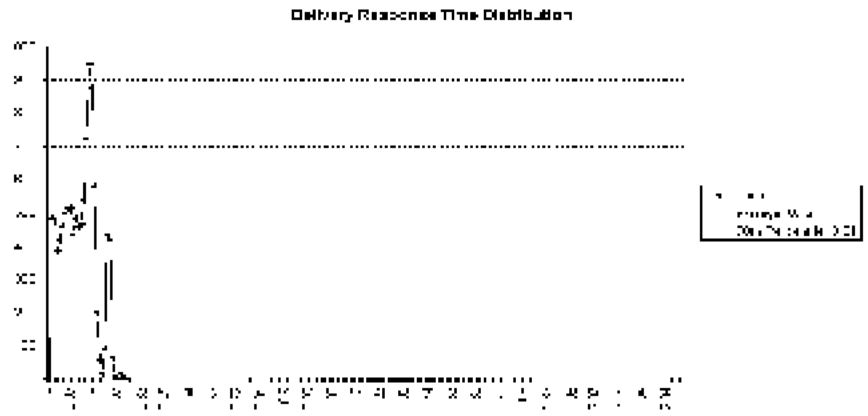
### 6.4.2 Payment Response Time Distribution



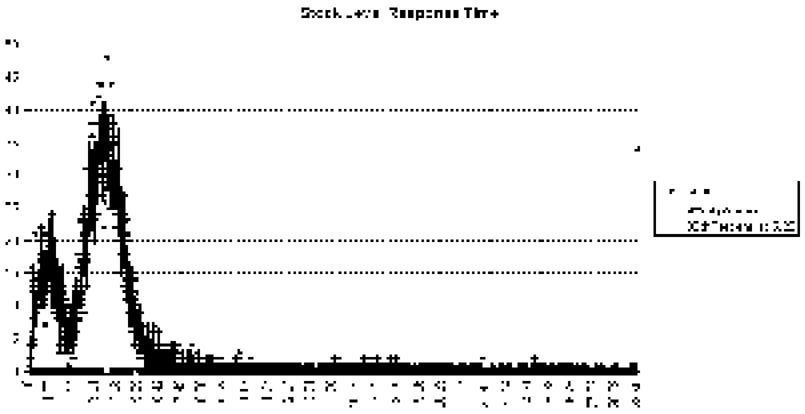
## 6.4.3 Order Status Response Time



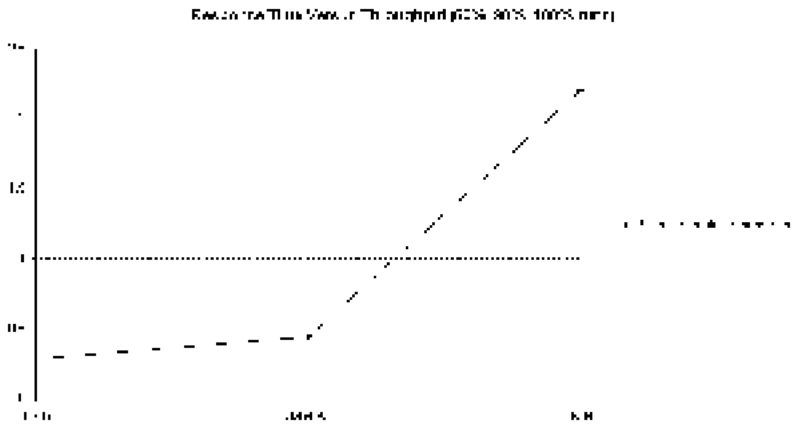
## 6.4.4 Delivery Response Time Distribution



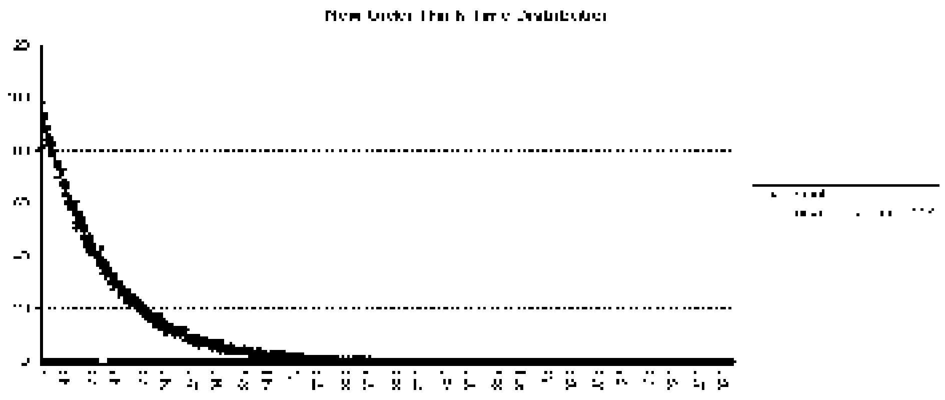
## 6.4.5 Stock Level Response Time



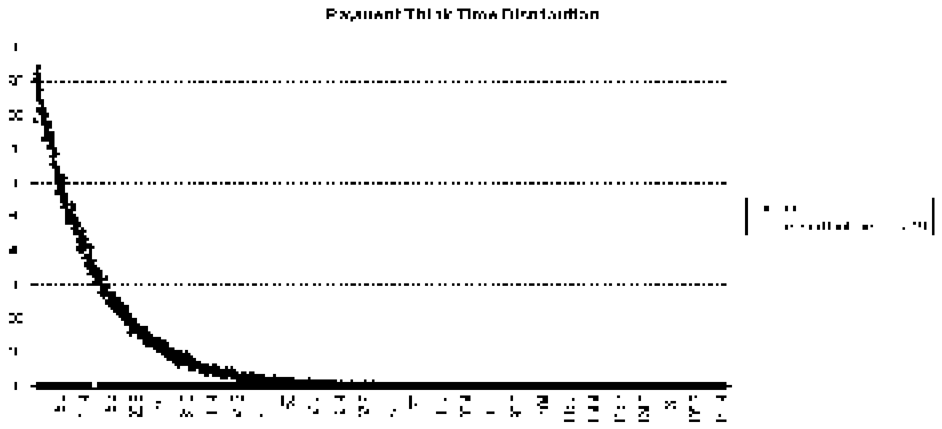
## 6.4.6 Response Time Versus Throughput



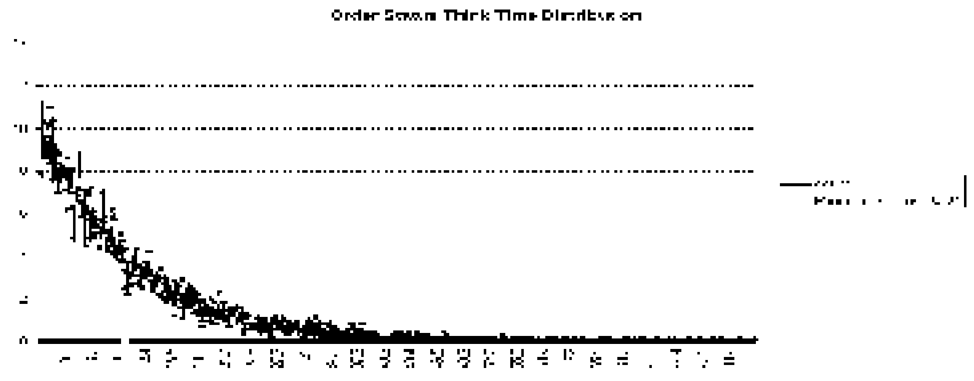
## 6.4.7 New Order Think Time Distribution



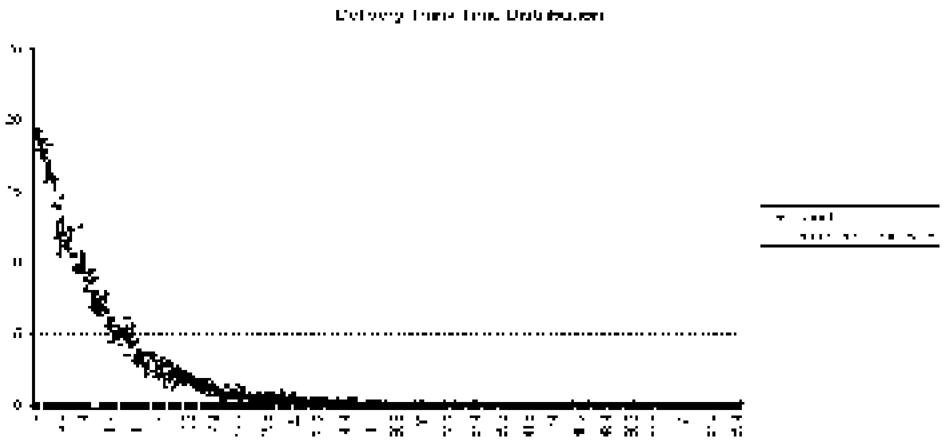
## 6.4.8 Payment Think Time Distribution



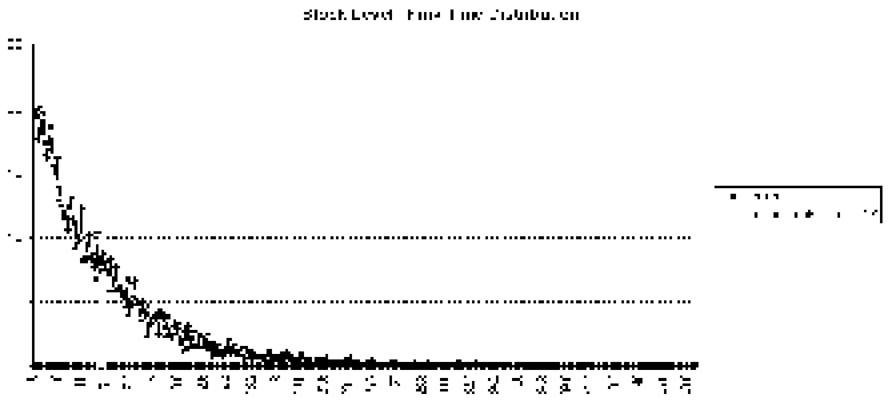
## 6.4.9 Order Status Think Time Distribution



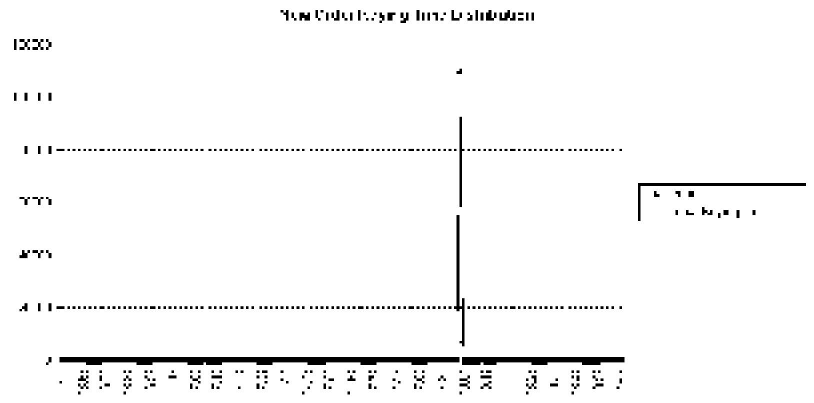
## 6.4.10 Delivery Think Time Distribution



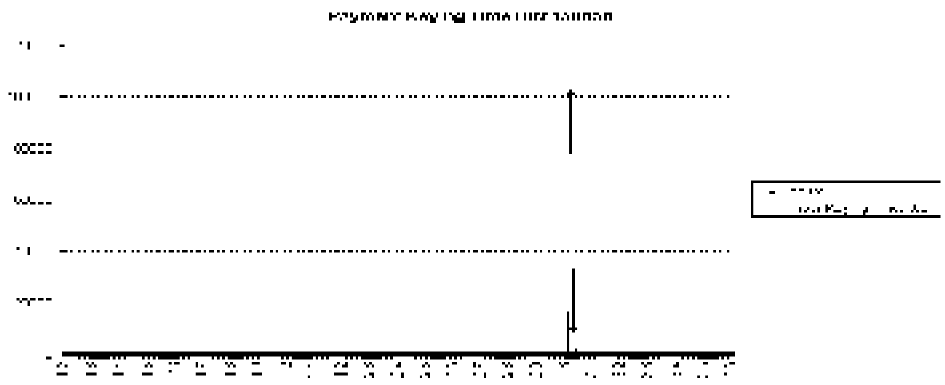
## 6.4.11 Stock Level Think Time Distribution



## 6.4.12 New Order Keying Time Distribution



### 6.4.13 Payment Keying Time Distribution

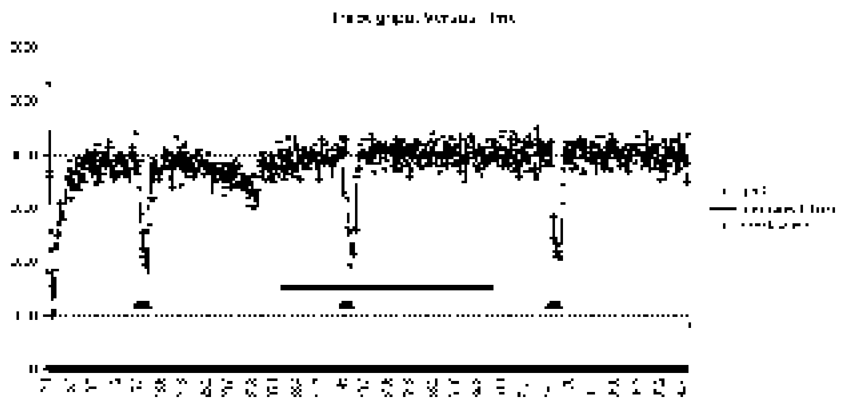


### 6.4.14 Order Status Keying Time Distribution

### 6.4.15 Delivery Keying Time Distribution

### 6.4.16 Stock Level Keying Time Distribution

## 6.4.17 Throughput Versus Time Distribution



## 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.*

The transaction throughput rate (tpmC®) and response time were relatively constant after the initial ‘ramp up’ period. The throughput and response time behavior were determined by examining data reported for each interval over the duration of the benchmark. Ramp up, steady state and ramp down regions are discernible in the graph (6.4.17).

## 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.*

### 6.6.1 Checkpoint

The checkpoint mechanism is an automatic means for guaranteeing that completed transactions are regularly written from SQL Server’s won disk cache to the database device. A checkpoint writes all “dirty pages”-cached pages that have been modified since the last checkpoint-to the database device.

### 6.6.2 Checkpoint Conditions

There are two types of checkpoints:

- Checkpoints that are executed automatically by SQL Server.
- Checkpoints that are forced by database owners of the SA with the CHECKPOINT statement.

Forcing dirty pages onto the database device means that all completed transactions are written out. By calling all completed transactions to be written out, the check point shortens the time it takes to recover, since the database pages are current and there are no transactions that need to be rolled forward.

### 6.6.3 Checkpoint Implementation

For each benchmark measurement after all users are active, the script **checkpoint_tpcc.sh** issues a checkpoint. A background process sleeps and performs another checkpoint every 30 minutes. The recovery interval (used to control the checkpoints executed automatically by SQL Server) is configured large enough that no other checkpoints occur during the measurement.

## 6.7 Reproducibility

*A description of the method used to determine the reproducibility of the measurement results.*

A second measurement achieved a throughput of 3902.60 tpmC® during a 30-minute, steady state interval.

## 6.8 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 30 minutes.

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

## 6.10 Transaction Mix

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

**Table 6.5: Transaction Mix**

| Type | Percentage |
|---|---|
| New-Order | 44.88% |
| Payment | 43.00% |
| Order-Status | 4.04% |
| Delivery | 4.06% |
| Stock-Level | 4.02% |

## 6.11 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 transaction must be disclosed. The percentage of remote order-lines entered per New-Order transaction must be disclosed. The percentage of selections made 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.*

Table 2.1 contains the required items.

## 6.12 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.*

Times in the following table are relative to the beginning of the driver-times phase of the test. The checkpoint interval is 30 minutes. The first checkpoint within the 30 minute measure interval was 1272 seconds from its start. In accord with *5.5.2.2*, there is no checkpoint within the “guard zones” 1800/4=450 seconds from the beginning and end of the measurement interval.

**Table 6.6: Checkpoints**

| Event | From (sec) | To (sec) | Duration (sec) |
|---|---|---|---|
| checkpoint | 900 | 1002 | 102 |
| measured interva | 2190 | 3990 | 1800 |
| checkpoint | 2701 | 2787 | 86 |
| checkpoint | 4501 | 4590 | 89 |

# *Section 7.0 – Clause 6 Related Items*

## 7.1 RTE description

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

The RTE (remote Terminal Emulator) on the driver system was developed at Hewlett Packard and is not commerically available.

For this instance of the TPC-C benchmark, two driver and two client systems were used. The drivers emulated 3300 users logged in to the clients. An overview of the benchmark software on the drivers, clients and server is shown in figure 7.1

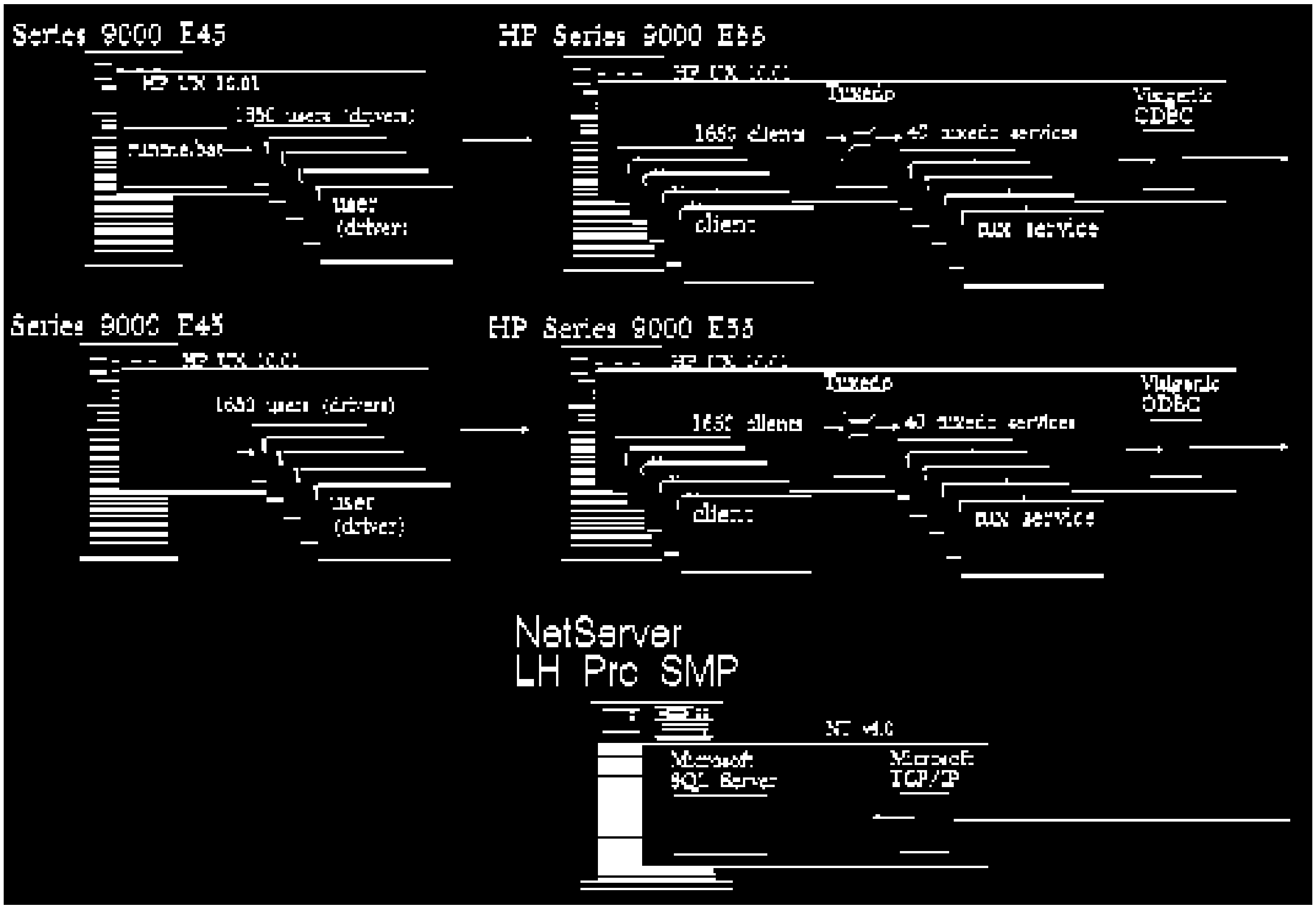
The benchmark is started with the RUNME command on the driver system. RUNME controls the overall execution of the benchmark. After reading a configuration file, RUNME starts the TUXEDO servers on the clients, collects pre-benchmark audit information and inserts a timestamp into a database audit table. When all the initial steps are completed, RUNME invokes another program, DRIVER, to start the benchmark. Results are collected into a single location at the completion of the run.

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.

QUALIFYis the post-processing analysis program. This is executed on the master RTE machine, RTE1, the first RTE. It produces the numerical summaries and histograms needed for the disclosure report.

## 7.2 Functional Diagram

*A complete functional diagram of the hardware and software of the benchmark configuration including the driver must be provided. the sponsor must list all hardware and software functionality of the driver and its interface to the SUT.*



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

Ethernet and 10 Base-T local area networks (LAN) with a bandwidth of 10 megabits per second are used in the tested/priced configurations.

## 7.4 Additional Production Information

***VISIGENIC ODBC SDK***

COMPANY DESCRIPTION:

Visigenic is a leading supplier of ODBC technology and database connectivity. Visigenic markets and supports the ODBC Driver Set and the Visigenic ODBC Software Development Kits (SDKs). Developers, VARs, and ISVs can write applications that communicate simultaneously with multiple databases and on multiple platforms - all through the ODBC standard interface.

VISIGENIC ODBC DRIVER SET:

The ODBC Driver Set provides cross-plarform access to multiple SQL databases from ODBC-enabled applications. The Driver Set includes drivers that provide your application access to enterprise-wide relational databases including Oracle, Informix, Sybase, Ingres, Microsoft SQL Server and IBM DB2 family. The operating systems currently supported include HP-UX, IBM AIX, SCO, OS/2, Macintosh, Windows, Windows NT, Solaris and Sun OS, with additional support being added continuously.

Visigenic Software, Inc
951 Mariner's Island Blvd, suite 460
San Mateo, CA 94494
415-286-1900

# *Section 8.0 – Clause 7 Related Items*

## 8.1 System Pricing

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

*The total 5 year price of the entire configuration must be reported, including: hardware, software, maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.*

The details of the hardware, software and maintenance components of this system are reported in the front of this report as part of the executive summary. All 3rd party quotations are included at the end of this report in Appendix E.

## 8.2 General Availability, Throughput, and Price Performance

*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 availabilty dates, the reported availability date for the priced system must be the date at which all components are committed to be available.*

All hardware and software components of this system are currently available.

*A statement of the measured tpmC as well as the respective calculations for the 5-year pricing, price/performance and the availability date must be included.*

| | |
|---|---|
| **MAXIMUM QUALIFIED THROUGHPUT:** | **3904.00 tpmC** |
| **PRICE per tpmC:** | **$110 per tpmC** |
| **HARDWARE AVAILABILITY:** | **Feb. 1, 1997** |
| **SOFTWARE AVAILABILITY:** | **Feb. 28, 1997** |

## 8.3 Country Specific Pricing

*Additional Clause 7 related items may be included in the Full Disclosure Report for each country specific priced item configuration. Country specific pricing is subject to Clause 7.1.7.*

The system is being priced for the United States of America.

## 8.4 Usage Pricing

*For any usage pricing, the sponser must disclose 1) Usage level at which the component was priced, 2) a statement of the company policy allowing such pricing.*

The component pricing based on usage is shown below:

a) Microsoft SQL Server v6.5 User License was priced for unlimited number of users.

b) As part of HP series 9000 model E55 system prices, two 2-user HP-UX 10.01 Licenses were priced.

# *9.0 Clause 9 Related Items*

## 9.1 Auditor's Information

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

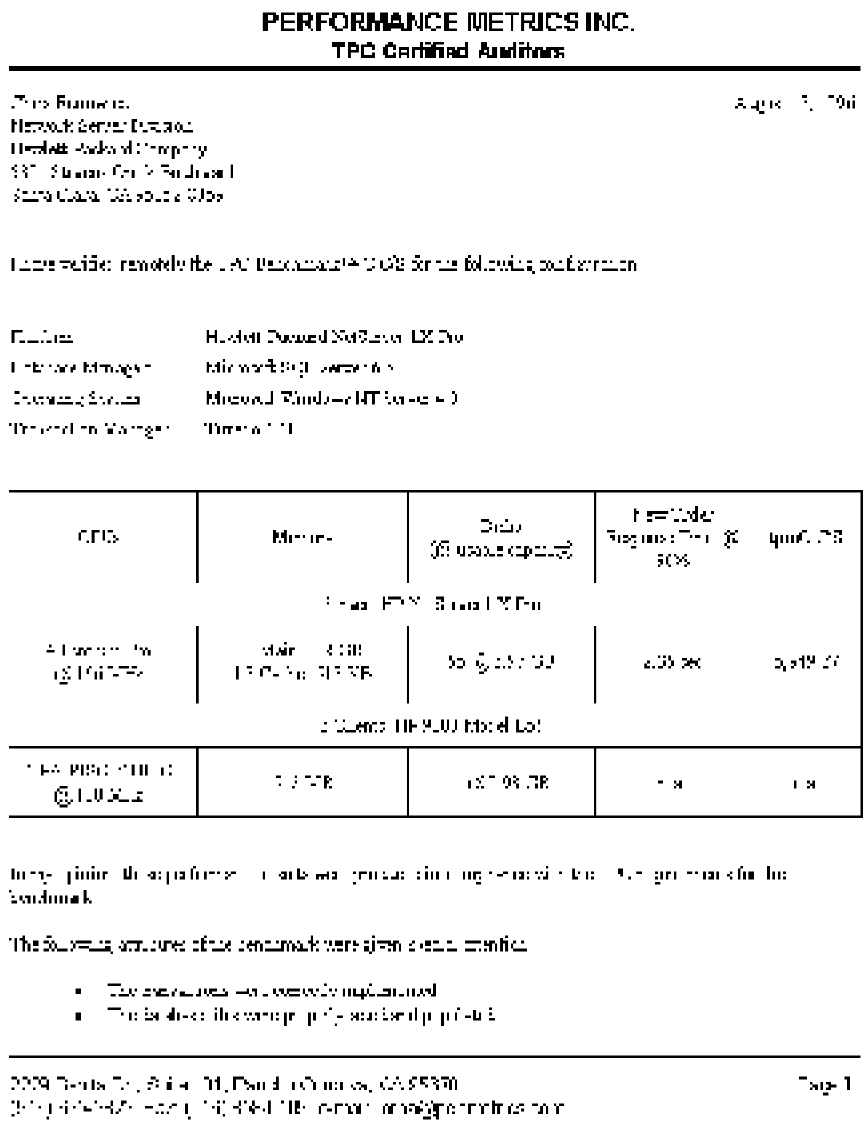
The test methodology and results of this TPC Benchmark C were audited by:

**Performance Metrics, Inc**
**TPC Certified Auditors**
2229 Benita Drive, Suite 101
Rancho Cordova, CA 95670
phone: 916-635-2822
fax: 916-8580109

The auditor was Richard Gimarc. A copy of the Attestation Letter received from the auditor is attached on the following pages.

Requests for this Full Disclosure Report (FDR) should sent to:

Hewlett-Packard Company/Network Server Division
Attention: Mary Johnson, MS 53U f/g
5301 Stevens Creek Blvd
Santa Clara, CA 95052-8059



**PERFORMANCE METRICS INC.**
**TPC Certified Auditors**

[illegible] August [illegible], 1996

Network Server Division
Hewlett-Packard Company
[illegible] Stevens Creek Boulevard
Santa Clara, CA [illegible]

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

| | |
|---|---|
| Platform: | Hewlett-Packard NetServer LX Pro |
| Database Manager: | Microsoft SQL Server 6.5 |
| Operating System: | Microsoft Windows NT Server 4.0 |
| Transaction Manager: | Tuxedo [illegible] |

| CPUs | Memory | Disks (total capacity) | 90% New Order Response Time | tpmC |
|---|---|---|---|---|
| Server: HP NetServer LX Pro | | | | |
| 4 Pentium Pro @ [illegible] MHz | Main: [illegible] GB; L2 Cache: [illegible] KB | 55 @ [illegible] GB | 2.35 sec | [illegible] |
| 2 Clients: HP 9000 Model [illegible] | | | | |
| 1 PA-RISC [illegible] @ 100 MHz | [illegible] MB | [illegible] GB | n/a | n/a |

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

The following attributes of the benchmark were given special attention:

- The transactions were correctly implemented.
- The database files were properly sized and populated.

2229 [illegible], Suite [illegible], Rancho Cordova, CA 95670 Page 1
[illegible] e-mail: [illegible]@perfmetrics.com

# Appendix A – Application Source

## A.1 Client Front-End

This appendix contains the source and makefiles for the Tuxedo client and server programs. All of the programs ran on the client machine.

*client/client.c*

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 96/04/15 15:15:37 $
 (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 "iobuf.h"
#include "tpcc.h"
#include <signal.h>
#define until(c)  while(!(c))
/* a generic transaction variable. */
generic_trans generic_transaction;
generic_trans *trans=&generic_transaction;
/* global variables set up during initialization */
int user;
ID warehouse;
ID district;
main(argc, argv)
    int argc;
    char **argv;
    {
    int key;
    /* setup the transactions */
    key = setup(argc, argv);
    /* 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();
    }
/**********************************************************************
***********************************************************************
Neworder form processing
***********************************************************************
**********************************************************************/
define_iobuf(neworder_form, 900);
int neworder(trans)
    neworder_trans *trans;
    {
    int key;
    display(neworder_form);
    key = neworder_read(trans);
    if (key != ENTER) return key;
    neworder_transaction(trans);
    neworder_write(trans);
    return key;
    }
int neworder_read(trans)
    neworder_trans *trans;
    {
```

```
int i;
int field;
int key;
int ol;
/* Our warehouse number is fixed */
trans->W_ID = warehouse;
trans->D_ID = EMPTY_NUM;
/* assume nothing set yet */
trans->C_ID = EMPTY_NUM;
for (i=0; i<15; i++)
   {
   trans->item[i].OL_I_ID = EMPTY_NUM;
   trans->item[i].OL_QUANTITY = EMPTY_NUM;
   trans->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, &trans->D_ID, 2);
            break;
      case 2: key = read_number(5, 12, &trans->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, 3, &trans->item[ol].OL_SUPPLY_W_ID,4);
         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, &trans->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, &trans->item[ol].OL_QUANTITY, 2);
         break;
      }
/* abort the screen if requested */
if (key != ENTER)
   return key;
/* calculate how many items were entered */
   for (i=15; i>0; i--)
      if ((trans->item[i-1].OL_I_ID != EMPTY_NUM) ||
         (trans->item[i-1].OL_SUPPLY_W_ID != EMPTY_NUM) ||
         (trans->item[i-1].OL_QUANTITY != EMPTY_NUM)) break;
   trans->O_OL_CNT = i;
   /* make sure all necessary fields are filled in */
   if (trans->D_ID == EMPTY_NUM)
      {field=1; msgline("Please specify district"); goto retry;}
   if (trans->C_ID == EMPTY_NUM)
      {field=2; msgline("Please specify customer id"); goto retry;}
   if (trans->O_OL_CNT == 0)
      {field=3; msgline("Please enter at least one orderline"); goto retry;}
   for (i=0; i<trans->O_OL_CNT; i++)
      {
      if (trans->item[i].OL_SUPPLY_W_ID == EMPTY_NUM)
         {field=i*3+3; msgline("Please enter supply warehouse"); goto retry;}
      if (trans->item[i].OL_I_ID == EMPTY_NUM)
         {field=i*3+4; msgline("Please enter Item id"); goto retry;}
      if (trans->item[i].OL_QUANTITY == EMPTY_NUM
       || trans->item[i].OL_QUANTITY <= 0)
         {field=i*3+5; msgline("Please enter quantity > 0"); goto retry;}
      }
   /* decide if they were all local */
   for (i=0; i<trans->O_OL_CNT; i++)
      if (trans->item[i].OL_SUPPLY_W_ID != trans->W_ID) break;
   trans->all_local = (i == trans->O_OL_CNT);
   /* display number of order lines */
   number(6, 42, trans->O_OL_CNT, 2);
   msgline("");
   flush();
   return key;
   }
neworder_write(t)
   neworder_trans *t;
   {
   int i;
   MONEY amount, total_amount, cost;
#ifdef DEBUG
      {
      /* Bret was here */
            FILE *fp;
            fp=fopen("/tmp/errorFile","a");
            if (fp == NULL)
            {
                  perror("errorFile open failed");
                  exit(1);
            }
            fprintf(fp,"debug: neworder_write: Final Fetch\n");
            fprintf(fp,"debug: t->C_CREDIT: 0x%x 0x%x 0x%x \n",
                t->C_CREDIT[0],
                t->C_CREDIT[1],
                t->C_CREDIT[2]);
            fprintf(fp,"debug: neworder_write t->status: %d \n",
                t->status);
            fclose(fp);
        }
#endif
    /* 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 */
            text(9+i, 19, t->item[i].I_NAME);
            number(9+i, 51, t->item[i].S_QUANTITY, 3);
            position(9+i, 58);   pushc(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);
    }

neworder_setup()
    {
    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, 4);
    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, 3, 4);
        empty(8+item, 10, 6);
        empty(8+item, 45, 2);
        }
    trigger();
    /* restore to the previous I/O buffer */
    out_buf = old;
    }
/**********************************************************************
***********************************************************************
Payment form processing
***********************************************************************
**********************************************************************/
define_iobuf(payment_form, 400);
int payment(trans)
    payment_trans *trans;
    {
    int key;
    display(payment_form);
    key = payment_read(trans);
    if (key != ENTER)  return key;
    payment_transaction(trans);
    payment_write(trans);
    return key;
    }
payment_setup()
    {
    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, 4);
    text(6, 42, "District:");
    empty(6, 52, 2);
    text(11, 1, "Customer:");
    empty(11, 11, 4);
    text(11, 17, "Cust-Warehouse:");
    empty(11, 33, 4);
    text(11, 39, "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:");
    trigger();
    out_buf = old;
    }
int payment_read(t)
    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, 4);
                break;
        case 4: key = read_number(11, 54, &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();
    return key;
    }
payment_write(t)
    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);
    }
```

```
/******************************************************************
******************************************************************
ORDSTAT form processing
******************************************************************
******************************************************************/
define_iobuf(ordstat_form, 300);
int ordstat(t)
    ordstat_trans *t;
    {
    int key;
    display(ordstat_form);
    key = ordstat_read(trans);
    if (key != ENTER)  return key;
    ordstat_transaction(trans);
    ordstat_write(trans);
    return key;
    }
ordstat_setup()
    {
    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, 4);
    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");
    trigger();
    /* done */
    out_buf = old;
    }
int ordstat_read(t)
    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();
   return key;
   }
ordstat_write(t)
   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, 4);
      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);
      }
   }
/*************************************************************
**************************************************************
delivery form processing
**************************************************************
**************************************************************/
define_iobuf(delivery_form, 300);
int delivery(t)
   delivery_trans *t;
   {
   int key;
   display(delivery_form);
   key = delivery_read(trans);
   if (key != ENTER)  return key;
   delivery_enque(trans);
   delivery_write(trans);
   return key;
   }
delivery_setup()
   {
   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, 4);
   text(6, 1, "Carrier Number:");
   empty(6, 17, 2);
   trigger();
   /* done */
   out_buf = old;
   }
int delivery_read(t)
```

```
    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)
        {field=1; msgline("Please enter the Carrier Number"); goto retry; }
    /* clear the message line */
    msgline("");
    flush();
    return key;
    }
delivery_write(t)
    delivery_trans *t;
    {
    if (t->status == OK)
        text(8, 1, "Execution Status: Delivery has been queued");
    else
        status(8, 1, t->status);
    }
/****************************************************************
****************************************************************
stocklev form processing
****************************************************************
****************************************************************/
define_iobuf(stocklev_form, 300);
int stocklev(t)
    stocklev_trans *t;
    {
    int key;
    display(stocklev_form);
    key = stocklev_read(trans);
    if (key != ENTER)  return key;
    stocklev_transaction(trans);
    stocklev_write(trans);
    return key;
    }
stocklev_setup()
    {
    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, 4);
    text(4, 19, "District:");
    number(4, 29, district, 2);
    text(6, 1, "Stock Level Threshold:");
    empty(6, 24, 2);
    text(8, 1, "low stock");
    trigger();
    /* done */
    out_buf = old;
    }
int stocklev_read(t)
    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 <= 0)
        {field=1; msgline("Please enter a threshold > 0"); goto retry; }
    /* clear the message line */
    msgline("");
    flush();
    return key;
    }
stocklev_write(t)
    stocklev_trans *t;
    {
    if (t->status == OK)
        number(8, 12, t->low_stock, 3);
    else
       status(10, 1, t->status);
    }
/************************************************************************
************************************************************************
login form processing
************************************************************************
*************************************************************************/
int login()
    {
    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, 4);
    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);
    trigger();
    /* 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, 4, Num);
                break;
            case 2:
                key = read_number(5, 34, &d_id, 2, Num);
                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 */
    flush();
    return key;
    }
/**********************************************************************
**********************************************************************
menu form processing
**********************************************************************
*********************************************************************/
menu_setup()
    {
    /* 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");
    }
```

```
int menu_read()
   {
   position(1, 1);
   trigger();
   return getkey();
   }
int next_field(current, key, max)
   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;
   }


msgline(str)
   char *str;
   {
   position(24, 1);
   clear_screen();
   string(str);
   flush();  /* Needed? */
   }
int setup(argc, argv)
   int argc;
   char **argv;
   {
   int key;
   /* Ignore SIGPIPE, since they occur normally */
   signal(SIGPIPE, SIG_IGN);
   /* 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);
   /* connect to the transaction processor */
   transaction_begin(user);
   return key;
   }

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

   /* detach from the delivery queue */
   delivery_done();
   /* clear the screen */
   position(1, 1);
   clear_screen();
   flush();
   }
/***********************************************************************
***********************************************************************
Screen Output Routines
************************************************************************
**********************************************************************/
number(row, col, n, width)
   int row;
   int col;
   int n;
   int width;
   {
   char str[81];
   fmt_num(str, n, width);
   text(row, col, str);
   }
real(row, col, x, width, dec)
   int row;
   int col;
   double x;
   int width;
   int dec;
   {
   char str[81];
   fmt_flt(str, x, width, dec);
```

```
    text(row, col, str);
    }
date(row, col, date_str)
    int row;
    int col;
    char *date_str;
    {
    text(row, col, date_str);
    }
date_only(row, col, date_str)
    int row;
    int col;
    char *date_str;
    {
    date_str[10] = '\0';
    text(row, col, date_str);
    }
money(row, col, x, width)
    int row;
    int col;
    double x;
    int width;
    {
    char str[81];
    fmt_money(str, x, width);
    text(row, col, str);
    }
long_text(row, col, str, width)
    int row, col, width;
    char *str;
    {
    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(*str);
        }
    }
text(row, col, str)
    int row;
    int col;
    char str[];
    {
    position(row, col);
    string(str);
    }
phone(row, col,str)
      int row;
      int col;
      char *str;
      {


       char temp[30];
       fmt_phone(temp,str);
       text(row,col,temp);
       }
zip(row, col, str)
    int row;
    int col;
    char *str;
    {
    char temp[30];
    fmt_zip(temp,str);
    text(row,col,temp);
    }
empty(row, col, len)
    int row;
    int col;
    int len;
    {
    position(row, col);
    while (len-- > 0)
        pushc('_');
    }
blanks(row, col, len)
    int row, col, len;
    {
    position(row, col);
    while (len-- > 0)
        pushc(' ');
    }
status(row, col, status)
/*******************************************************************
```

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

   if (status == OK)
       string("Transaction Committed");
   else if (status == E_INVALID_ITEM)
       string("Item number is not valid");
   else
       {
       string("Rollback -- ");
       number(row, col+30, status, 5);
       }
   }
/***************************************************************************
***************************************************************************
ASCII terminal control
***************************************************************************
**************************************************************************/
trigger()
/**************************************************************************
trigger sends a turnaround sequence to let the driver know to send input
***************************************************************************/
    {
    pushc(TRIGGER);
    }
position(row, col)
/*********************************************************************
position positions the cursor at the given row and column
*********************************************************************/
    int row;
    int col;
    {
    pushc(ESCAPE);
    pushc('[');
    if (row >= 10)
        pushc('0' + row/10);
    pushc('0'+ row%10);
    pushc(';');
    if (col >= 10)
        pushc('0' + col/10);
    pushc('0' + col%10);
    pushc('H');
    }
clear_screen()
/************************************************************************
clear_screen clears the iobuf from cursor position to end of iobuf
*************************************************************************/
    {
    pushc(ESCAPE);
    pushc('[');
    pushc('J');
    }
/**********************************************************************
**********************************************************************
Screen Input Routines
**********************************************************************
*********************************************************************/
#define funny(key)  (key != ENTER && key !=TAB && key != BACKTAB)
read_number(row, col, n, width)
/*******************************************************************
read_number reads an integer field
*******************************************************************/
    int row;
    int col;
    int *n;
    int width;
    {
    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(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;
    }
int read_money(row, col, m, width)
    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(BELL);
        err = YES;
        }
    money(row, col, *m, width);
    if (err) msgline("");
    return key;
    }
int read_real(row, col, x, width, dec)
    int row, col, width;
    double *x;
    {
    char temp[81];
    int key;
    int err;

    /* generate the current characters */
    fmt_flt(temp, *x, width, dec);
    err = NO;
    /* repeat until a valid number or a funny key is pressed */
    for (;;)
        {
        key = getfield(row, col, temp, width);
        if (funny(key)) return key;
        /* convert the field to a number */
        *x = cvt_flt(temp);
        if (*x != INVALID_FLT) break;
        msgline("Please enter a valid floating pt number");
        pushc(BELL);
        err = YES;
        }
    /* display the new number */
    real(row, col, *x, width, dec);
    if (err) msgline("");
    return key;
    }
int read_text(row, col, s, width)
    int row, col, width;
    char *s;
    {
    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;
    }

int getfield(row, col, buf, width,ftype)
    int row, col, width;
    char buf[];
    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);
    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(key);
           }
       /* CASE: char is BACKSPACE.  Erase last character. */
       else if (key == BACKSPACE && pos > 0)
           {
           pos--;
           buf[pos] = '_';
           pushc(BACKSPACE);
           pushc('_');
           pushc(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(BELL);
      }
    debug("getfield: final key: %d  buf=%*s\n", key, width, buf);
    return key;
    }
int valid_char(key, ftype)
/***********************************************************************
valid_char is true if the key is valid for this type of field
***********************************************************************/
    int key;
    FIELD_TYPE ftype;
    {
    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;
```

*lib/tpcc.h*

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 96/07/11 16:52:21 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*******************************************************************************/
#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                    */
/*********************************************************************/
  int no_warehouse;                         /*  scaling factor    */
  int no_item;                          /*  100000           */
  int no_dist_pw;                         /*      10           */
  int no_cust_pd;                         /*    3000           */
  int no_ord_pd;                          /*    3000           */
  int no_new_pd;                           /*     900           */
  int tpcc_load_seed;                      /*     900           */
/* 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;
    } 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;
    } 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[20];
   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[20];
   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[20];
   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
/* Error message strings */
static char *e_mesg[]={"Transaction complete.","Error","Invalid item number.",
                    "Not enough orders.", "Database ERROR !!!!"};
#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 */
static char *transaction_name[] =
   {"", "New_Order", "Payment", "Order-Status",
       "Delivery", "Stock-Level", "Deferred-Delivery"};
/* size of each transaction record */
static int transaction_size[] = {0,
    sizeof(neworder_trans),
    sizeof(payment_trans),
    sizeof(ordstat_trans),
    sizeof(delivery_trans),
    sizeof(stocklev_trans),
    sizeof(delivery_trans),
    0};
/* valid response time for each transaction */
static TIME valid_response[] = {0, 5, 5, 5, 5, 20};
#endif /* TPCC_INCLUDED */
```

## A.2 Transaction Source

*sqlserver/transactionb.c*

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 96/07/31 10:29:10 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*******************************************************************************/
#include "tpcc.h"
#include "errno.h"
#define MaxTries 10
int userNo;
#include <string.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#define MSG_LNG 256
/* Maximum message length */
```

```
#define max(a,b) (a>b?a:b)
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#define FAR
#include <sql.h>
#include <sqlext.h>
#include <databuf.h>
#include "odbc.h"
#include "trans_type.h"
#include <datetime.h>
void display_neword(char * msg, neworder_trans *t);
void display_payment(char * msg, payment_trans *t);
void display_ordstat(char * msg, ordstat_trans *t);
void display_stocklev(char * msg, stocklev_trans *t);
void display_delivery(char * msg, delivery_trans *t);
/** Global Order ID -mvn- **/
int fdel;
int o_id[10];
int xact_type;
/* Some local defines */
short commit_flag;
short d_id;
void sleep_before_retry();
/* For ODBC */
unsigned char odbc_buffer[128];
char errorName[16];
/* For DEBUG */
FILE *fp;
#define DEBUG
#ifdef DEBUG
int debug_flag;
#endif
void neworder_transaction(t)
        neworder_trans *t;
{
        int rc;
        int try;
        int i;
        xact_type = XACT_NEWO;
        /* return status in t->status; set in body, may override here */
        /* assume local order */
        t->all_local = 1;
        for (i=0;  i<t->O_OL_CNT; i++) {
        if (t->item[i].OL_SUPPLY_W_ID != t->W_ID) t->all_local = 0;
        }
#ifdef SORT_LINES
        sort_order_lines();
#endif
        for (try=0; try<MaxTries; try++) {
        if (try > 0) display_neword("Repeating", t);
        commit_flag = TRUE;
        o_ol_done = 0;
        o_ol_now = t->O_OL_CNT-o_ol_done;
        if ((rc=new_order_body(t)) != YES) {
                /* deal with error condition here, t->status is set */
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,
                        "new_order_rpc: return from new_order_body=%d\n",
                        rc);
        if (try > 0)
        fprintf(fp,"new_order_rpc: try %d \n",try);
        fclose(fp);
#endif
        if (rc == SQL_ERROR) {
        fp=fopen(errorName,"a");
        fprintf(fp,"new_order_rpc: error\n");
        fclose(fp);
#ifdef DEBUG
        dump_neworder_params();
#endif
        display_neword("Failed", t);
        return; }
        /* else deadlock */
        else if( rc == DEADLOCK ) {
#ifdef DEBUG
        debug_flag=1;
        fp=fopen(errorName,"a");
        fprintf(fp,"new_order_rpc: deadlock\n");
        fclose(fp);
        dump_neworder_params();
#endif
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
        fp=fopen(errorName,"a");
        fprintf(fp,
                                "neworder_rpc: SQLFreeStmt rc=%d\n",
                                rc);
        fclose(fp);
        return;
        }
        sleep_before_retry();
        continue;
        } else {
        fp=fopen(errorName,"a");
        fprintf(fp,
                          "neworder_rpc: SQL Unknown Error rc=%d\n",
                          rc);
        fclose(fp);
                        return;
        }
        }
            /* it was YES check try count for message */
        if (try > 0) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"neworder_rpc: try %d Success!!\n",try);
        dump_neworder_params();
        fclose(fp);
#endif
        }
        break;
        }  /* end of for loop on MaxTries */
        if (try >= MaxTries) {
        display_neword("Failed", t);
            t->status=E_DB_ERROR;
        return;
        }
        return;
}  /* end of neworder_transaction() */
int new_order_body (t)
        neworder_trans *t;
{
        RETCODE rc;
        int i,j,num_ol;
        TIMESTAMP_STRUCT o_entry_d;
        double ol_amount;
        char generic[4];
        t->status = E_DB_ERROR;  /* multiple returns possible for problems */
        num_ol=t->O_OL_CNT;
        strcpy((char *) odbc_buffer, "{call tpcc_neworder(?,?,?,?,?" );
        for (i=0;i<num_ol; i++)
          strcat((char *) odbc_buffer, ",?,?,?" );
        strcat((char *) odbc_buffer, ")}");
        /* Bind Parameters */
#ifdef DEBUG
        if(debug_flag) {
        fp=fopen(errorName,"a");
        fprintf(fp,"debug: neworder_body: Starting BindParameters\n");
        fclose(fp);
        }
#endif
        SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_INTEGER, 0, 0, &t->W_ID, 0, NULL);
        SQLBindParameter(hstmt, 2, SQL_PARAM_INPUT, SQL_C_LONG,
        SQL_INTEGER, 0, 0, &t->D_ID, 0, NULL);
        SQLBindParameter(hstmt, 3, SQL_PARAM_INPUT, SQL_C_LONG,
        SQL_INTEGER, 0, 0, &t->C_ID, 0, NULL);
        SQLBindParameter(hstmt, 4, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_INTEGER, 0, 0, &t->O_OL_CNT, 0, NULL);
        SQLBindParameter(hstmt, 5, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_INTEGER, 0, 0, &t->all_local, 0, NULL);
#ifdef DEBUG
        if(debug_flag) {
        fp=fopen(errorName,"a");
        fprintf(fp,
             "debug: neworder_body: Finished Initial BindParameters\n");
        fclose(fp);
        }
#endif
        /* now, deal with the order lines */
        for(i = 0; i < num_ol; i++) {
        int parm_num = 6 + 3*i;
#ifdef DEBUG
        if(debug_flag) {
        fp=fopen(errorName,"a");
        fprintf(fp,
                "debug: neworder_body: BindP Loop i=%d parm_num=%d\n",
        i,parm_num);
        fclose(fp);
        }
#endif
        SQLBindParameter(hstmt, (UWORD)(parm_num++), SQL_PARAM_INPUT,
                SQL_C_SLONG, SQL_INTEGER, 0, 0,
                &t->item[i].OL_I_ID, 0, NULL);
        SQLBindParameter(hstmt, (UWORD)(parm_num++), SQL_PARAM_INPUT,
                SQL_C_SLONG, SQL_INTEGER, 0, 0,
                &t->item[i].OL_SUPPLY_W_ID,
        0, NULL);
        SQLBindParameter(hstmt, (UWORD)(parm_num), SQL_PARAM_INPUT,
                SQL_C_SLONG, SQL_INTEGER, 0, 0,
                &t->item[i].OL_QUANTITY,
        0, NULL);
        }
#ifdef DEBUG
        if(debug_flag) {
        fp=fopen(errorName,"a");
        fprintf(fp,"debug: neworder_body: SQLExecDirect\n");
        fclose(fp);
        }
#endif
        rc = SQLExecDirect(hstmt, odbc_buffer, SQL_NTS);
        if(rc != SQL_SUCCESS ) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        if(rc == SQL_SUCCESS_WITH_INFO)
        fprintf(fp,
        "neworder_bdy: SQLExDir=Success With Info\n");
        else
        fprintf(fp,"neworder_body: SQLExecDirect\n");
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        for (i = o_ol_done; i < (int)(o_ol_done+o_ol_now); i++) {
#ifdef DEBUG
        if(debug_flag) {
        fp=fopen(errorName,"a");
        fprintf(fp,"debug: neworder_body: BindCol Loop i=%d\n",i);
        fprintf(fp,"debug: BindCol Loop: o_ol_done=%d\n",o_ol_done);
        fprintf(fp,"debug: BindCol Loop: o_ol_now=%d\n",o_ol_now);
        fclose(fp); }
#endif
        SQLBindCol(hstmt, 1, SQL_C_CHAR,
                &t->item[i].I_NAME,
        sizeof(t->item[i].I_NAME), NULL);
        SQLBindCol(hstmt, 2, SQL_C_SLONG,
        &t->item[i].S_QUANTITY, 0, NULL);
            SQLBindCol(hstmt, 3, SQL_C_CHAR,
                &generic[0], sizeof(generic), NULL);
        SQLBindCol(hstmt, 4, SQL_C_DOUBLE,
        &t->item[i].I_PRICE, 0, NULL);
        SQLBindCol(hstmt, 5, SQL_C_DOUBLE,
        &ol_amount, 0, NULL);

#ifdef DEBUG
        if(debug_flag) {
        fp=fopen(errorName,"a");
        fprintf(fp,"debug: neworder_body: SQLFetch\n");
        fclose(fp);
        }
#endif
        rc = SQLFetch(hstmt);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,
        "neworder_body: SQLFetch i=%d o_ol_done=%d o_ol_now=%d\n",
        i, o_ol_done, o_ol_now);
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        t->item[i].I_PRICE = t->item[i].I_PRICE * 100;
        t->item[i].brand_generic = generic[0];
#ifdef DEBUG
        if(debug_flag) {
        fp=fopen(errorName,"a");
        fprintf(fp,"debug: neworder_body: SQLMoreResults\n");
        fclose(fp); }
#endif
        rc = SQLMoreResults(hstmt);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"neworder_body: SQLMoreResults\n");
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        }   /* end of the for loop on order lines */
#ifdef DEBUG
        if(debug_flag) {
        fp=fopen(errorName,"a");
        fprintf(fp,"debug: neworder_body: Final Binds\n");
        fclose(fp); }
#endif
        SQLBindCol(hstmt, 1, SQL_C_DOUBLE, &t->W_TAX, 0, NULL);
        SQLBindCol(hstmt, 2, SQL_C_DOUBLE, &t->D_TAX, 0, NULL);
        SQLBindCol(hstmt, 3, SQL_C_SLONG, &t->O_ID, 0, NULL);
        SQLBindCol(hstmt, 4, SQL_C_CHAR, &t->C_LAST,
        sizeof(t->C_LAST), NULL);
        SQLBindCol(hstmt, 5, SQL_C_DOUBLE, &t->C_DISCOUNT, 0, NULL);
        SQLBindCol(hstmt, 6, SQL_C_CHAR, &t->C_CREDIT,
        sizeof(t->C_CREDIT), NULL);
        SQLBindCol(hstmt, 7, SQL_C_TIMESTAMP, &o_entry_d, 0, NULL);
        SQLBindCol(hstmt, 8, SQL_C_SSHORT, &commit_flag, 0, NULL);
        rc = SQLFetch(hstmt);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"neworder_body: SQLFetch2\n");
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        fmt_date(&t->O_ENTRY_D,&o_entry_d);
#ifdef DEBUG
        if(debug_flag) {
        fp=fopen(errorName,"a");
        fprintf(fp,"debug: neworder_body: SQLFreeStmt\n");
        fclose(fp);
        }
#endif
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"neworder_body: SQLFreeStmt\n");
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        if (commit_flag)
                t->status = OK;
        else
        t->status = E_INVALID_ITEM;
        return YES;
}  /* end of new_order body */
void payment_transaction (t)
        payment_trans *t;
{
        int rc;
        int try;
        /* move the transaction data passed from rte into the global area */

        xact_type = XACT_PAYM;
        for (try=0; try<MaxTries; try++) {
        if (try>0) display_payment("Repeating", t);
        if ((rc=payment_body(t)) != YES) {
        if (rc == SQL_ERROR) {
        display_payment("Failed", t);
        return;
        }
        /* else deadlock */
        else if( rc == DEADLOCK ) {
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
        fp=fopen(errorName,"a");
        fprintf(fp,"payment_rpc: SQLFreeStmt\n");
        fclose(fp);
                                return;
        }
#ifdef DEALLOC
        if(pmt_dataptr_d.c_id == 0) {
        SQLExecDirect(hstmt,
                                "deallocate c_payment", SQL_NTS);
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,
                                        "payment_rpc: SQLFreeStmt\n"
                                        );
        fclose(fp);
#endif
        return;
        }
        }
#endif /* DEALLOC */
        sleep_before_retry();
        continue;
        } else {
        fp=fopen(errorName,"a");
        fprintf(fp,
                        "payment_rpc: SQL Unknown Error rc=%d\n",
                        rc);
        fclose(fp);
        return;
        }
        } /* end of was not YES */
        if (try > 0) {
        fp=fopen(errorName,"a");
        fprintf(fp,"payment_rpc: try %d Success!!\n",try);
        fclose(fp);
        }
        break;
        }
        if (try >= MaxTries) {
        display_payment("Failed", t);
                t->status = E_DB_ERROR;
        return;
        }
        return;
}       /* end of payment_transaction() */
int payment_body (t)
        payment_trans *t;
{
        RETCODE rc;
        UWORD rowStatus[5];
        UDWORD rowfetched;
        TIMESTAMP_STRUCT pay_date;
        double sqlAmount;
        t->status = E_DB_ERROR;  /* multiple returns possible for problems */
        if (t->byname)
        t->C_ID = 0;
        strcpy((char *) odbc_buffer, "{call tpcc_payment(?,?,?,?,?,?,?)}");
        /* Bind Parameters for payment stored procedure */
        rc = SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_SMALLINT, 0, 0, &t->W_ID, 0, NULL);
        rc = SQLBindParameter(hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_SMALLINT, 0, 0, &t->C_W_ID, 0, NULL);
        sqlAmount = t->H_AMOUNT/100.00;
        rc = SQLBindParameter(hstmt, 3, SQL_PARAM_INPUT, SQL_C_DOUBLE,
        SQL_NUMERIC, 6, 2, &sqlAmount, 0, NULL);
        rc = SQLBindParameter(hstmt, 4, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_TINYINT, 0, 0, &t->D_ID, 0, NULL);
        rc = SQLBindParameter(hstmt, 5, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_TINYINT, 0, 0, &t->C_D_ID, 0, NULL);
        rc = SQLBindParameter(hstmt, 6, SQL_PARAM_INPUT,SQL_C_LONG,
                    SQL_INTEGER, SQL_NTS, 0, &t->C_ID, 0, NULL);
        rc = SQLBindParameter(hstmt, 7, SQL_PARAM_INPUT, SQL_C_CHAR,
        SQL_CHAR, SQL_NTS, 0, &t->C_LAST,
        sizeof(t->C_LAST), NULL);
        rc = SQLExecDirect(hstmt, odbc_buffer, SQL_NTS);
        if(rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"payment_body_odbc1: SQLFreeStmt\n");
                fprintf(fp, "t->H_AMOUNT: %lf \n", t->H_AMOUNT);
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        SQLBindCol(hstmt, 1, SQL_C_LONG,
                &t->C_ID, 0, NULL);
        SQLBindCol(hstmt, 2, SQL_C_CHAR, t->C_LAST, sizeof(t->C_LAST),
NULL);
        SQLBindCol(hstmt, 3, SQL_C_TIMESTAMP, &pay_date, 0, NULL);
        SQLBindCol(hstmt, 4, SQL_C_CHAR, t->W_STREET_1,
        sizeof(t->W_STREET_1), NULL);
        SQLBindCol(hstmt, 5, SQL_C_CHAR, t->W_STREET_2,
        sizeof(t->W_STREET_2), NULL);
        SQLBindCol(hstmt, 6, SQL_C_CHAR, t->W_CITY, sizeof(t->W_CITY),
NULL);
        SQLBindCol(hstmt, 7, SQL_C_CHAR, t->W_STATE, sizeof(t->W_STATE),
NULL);
        SQLBindCol(hstmt, 8, SQL_C_CHAR, t->W_ZIP, sizeof(t->W_ZIP), NULL);
        SQLBindCol(hstmt, 9, SQL_C_CHAR, t->D_STREET_1,
        sizeof(t->D_STREET_1), NULL);
        SQLBindCol(hstmt, 10, SQL_C_CHAR, t->D_STREET_2,
        sizeof(t->D_STREET_2), NULL);
        SQLBindCol(hstmt, 11, SQL_C_CHAR, t->D_CITY, sizeof(t->D_CITY),
NULL);
        SQLBindCol(hstmt, 12, SQL_C_CHAR, t->D_STATE,
                sizeof(t->D_STATE), NULL);
        SQLBindCol(hstmt, 13, SQL_C_CHAR, t->D_ZIP, sizeof(t->D_ZIP), NULL);
        SQLBindCol(hstmt, 14, SQL_C_CHAR, t->C_FIRST,
        sizeof(t->C_FIRST), NULL);
        SQLBindCol(hstmt, 15, SQL_C_CHAR, t->C_MIDDLE,
        sizeof(t->C_MIDDLE), NULL);
        SQLBindCol(hstmt, 16, SQL_C_CHAR, t->C_STREET_1,
        sizeof(t->C_STREET_1), NULL);
        SQLBindCol(hstmt, 17, SQL_C_CHAR, t->C_STREET_2,
        sizeof(t->C_STREET_2), NULL);
        SQLBindCol(hstmt, 18, SQL_C_CHAR, t->C_CITY, sizeof(t->C_CITY),
NULL);
        SQLBindCol(hstmt, 19, SQL_C_CHAR, t->C_STATE,
        sizeof(t->C_STATE), NULL);
        SQLBindCol(hstmt, 20, SQL_C_CHAR, t->C_ZIP, sizeof(t->C_ZIP), NULL);
        SQLBindCol(hstmt, 21, SQL_C_CHAR, t->C_PHONE,
        sizeof(t->C_PHONE), NULL);
        SQLBindCol(hstmt, 22, SQL_C_CHAR, t->C_SINCE,
        sizeof(t->C_SINCE), NULL);
        SQLBindCol(hstmt, 23, SQL_C_CHAR, t->C_CREDIT,
        sizeof(t->C_CREDIT), NULL);
        SQLBindCol(hstmt, 24, SQL_C_DOUBLE, &t->C_CREDIT_LIM,
        sizeof(t->C_CREDIT_LIM), NULL);
        SQLBindCol(hstmt, 25, SQL_C_DOUBLE, &t->C_DISCOUNT,
        sizeof(t->C_DISCOUNT), NULL);
        SQLBindCol(hstmt, 26, SQL_C_DOUBLE, &t->C_BALANCE,
        sizeof(t->C_BALANCE), NULL);
        SQLBindCol(hstmt, 27, SQL_C_CHAR, t->C_DATA, sizeof(t->C_DATA),
NULL);
        rc = SQLFetch(hstmt);
        /* rc = SQLExtendedFetch(hstmt, SQL_FETCH_NEXT, (SDWORD) 0,
                        &rowfetched, &rowStatus[0] ); */
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"payment_body_odbc2: SQLFreeStmt\n");
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        fmt_date(&t->H_DATE,&pay_date);
        t->C_CREDIT_LIM = t->C_CREDIT_LIM * 100;
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"payment_body_odbc3: SQLFreeStmt\n");
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        t->status = OK;
        return YES;
}  /* end of payment_body() */
void ordstat_transaction(t)
        ordstat_trans *t;
{
        int rc;
        int try;

        /* ords_dataptr->w_id = t->W_ID; */
        /* ords_dataptr->d_id = t->D_ID; */
        /* ords_dataptr->c_id = t->C_ID; */
        /* strcpy(ords_dataptr->c_last,t->C_LAST); */
        xact_type = XACT_ORDS;
        for (try=0; try<MaxTries; try++) {
        if (try>0) display_ordstat("Repeating", t);
        if ((rc=ordstat_body(t)) != YES) {
        if (rc == SQL_ERROR) {
        display_ordstat("Failed", t);
        return;
        } else if( rc == DEADLOCK ) {
#ifdef DEALLOC
        if(ords_dataptr_d.c_id == 0) {
        SQLExecDirect(hstmt,
                                "deallocate c_orderstatus", SQL_NTS);
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
        fp=fopen(errorName,"a");
        fprintf(fp,
                                 "order_status_rpc: SQLFreeStmt\n");
        fclose(fp);
        return;
        }
        }
#endif /* DEALLOC */
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,
                                "order_status_rpc: SQLFreeStmt\n");
        fclose(fp);
#endif
        return;
        }
        sleep_before_retry();
        continue;
        } else {
        fp=fopen(errorName,"a");
        fprintf(fp,
                        "order_status_rpc: SQL Unknown Error rc=%d\n",
                         rc);
        fclose(fp);
        return;
        }
        }
        if (try > 0) {
        fp=fopen(errorName,"a");
        fprintf(fp,"order_status_rpc: try %d Success!!\n",try);
        fclose(fp);
        }
        break;
        }   /* end of the MaxTries for loop  */
        if (try >= MaxTries) {
```

```
        display_ordstat("Failed", t);
                t->status = E_DB_ERROR;
        return;
        }
        return;
}       /* end of ordstat_transaction() */
#ifdef DEBUG
void
mem_dump(char *s, char *p, int len)
{
        int i;
        fprintf(fp, "%s:\n\t", s);\
        for(i=0;i<len;i++)
        fprintf(fp, "%2.2x ", 0xff & *p++);
        fprintf(fp, "\n");

}
#endif
int ordstat_body (t)
        ordstat_trans *t;
{
        int not_done;
        int i;
        int count = 0;
        RETCODE rc;
        TIMESTAMP_STRUCT delivery_d;
        if (t->byname)
        {
        t->C_ID = 0;
        }
        t->status = E_DB_ERROR;  /* multiple returns possible for problems */

        /* Bind Parameters */
        SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_SMALLINT, 0, 0, &t->W_ID, 0, NULL);
        SQLBindParameter(hstmt, 2, SQL_PARAM_INPUT,
        SQL_C_SLONG, SQL_TINYINT, 0, 0, &t->D_ID, 0, NULL);
        SQLBindParameter(hstmt, 3, SQL_PARAM_INPUT,
        SQL_C_SLONG, SQL_INTEGER, 0, 0, &t->C_ID, 0, NULL);
        SQLBindParameter(hstmt, 4, SQL_PARAM_INPUT,
        SQL_C_CHAR, SQL_CHAR, SQL_NTS, 0, &t->C_LAST,
        sizeof(t->C_LAST), NULL);
        rc = SQLExecDirect(hstmt,
            (unsigned char *) "{call tpcc_orderstatus(?,?,?,?)}", SQL_NTS);
        if(rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        not_done = TRUE; i = 0;
        while (not_done) {
        SQLBindCol(hstmt, 1, SQL_C_SSHORT,
                    &t->item[i].OL_SUPPLY_W_ID,
                    0, NULL);
        SQLBindCol(hstmt, 2, SQL_C_SLONG,
                    &t->item[i].OL_I_ID,
                    0, NULL);
        SQLBindCol(hstmt, 3, SQL_C_SSHORT,
                    &t->item[i].OL_QUANTITY,
                    0, NULL);
        SQLBindCol(hstmt, 4, SQL_C_DOUBLE,
                    &t->item[i].OL_AMOUNT,
                    0, NULL);
        SQLBindCol(hstmt, 5, SQL_C_TIMESTAMP,
                    &delivery_d, 0, NULL);

        rc = SQLFetch(hstmt);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        if (rc == SQL_NO_DATA_FOUND)
        not_done = FALSE;
        fmt_date(&t->item[i].OL_DELIVERY_DATE, &delivery_d);
        t->item[i].OL_AMOUNT = t->item[i].OL_AMOUNT * 100;
        i++;
        }   /* end of the while */
        t->ol_cnt = i -1;
        rc = SQLMoreResults(hstmt);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        SQLBindCol(hstmt, 1, SQL_C_LONG,
                &t->C_ID, 0, NULL);
        SQLBindCol(hstmt, 2, SQL_C_CHAR,
                &t->C_LAST,
                sizeof(t->C_LAST), NULL);
        SQLBindCol(hstmt, 3, SQL_C_CHAR,
                &t->C_FIRST,
                sizeof(t->C_FIRST), NULL);
        SQLBindCol(hstmt, 4, SQL_C_CHAR,
                &t->C_MIDDLE,
                sizeof(t->C_MIDDLE), NULL);
        SQLBindCol(hstmt, 5, SQL_C_CHAR,
                &t->O_ENTRY_DATE,
                sizeof(t->O_ENTRY_DATE), NULL);
        SQLBindCol(hstmt, 6, SQL_C_SSHORT,
                &t->O_CARRIER_ID,
                0, NULL);
        SQLBindCol(hstmt, 7, SQL_C_DOUBLE,
                &t->C_BALANCE,
                0, NULL);
        SQLBindCol(hstmt, 8, SQL_C_SLONG,
                &t->O_ID,
                0, NULL);
        rc = SQLFetch(hstmt);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        mem_dump("C_ID", (char *)&t->C_ID, sizeof(t->C_ID));
        mem_dump("C_LAST", (char *)&t->C_LAST, sizeof(t->C_LAST));
        mem_dump("C_FIRST", (char *)&t->C_FIRST, sizeof(t->C_FIRST));
        mem_dump("C_MIDDLE", (char *)&t->C_MIDDLE, sizeof(t->C_MIDDLE));
        mem_dump("O_ENTRY_DATE", (char *)&t->O_ENTRY_DATE,
        sizeof(t->O_ENTRY_DATE));
        mem_dump("O_CARRIER_ID", (char *)&t->O_CARRIER_ID,
        sizeof(t->O_CARRIER_ID));
        mem_dump("C_BALANCE", (char *)&t->C_BALANCE,
        sizeof(t->C_BALANCE));
        mem_dump("O_ID", (char *)&t->O_ID, sizeof(t->O_ID));
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"order_status_body: SQLFreeStmt\n");
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        t->status = OK;
        return YES;
}  /* end of ordstat_body()  */
void stocklev_transaction(t)
        stocklev_trans *t;
{
        int rc;
        int try;

        xact_type = XACT_STOCK;
        for (try = 0; try < MaxTries; try ++) {
        if (try > 0) display_stocklev("Repeating", t);
        if ((rc=stocklev_body(t)) != YES) {
        if (rc == SQL_ERROR) {
        display_stocklev("Failed", t);
        return;
        } else if( rc == DEADLOCK ) {
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
        fp=fopen(errorName,"a");
        fprintf(fp,
                                "stock_level_rpc: SQLFreeStmt\n");
        fclose(fp);
        return;
        }
        sleep_before_retry();
        continue;
        } else {
        fp=fopen(errorName,"a");
        fprintf(fp,
                          "stock_level_rpc: SQL Unknown Error rc=%d\n",
                          rc);
        fclose(fp);
        return;
        }
        }
        if (try > 0) {
        fp=fopen(errorName,"a");
        fprintf(fp,"stock_level_rpc: try %d Success!!\n",try);
        fclose(fp);
        }
        break;
        } /* end of the for loop */
        if (try >= MaxTries) {
        display_stocklev("Failed", t);
                t->status = E_DB_ERROR;
        return;
        }
        return;
}       /* end of stocklev_transaction() */
int stocklev_body (t)
        stocklev_trans *t;
{
        RETCODE rc;
        t->status = E_DB_ERROR;  /* multiple returns possible for problems */
        /* Bind Parameters */
        SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_SMALLINT, 0, 0, &t->W_ID, 0, NULL);
        SQLBindParameter(hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_TINYINT, 0, 0, &t->D_ID, 0, NULL);
        SQLBindParameter(hstmt, 3, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_SMALLINT, 0, 0, &t->threshold, 0, NULL);
        rc = SQLExecDirect(hstmt,
                        (unsigned char *) "{call tpcc_stocklevel(?,?,?)}",
        SQL_NTS);
        if(rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        SQLBindCol(hstmt, 1, SQL_C_SLONG,
                &t->low_stock, 0, NULL);
        rc = SQLFetch(hstmt);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"stock_level_body: SQLFreeStmt\n");
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        t->status = OK;
        return YES;
}       /*( end of stocklev_body()  */
int delivery_transaction (t)
    delivery_trans *t;
{
        int rc;
        int try;
        xact_type = XACT_DEL;
        for (try = 0; try < MaxTries; try++) {
        if (try > 0) display_delivery("Repeating", t);
        if ((rc=delivery_body(t)) != YES) {
        if (rc == SQL_ERROR) {
        display_delivery("Failed", t);
                        return INVALID_DATA;
        }
        /* else deadlock */
        else if( rc == DEADLOCK ) {
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
        fp=fopen(errorName,"a");
        fprintf(fp,
        "delivery_trans: SQLFreeStmt\n");
        fclose(fp);
        return;
        }
        sleep(1);
        continue;
        } else {
        fp=fopen(errorName,"a");
        fprintf(fp,"delivery_trans: SQL Unknown Error rc=%d\n",rc);
        fclose(fp);
        return;
        }
        }
        if (try > 0) {
        fp=fopen(errorName,"a");
        fprintf(fp,"delivery_trans: try %d Success!!\n",try);
        fclose(fp);
        }
        break;

        }

        if (try >= MaxTries) {
        display_delivery("Failed", t);
                t->status = E_DB_ERROR;
                return;
        }
        return;
}   /* end of delivery_transaction  */
int delivery_body (t)
    delivery_trans *t;
{
        RETCODE rc;
        int dist;
        /* Bind Parameters for the delivery stored procedure */
        rc = SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_SMALLINT, 0, 0, &t->W_ID, 0, NULL);
        rc = SQLBindParameter(hstmt, 2, SQL_PARAM_INPUT, SQL_C_SLONG,
        SQL_SMALLINT, 0, 0, &t->O_CARRIER_ID, 0, NULL);
        rc = SQLExecDirect(hstmt,
                (unsigned char *) "{call tpcc_delivery(?,?)}", SQL_NTS);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        /* all the district info is returned with one fetch from proc */
        for (dist=0;dist<10;dist++) {
        rc = SQLBindCol(hstmt, (UWORD)(dist+1), SQL_C_SLONG,
                &t->order[dist].O_ID, 0, NULL);
        }
        rc = SQLFetch(hstmt);
        if(rc == SQL_ERROR) {
        fp=fopen(errorName,"a");
        fprintf(fp,
                "delivery_body: SQLFetch rc=%d, o_id=%d\n", rc, o_id);
        fclose(fp);
        return ODBCError(henv, hdbc, hstmt);
        }
        /*
        ** Print delivery information (w_id, d_id, o_id ).
        ** If o_id == NULL, there were no new orders in that (w_id, d_id).
        */
        rc = SQLFreeStmt(hstmt, SQL_CLOSE);
        if(rc == SQL_ERROR) {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"big_delivery_body: SQLFreeStmt\n");
        fclose(fp);
#endif
        return ODBCError(henv, hdbc, hstmt);
        }
        /* now set required t->status information  */
        for (dist=0;dist<10;dist++) {
        if (t->order[dist].O_ID == 0)
        t->order[dist].status=E_NOT_ENOUGH_ORDERS;
        else
        t->order[dist].status = OK;
        }
        return YES;
}  /* end of delivery_body()  */
/* void sleep_before_retry() { sleep(1); } */
void display_neword(msg, t)
char * msg;
neworder_trans *t;
{
        int i;
        fp=fopen(errorName,"a");
        fprintf(fp,"display_neword:%s\n",msg);
        fprintf(fp,"New Order w=%d, d=%d, c=%d, %d lines: [",
        t->W_ID, t->D_ID, t->C_ID, t->O_OL_CNT);
        for (i=0; i<(int)t->O_OL_CNT; i++)
        fprintf(fp," %d", t->item[i].OL_I_ID);
        fprintf(fp,"]\n");
        fclose(fp);
}  /* end of display_neword(msg,t) */
void display_payment(msg, t)
```

```
char * msg;
payment_trans *t;
{
        int i;
        fp=fopen(errorName,"a");
        fprintf(fp,"display_payment:%s\n",msg);
        fprintf(fp,"Payment w=%d/%d, d=%d/%d, c=%d l=%s\n", t->W_ID,
        t->C_W_ID, t->D_ID, t->C_D_ID, t->C_ID, t->C_LAST);
        fclose(fp);
}  /* end of display_payment(msg,t) */
void display_ordstat(msg, t)
char * msg;
ordstat_trans *t;
{
        int i;
        fp=fopen(errorName,"a");
        fprintf(fp,"display_ordstat:%s\n",msg);
        fprintf(fp,"Order Status cw=%d, cd=%d, c=%d l=%s\n", t->W_ID,
        t->D_ID, t->C_ID, t->C_LAST);
        fclose(fp);
}  /* end of display_ordstat(msg,t) */
void display_stocklev(msg, t)
char * msg;
stocklev_trans *t;
{
        int i;
        fp=fopen(errorName,"a");
        fprintf(fp,"display_stocklev:%s\n",msg);
        fprintf(fp,"Stock Level w=%d, d=%d, th=%d\n", t->W_ID,
        t->D_ID, t->threshold);
        fclose(fp);
}  /* end of display_stocklev(msg,t) */
void display_delivery(msg, t)
char * msg;
delivery_trans *t;
{
        int i;
        fp=fopen(errorName,"a");
        fprintf(fp,"display_delivery:%s\n",msg);
        fprintf(fp,"Delivery w=%d, carrier=%d\n", t->W_ID, t->O_CARRIER_ID);
        fclose(fp);
}  /* end of display_delivery(msg,t) */
#ifdef SORT_LINES
sort_order_lines () {
        /*
        ** Sort order_lines in a new_order by i_id. Reduces possibility of deadlock.
        ** Brute force insertion sort -- works OK for <= 15 rows.
        */
        int i, j; tux_item_line temp;
        for (j=1; j<newo_dataptr_d.o_ol_cnt; j++) {
        if (newo_dataptr_d.ol_table[j-1].ol_i_id > newo_dataptr_d.ol_table[j].ol_i_id) {
        temp = newo_dataptr_d.ol_table[j];
        newo_dataptr_d.ol_table[j] = newo_dataptr_d.ol_table[j-1];
        for (i=j-2; i>=0 && temp.ol_i_id < newo_dataptr_d.ol_table[i].ol_i_id; i--) {
        newo_dataptr_d.ol_table[i+1] = newo_dataptr_d.ol_table[i];
        } newo_dataptr_d.ol_table[i+1] = temp;
        }
        }
}
#endif
/*******************************************************************
/* Allocate environment and connection handles                    *
/* Connect to the data source                                     *
/* Allocate a statement handle                                    *
/*******************************************************************/
int connect_odbc_user() {
        RETCODE rc;
        UCHAR db[51];
        SWORD dblen;
        char del_fifo[] = DEL_FIFO;
        extern int fdel;
        pid=getpid();
        /* Initialize the error log file Name */
        sprintf(errorName,"errorlog.%d",pid);
        /* Done with initialization */
        rc = SQLAllocEnv(&henv);
        if (rc != SQL_SUCCESS & rc != SQL_SUCCESS_WITH_INFO)
        {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return(ODBCError(SQL_NULL_HENV, SQL_NULL_HDBC,
SQL_NULL_HSTMT));
        }
        rc = SQLAllocConnect(henv, &hdbc);
        if (rc != SQL_SUCCESS & rc != SQL_SUCCESS_WITH_INFO)
        {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return(ODBCError(henv, SQL_NULL_HDBC, SQL_NULL_HSTMT));
        }
        rc = SQLConnect(hdbc, dsn, SQL_NTS, user, SQL_NTS, passwd,
SQL_NTS);
        if (rc != SQL_SUCCESS & rc != SQL_SUCCESS_WITH_INFO)
        {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return(ODBCError(henv, hdbc, SQL_NULL_HSTMT));
        }
        rc = SQLGetInfo(hdbc, SQL_DBMS_NAME, &db, (SWORD) sizeof(db),
&dblen);
        if (rc != SQL_SUCCESS & rc != SQL_SUCCESS_WITH_INFO)
        {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return(ODBCError(henv, hdbc, SQL_NULL_HSTMT));
        }
        /* printf("\nODBC connection to %s successful.\n\n",db); */
        rc = SQLAllocStmt(hdbc, &hstmt);
        if (rc != SQL_SUCCESS & rc != SQL_SUCCESS_WITH_INFO)
        {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return(ODBCError(henv, hdbc, SQL_NULL_HSTMT));
        }
        rc = SQLSetStmtOption(hstmt, SQL_QUERY_TIMEOUT, 300L);
        if (rc != SQL_SUCCESS & rc != SQL_SUCCESS_WITH_INFO)
        {
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"calling ODBCError from line %d, file %s\n",
        __LINE__, __FILE__);
        fclose(fp);
#endif
        return(ODBCError(henv, hdbc, SQL_NULL_HSTMT));
        }
/*      if( (fdel=open(del_fifo,O_RDWR)) < 0 ) {
        fprintf(stderr,"\nError in opening FIFO: %s, Errno=%d\n", del_fifo, errno);
        return(-1); } */
        return 0;
} /* end of connect_odbc_user  */
#ifdef DEBUG
dump_neworder_params(t)
neworder_trans *t;
{
        fp=fopen(errorName,"a");
        fprintf(fp," t->W_ID %d\n", t->W_ID);
        fprintf(fp," newo_dataptr_d.d_id %d\n", t->D_ID);
        fprintf(fp," newo_dataptr_d.c_id %d\n", t->C_ID);
        fprintf(fp," newo_dataptr_d.o_ol_cnt %d\n", t->O_OL_CNT);
        fprintf(fp," newo_dataptr_d.o_all_local %d\n", t->all_local);
        fclose(fp);
}
#endif
/**********************************************************************
/* ODBCError - Use SQLError to get error data, then print it. *
/*********************************************************************/
int ODBCError(henv, hdbc, hstmt) HENV henv; HDBC hdbc; HSTMT hstmt; {
        struct timeval now;
        time_t timenow;
        FILE *fp; RETCODE rc;
        /* general return code for API */
        UCHAR szSqlState[MSG_LNG];
        /* SQL state string */
        SDWORD pfNativeError;
        /* Native error code */
        UCHAR szErrorMsg[MSG_LNG];
        /* Error msg text buffer pointer*/
        SWORD pcbErrorMsg;
        /* Error msg text Available bytes*/
        char msgtext[MSG_LNG];
        /* message text work area */
        int retcode = SQL_ERROR;
        rc = SQLError(henv, hdbc, hstmt, szSqlState,
        &pfNativeError, szErrorMsg, MSG_LNG, &pcbErrorMsg);
        if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO) {
        switch (rc) {
        case SQL_NO_DATA_FOUND:
        fp=fopen(errorName,"a");
        fprintf(fp,"SQLERROR() couldn't find text, RC=%d\n", rc);
        fclose(fp);
        break;
        case SQL_ERROR:
        fp=fopen(errorName,"a");
        fprintf(fp,"SQLERROR() couldn't access text, RC=%d\n", rc);
        fclose(fp);
        break;
        case SQL_INVALID_HANDLE:
        fp=fopen(errorName,"a");
        fprintf(fp,"SQLERROR() had invalid handle, RC=%d\n", rc);
        fclose(fp);
        break;
        default:
        fp=fopen(errorName,"a");
        fprintf(fp,"SQLERROR() unknown return code, RC=%d\n", rc);
        fclose(fp);
        break;
        }
        } else if (pfNativeError == 1205) {
        retcode = DEADLOCK;
#ifdef DEBUG
        fp=fopen(errorName,"a");
        fprintf(fp,"ODBCError: %d:Deadlock detected\n",pid);
        fprintf(fp,"ODBCError: STATE=%s, CODE=%ld, MSG=%s\n",
        szSqlState, pfNativeError, szErrorMsg); fclose(fp);
#endif
        } else {
        fp=fopen(errorName,"a");
        gettimeofday(&now, NULL);
        timenow = now.tv_sec;
        fprintf(fp,"ODBCError: %d: %s",pid,ctime(&timenow));
        retcode = SQL_ERROR;
        fprintf(fp,"ODBCError: STATE=%s, CODE=%ld, MSG=%s\n",
        szSqlState, pfNativeError, szErrorMsg);
        fclose(fp);
        }
return retcode;
} /* end ODBC_error function */
/*********************************************************************
/* The following function is included for completeness, but is not *
/* relevant for understanding the function of ODBC. *
/*********************************************************************/
#define MAX_NUM_PRECISION 15
/*********************************************************************
/* Define max length of char string representation of number as: *
/* = max(precision) + leading sign + E + exp sign + max exp leng *
/* = 15 + 1 + 1 + 1 + 2 * /* = 15 + 5 *
/*********************************************************************/
#define MAX_NUM_STRING_SIZE (MAX_NUM_PRECISION + 5)
UDWORD display_size (coltype, collen, colname) SWORD coltype;
UDWORD collen; UCHAR *colname;
{ FILE *fp; fp=fopen(errorName,"a");
switch (coltype) {
        case SQL_CHAR:
        case SQL_VARCHAR:
        case SQL_DATE:
        case SQL_TIMESTAMP:
        case SQL_BIT:
        return(max((int) collen, (int) strlen((char *) colname)));
        case SQL_SMALLINT:
        case SQL_INTEGER:
        case SQL_TINYINT:
        return(max((int) collen+1, (int) strlen((char *) colname)));
        case SQL_DECIMAL:
        case SQL_NUMERIC:
        return(max((int) collen+2, (int) strlen((char *) colname)));
        case SQL_REAL:
        case SQL_FLOAT:
        case SQL_DOUBLE:
        return(max((int) MAX_NUM_STRING_SIZE, (int)strlen((char *) colname)));
        case SQL_BINARY:
        case SQL_VARBINARY:
        return(max((int) 2*collen, (int) strlen((char *) colname)));
        case SQL_LONGVARBINARY:
        case SQL_LONGVARCHAR: fprintf(fp,"Unsupported datatype, %d\n",
coltype);
        return (0);
        default: fprintf(fp,"Unknown datatype, %d\n", coltype);
        return (0);
        }
/* end switch (coltype) */
        fclose(fp);
} /* end display_size function */
/*********************************************************************
/* Use K&R getline function to get an input line from stdin. *
/*********************************************************************/
int getline (char s[], int lim) {
        int c, i;
        for (i=0; i < lim-1 && (c=getchar()) != EOF && c!= '\n' ; ++i)
        s[i] = c;
        s[i] = '\0' ;
        return i;
    }
    /*============================================================
    ===========
    ** Function name: ODBCExit **
    ** Description:
    **
    **============================================================
    ===========*/
    void ODBCExit(HDBC hdbc, HSTMT hstmt) {
        SQLFreeStmt(hstmt, SQL_DROP);
        SQLDisconnect(hdbc);
        SQLFreeConnect(hdbc);
    }
    transaction_begin(u)
      int u;
    {
      char *packet;
      userNo = u;
      connect_odbc_user();
    }
    transaction_done()
    {
      ODBCExit(hdbc,hstmt);
    }
    #define INT2(p) ((short *)(p)+1)
    #define INT1(p) ((char *)(p)+3)
    void sleep_before_retry()
    {
      delay(.1);
    }
```

*client/Makefile*

```
    #*****************************************************************************
    #@(#) Version: A.10.10 $Date: 96/04/15 15:15:28 $
    #
    #(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
    #*****************************************************************************
    #
    # Makefile for compiling the client, batch-tpcc, and service code
#
OH      = ${ORACLE_HOME}
P       = ${WORK_DIR}/src
I       = $(P)/lib
L       = $(P)/lib
D       = $(P)/driver
Q       = $(P)/que
S       = $(P)/client
OPT        = -Wl,+s,-a,archive_shared +O4 +Ofastaccess +Onolimit +Oentrysched
OPT_DEBUG  = -Wl,-a,archive_shared -g
LDOPTS     = -a archive +Oprocelim +Ofastaccess
TUXEDO     = -D_HPUX_SOURCE ${ROOTDIR}/include ${OPT}
ORA_LOAD = -L${OH}/lib ${OH}/lib/osntab.o -lbench -locic -lsqlnet -lnetv2 -
lnetwork -lora -lsqlnet -lora -lnlsrtl3 -lnlsrtl -lc3v6 -lcore3 -lcore -lm -lnlsrtl3 -lnlsrtl -
lnsg -lpls -lcore3 -lnlsrtl3 -lnlsrtl -lstublm -lc -lm
LDFLAGS_SYB= ${OPT} ${L}/tpc_lib.a -L${SYBASE}/lib -lsybdb -lm
#LDFLAGS_SQL= ${OPT_DEBUG} ${L}/tpc_lib.a -
L/vsbuild/v1.10/build/com/obj/inst -lodbc -lm
#LDFLAGS_SQL= ${OPT} ${L}/tpc_lib.a -L/vsbuild/v1.10/build/com/obj/inst -
lodbc -lm
LDFLAGS_SQL= ${OPT} ${L}/tpc_lib.a -L/opt/odbc/lib -lodbc -lm
LDFLAGS_ORA= ${OPT} ${L}/tpc_lib.a ${ORA_LOAD}
ORA_INCLUDE= -I ${OH}/rdbms/demo
SYB_INCLUDE= -I /usr/sys/include
VIS_INCLUDE= -I ${VISIGENIC}/include
TUX_INCLUDE= -I ${ROOTDIR}/include
INCLUDE    = -I. -I$L
CFLAGS   = ${OPT} ${INCLUDE} ${TUX_INCLUDE}
CFLAGS_SYB = -Aa -DVG_UNIX ${OPT} ${INCLUDE} ${TUX_INCLUDE}
${SYB_INCLUDE} ${VIS_INCLUDE}
CFLAGS_ORA = -Aa -D_HPUX_SOURCE ${OPT} ${INCLUDE}
${ORA_INCLUDE} ${TUX_INCLUDE}
#CFLAGS_SQL = -Aa -Dunix -D_HPUX_SOURCE -DVG_UNIX ${OPT_DEBUG}
${INCLUDE} ${TUX_INCLUDE} ${SQL_INCLUDE} ${VIS_INCLUDE}
CFLAGS_SQL = -Aa -Dunix -D_HPUX_SOURCE -DVG_UNIX ${OPT}
${INCLUDE} ${TUX_INCLUDE} ${SQL_INCLUDE} ${VIS_INCLUDE}
PROGRAMS     = client service startup client_batch msg_server raw
PROGRAMS_SYB = client service_syb startup client_batch_syb
msg_server_syb raw
PROGRAMS_ORA = client service_ora startup client_batch_ora msg_server_ora
raw
PROGRAMS_SQL = client service_sql startup client_batch_sql msg_server_sql
raw
oracle: ${PROGRAMS_ORA}
        mv ${PROGRAMS} ${WORK_DIR}/bin/
sybase: ${PROGRAMS_SYB}
        mv ${PROGRAMS} ${WORK_DIR}/bin/
```

```
sqlserver: ${PROGRAMS_SQL}
        mv ${PROGRAMS} ${WORK_DIR}/bin/
all: ${PROGRAMS}
${S}/sybase/transaction.o: ${S}/sybase/transaction.c
        $(CC) ${CFLAGS_SYB} $(L)/tpc_lib.a -c ${S}/sybase/transaction.c;
${S}/sqlserver/transactionb.o: ${S}/sqlserver/transactionb.c
        $(CC) ${CFLAGS_SQL} $(L)/tpc_lib.a -c ${S}/sqlserver/transactionb.c;
${S}/oracle/transaction.o: ${S}/oracle/transaction.c
        $(CC) ${CFLAGS_ORA} $(L)/tpc_lib.a -c ${S}/oracle/transaction.c;
raw: raw.o
        cc ${CFLAGS_SQL} raw.o $(L)/tpc_lib.a -o raw
raw.o: raw.c
        $(CC) ${CFLAGS_SQL} $(L)/tpc_lib.a -c raw.c
startup: startup.o $(L)/tpc_lib.a
        cc ${CFLAGS} startup.o $(L)/tpc_lib.a -o startup
        chmod a+rw startup
client: client.o  tux_transaction.o $(L)/tpc_lib.a
        ${ROOTDIR}/bin/buildclient -v -f \
        "client.o tux_transaction.o $(L)/tpc_lib.a -lm" -o client
service_syb: service.o ${S}/sybase/transaction.o $(L)/tpc_lib.a
        ${ROOTDIR}/bin/buildserver -v -b shm \
                -s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC \
                -o service \
                -f "service.o transaction.o $L/tpc_lib.a \
                   ${SYBASE}/lib/libsybdb.a -lm";
service_ora: service.o ${S}/oracle/transaction.o $(L)/tpc_lib.a
        ${ROOTDIR}/bin/buildserver -v -b shm \
                -s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC \
                -o service \
                -f 'service.o transaction.o $L/tpc_lib.a -L${OH}/lib \
            ${OH}/lib/osntab.o' \
                -l '-lbench -locic' \
                -l '-lsqlnet -lnetv2 -lnetwork -lora -lsqlnet -lora' \
                -l '-lnlsrtl3 -lnlsrtl -lc3v6 -lcore3 -lcore -lm' \
                -l '-lnlsrtl3 -lnlsrtl -lnsg -lpls -lcore3 -lnlsrtl3' \
                -l '-lnlsrtl -lstublm -lc -lm';
service_sql: service.o ${S}/sqlserver/transactionb.o $(L)/tpc_lib.a
        ${ROOTDIR}/bin/buildserver -v -b shm \
                -s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s
DVRY_SVC \
                -o service \
                -f "service.o transactionb.o $L/tpc_lib.a \
                   /vsbuild/v1.10/build/com/obj/inst/libodbc.sl"
client_batch_ora: $(D)/driver.o transaction.o $(Q)/dummy_que.o $(L)/tpc_lib.a  \
                $(L)/server_default.o
        $(CC) $(D)/driver.o transaction.o $(Q)/dummy_que.o $(L)/server_default.o
$(L)/tpc_lib.a ${LDFLAGS_ORA} -o client_batch;
client_batch_syb: $(D)/driver.o transaction.o $(Q)/dummy_que.o $(L)/tpc_lib.a  \
                $(L)/server_default.o
        $(CC) $(D)/driver.o transaction.o $(Q)/dummy_que.o $(L)/server_default.o
$(L)/tpc_lib.a ${LDFLAGS_SYB} -o client_batch;
client_batch_sql: $(D)/driver.o transactionb.o $(Q)/dummy_que.o $(L)/tpc_lib.a  \
                $(L)/server_default.o
        $(CC) $(D)/driver.o transactionb.o $(Q)/dummy_que.o $(L)/server_default.o
$(L)/tpc_lib.a ${LDFLAGS_SQL} -o client_batch;
msg_server_ora: $(Q)/msg_server.o  transaction.o $(L)/tpc_lib.a
        $(CC) $(Q)/msg_server.o transaction.o ${LDFLAGS_ORA} -o msg_server;
msg_server_syb: $(Q)/msg_server.o  transaction.o $(L)/tpc_lib.a
        $(CC) $(Q)/msg_server.o transaction.o ${LDFLAGS_SYB} -o msg_server;
msg_server_sql: $(Q)/msg_server.o  transactionb.o $(L)/tpc_lib.a
        $(CC) $(Q)/msg_server.o transactionb.o ${LDFLAGS_SQL} -o msg_server;

clean:
        rm -f *.o
clobber: clean
        rm -f ${PROGRAMS}
```

*client/service.c*

```
/******************************************************************************
 @(#) Version: A.10.10 $Date: 96/06/10 14:46:59 $
 (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;
    /*
     * 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;
     }
   }
   message("TUXEDO service %s has started\n", cmd);
   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 */
   message("TUXEDO service %s has shutdown \n", cmd);
}
```

*client/tux_transaction.c*

```
/******************************************************************************
 @(#) Version: A.10.10 $Date: 96/04/15 15:16:17 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
******************************************************************************/
#include <varargs.h>
#include <errno.h>
extern int errno;
#include "atmi.h"
#include "Uunix.h"
#include "tpcc.h"
int user;
neworder_trans *neworder_ptr;
payment_trans  *payment_ptr;
ordstat_trans *ordstat_ptr;
```

```
stocklev_trans *stocklev_ptr;
delivery_trans *delivery_ptr;
int result;
transaction_begin(u)
   int u;
   {
   /* keep track of which user we are (for error messages only) */
   user = u;
   /* attach to Tuxedo */
   if (tpinit( (TPINIT *)NULL) == -1)
      tux_error("Failed to attach to Tuxedo\n");
   /* allocate structures for each transaction */
   neworder_ptr = tpalloc("CARRAY", NULL, sizeof(neworder_trans));
   payment_ptr = tpalloc("CARRAY", NULL, sizeof(payment_trans));
   ordstat_ptr = tpalloc("CARRAY", NULL, sizeof(ordstat_trans));
   stocklev_ptr = tpalloc("CARRAY", NULL, sizeof(stocklev_trans));
   delivery_ptr = tpalloc("CARRAY", NULL, sizeof(delivery_trans));
   if (neworder_ptr == NULL || payment_ptr == NULL || ordstat_ptr == NULL
      || stocklev_ptr == NULL || delivery_ptr == NULL)
      tux_error("Unable to allocate Tuxedo memory\n");
   }
transaction_done()
   {
   if (tpterm() == -1)
      tux_error("Unable to detach from Tuxedo\n");
   }
void neworder_transaction(t)
   neworder_trans *t;
   {
   *neworder_ptr = *t;
   if (tpcall("NEWO_SVC", neworder_ptr, sizeof(neworder_trans),
          &neworder_ptr, &result, TPSIGRSTRT|TPNOTIME) == -1)
      tux_error("Tuxedo failed for neworder transaction\n");
   *t = *neworder_ptr;
   }
void payment_transaction(t)
   payment_trans *t;
   {
   *payment_ptr = *t;
   if (tpcall("PMT_SVC", payment_ptr, sizeof(payment_trans),
                  &payment_ptr, &result, TPSIGRSTRT|TPNOTIME) == -1)
      tux_error("Tuxedo failed for payment transaction\n");
   *t = *payment_ptr;
   }
void ordstat_transaction(t)
   ordstat_trans *t;
   {
   *ordstat_ptr = *t;
   if (tpcall("ORDS_SVC", ordstat_ptr, sizeof(ordstat_trans),
                  &ordstat_ptr, &result, TPSIGRSTRT|TPNOTIME) == -1)
       tux_error("Tuxedo failed for ordstat transaction\n");
   *t = *ordstat_ptr;
   }
stocklev_transaction(t)
   stocklev_trans *t;
   {
   *stocklev_ptr = *t;
   if (tpcall("STKL_SVC", stocklev_ptr, sizeof(stocklev_trans),
                  &stocklev_ptr, &result, TPSIGRSTRT|TPNOTIME) == -1)
      tux_error("Tuxedo failed for stocklev transaction\n");
   *t = *stocklev_ptr;
   }
delivery_init(u)
   int u;
   {
   }
delivery_enque(t)
   delivery_trans *t;
   {
   gettimeofday(&t->enque, NULL);
   t->status = OK;
   *delivery_ptr = *t;
   if (tpacall("DVRY_SVC", delivery_ptr, sizeof(delivery_trans),
            TPNOREPLY) == -1)
      tux_error("Tuxedo failed enqueing delivery transaction\n");
   }
delivery_done()
   {
   }
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");
exit(1);
}
```

## A.3 Driver

*driver/generate.c*

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 96/08/13 19:49:56 $

 (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 <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;

extern int no_warehouse;
extern int no_item;
extern int no_dist_pw;
extern int no_cust_pd;
extern int no_ord_pd;
extern int no_new_pd;
extern int tpcc_load_seed;

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);
        }

    /* 1% of transactions roll back. Give the last order line a bad item */
    if (RandomNumber(1, 100) == 1)
        t->item[t->O_OL_CNT - 1].OL_I_ID = -1;
    }

payment_gen(t)
    payment_trans *t;
    {

    /* home warehouse is fixed */
    t->W_ID = warehouse;

    /* Random district */
    t->D_ID = RandomNumber(1, no_dist_pw);

    /* Customer is from remote warehouse and district 15% of the time */
    t->C_W_ID = RandomWarehouse(warehouse, scale, 15);
    if (t->C_W_ID == t->W_ID)
        t->C_D_ID = t->D_ID;
    else
        t->C_D_ID = RandomNumber(1, no_dist_pw);

    /* by name 60% of the time */
    t->byname = RandomNumber(1, 100) <= 60;
    if (t->byname)
        LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1,
CLAST_CONST_C),
         t->C_LAST);
    else
        t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);

    /* amount is random from [1.00..5,000.00] */
    t->H_AMOUNT = RandomNumber(100, 500000);

    }

ordstat_gen(t)
    ordstat_trans *t;
    {

    /* home warehouse is fixed */
    t->W_ID = warehouse;

    /* district is randomly selected from warehouse */
    t->D_ID = RandomNumber(1, no_dist_pw);

    /* by name 60% of the time */
    t->byname = RandomNumber(1, 100) <= 60;
    if (t->byname)
        LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1,
CLAST_CONST_C),
         t->C_LAST);
    else
        t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);
    }

delivery_gen(t)
    delivery_trans *t;
    {
    t->W_ID = warehouse;
    t->O_CARRIER_ID = RandomNumber(1,10);
    }

stocklev_gen(t)
    stocklev_trans *t;
    {
    t->W_ID = warehouse;
    t->D_ID = district;
    t->threshold = RandomNumber(10, 20);
    }

int get_trans_type()
/************************************************************************
 * get_trans_type selects a transaction according to the weighted average
 * For TPC-C rev 3.0 and less and TPC-C rev 3.2 this is:
 *       new-order :  ???
 *       payment   : 43.0%
 *       order stat:  4.0%
 *       delivery  :  4.0%
 *       stock     :  4.0%
 ************************************************************************/
    {
    static double weight[] = { 0.0,  0.0, .4305, .0405, .0405, .0405};
    double drand48();
    int type;
    double r;

    /* choose a random number between 0.0 and 1.0 */
    if (trans_type == 0) {
        r = drand48();

        /*
         * 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 {
      /* user wants only a certain type (say all stocklevel) so do that
         instead */
      type = trans_type;
  }
  /* return the value of the selected card, or NEWORDER if none selected */
  return type;
  }
```

*lib/date.c*

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 96/04/02 16:26:09 $

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

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

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

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

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

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

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


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

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


#define Feb29 (31+29-1)

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

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

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

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

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

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

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

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

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

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

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

*lib/errlog.c*

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 96/06/11 10:46:41 $

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

#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
extern int errno;

int userid;



error(format, args)
/************************************************************************
error formats a message and outputs it to a standard location (stderr for now)
************************************************************************/
```

```
    char *format;
    int args;
    {
    va_list argptr;

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

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

    /* done */
    va_end(argptr);

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


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

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

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

    /* done */
    va_end(argptr);

    /* display the system error message */
    message("    System error message: %s\n", strerror(save_errno));
    /* take an error exit */
    exit(1);
    }



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

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

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

    /* done */
    va_end(argptr);
    }



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

*****************************************************/
    char *format;
    va_list argptr;
{
  char buf[3*1024];

  /* format a message id */
  sprintf(buf, "User %-6d Pid %-6d ", 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;
  {
  int  fd;
  char  *fname;

  /* 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 */
  fd= open(fname, O_WRONLY | O_CREAT, 0666);
  if (fd < 0)
    console_error("Can't open tpc error log file 'ERROR_LOG'\n");
  lockf(fd, F_LOCK, 0);

  /* write the new text at the end of the file */
  lseek(fd, 0, SEEK_END);
  write(fd, buf, size);

  /* release the file */
  /* fsync(fd); */
  lockf(fd, F_ULOCK, 0);
  close(fd);
  }
```

```
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: 96/04/02 16:26:25 $

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

```
    /******************************************************************************
     @(#) Version: A.10.10 $Date: 96/04/02 16:26:25 $

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



    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/iobuf.h*

```
/******************************************************************************
 @(#) Version: A.10.10 $Date: 96/08/06 19:33:00 $

 (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/random.c*

```
/******************************************************************************
 @(#) Version: A.10.10 $Date: 96/05/20 11:05:46 $

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

double drand48();
```

```
void MakeZip(zip)
/******************************************************************
MakeZip makes a zip code string 000011111 to 999911111
******************************************************************/
    TEXT zip[9+1];
{
    int i;
    MakeNumberString(4, 4, zip);
    for (i=4; i<9; i++)
        zip[i] = '1';
    zip[9] = '\0';
}


void MakeAddress(str1, str2, city, state, zip)
    TEXT str1[20+1];
    TEXT str2[20+1];
    TEXT city[20+1];
    TEXT state[2+1];
    TEXT zip[9+1];
{
    MakeAlphaString(10,20,str1);
    MakeAlphaString(10,20,str2);
    MakeAlphaString(10,20,city);
    MakeAlphaString(2,2,state);
    MakeZip(zip);
}

void LastName(num, name)
/***************************************************************
Lastname generates a lastname from a number.
***************************************************************/
    unsigned int num;
    TEXT name[20+1];
    {
    int i;
    static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
                        "ESE", "ANTI", "CALLY", "ATION", "EING"};

    strcpy(name, n[(num/100)%10]);
    strcat(name, n[(num/10) %10]);
    strcat(name, n[(num/1)  %10]);
    }

int MakeNumberString(min, max, num)
    int min;
    int max;
    TEXT num[];
    {
    static 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;
    }

int MakeAlphaString(min, max, str)
    int min;
    int max;
    TEXT str[];
    {
    static char character[] = "abcdefghijklmnopqrstuvwxyz";
    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 Original(str)
    TEXT str[];
    {
    int pos;
    int len;

    len = strlen(str);
    if (len < 8) return;

    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’;
    }

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();

    x = -log(1.0-drand48()) * mean;

    return x;
    }


void Randomize()
    {
    srand48(time(0)+getpid());
    }


int RandomNumber(min, max)
/**************************************************************************
RandomNumber selects a uniform random number from min to max inclusive
**************************************************************************/
    int min;
    int max;
    {
    int r;
    r = (int)(drand48() * (max - min + 1)) + min;
    return r;
    }


int NURandomNumber(a, min, max, c)
/************************************************************************
NURandomNumber selects a non-uniform random number
************************************************************************/
    int a;
    int min;
    int max;
    int c;
    {
    int r;

    r = ((RandomNumber(0, a) | RandomNumber(min, max)) + c)
        % (max - min + 1) + min;

    return r;
    }
```

*lib/random.h*

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 96/05/20 11:06:40 $

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

double drand48();

void MakeZip();
void MakeAddress();
void LastName();
int MakeNumberString();
ID RandomWarehouse();
int MakeAlphaString();
void Original();
void RandomPermutation();
void RandomDelay();
double exponential();
void Randomize();
int RandomNumber();
int NURandomNumber();
#endif
```

*lib/results_file.c*

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 96/08/06 11:56:24 $

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

## Build

*diskinit.sql*

```
/*  TPC-C Benchmark Kit                                    */
/*                                                         */
/*  DISKINIT.SQL                                           */
/*                                                         */
/*  This script is used create the 300 warehouse database
devices  */
use master
go
disk init name = "c_log1_dev",
   physname = "q:",
   vdevno   = 14,
   size     = 2048000
go
disk init name = "c_log2_dev",
   physname = "r:",
   vdevno   = 15,
   size     = 2048000
go
disk init name = "c_ordln1_dev",
   physname = "d:",
   vdevno   = 18,
   size     = 1198080
go
disk init name = "c_ordln2_dev",
   physname = "e:",
   vdevno   = 19,
   size     = 1198080
go
disk init name = "c_ordln3_dev",
   physname = "f:",
   vdevno   = 20,
   size     = 1198080
go
disk init name = "c_ordln4_dev",
   physname = "h:",
   vdevno   = 21,
   size     = 1198080
go
disk init name = "c_cs1_dev",
   physname = "i:",
   vdevno   = 22,
   size     = 2688237
go
disk init name = "c_cs2_dev",
   physname = "j:",
   vdevno   = 23,
   size     = 2688237
go
disk init name = "c_cs3_dev",
   physname = "k:",
   vdevno   = 24,
   size     = 2688237
go
disk init name = "c_cs4_dev",
   physname = "l:",
   vdevno   = 25,
   size     = 2688237
go
disk init name = "c_misc1_dev",
   physname = "m:",
   vdevno   = 26,
   size     = 202500
go
disk init name = "c_misc2_dev",
   physname = "n:",
   vdevno   = 27,
   size     = 202499
go
disk init name = "c_misc3_dev",
   physname = "o:",
   vdevno   = 28,
   size     = 693580
```

```
go
disk init name = "c_misc4_dev",
   physname = "p:",
   vdevno   = 29,
   size     = 693580
go
```

*createdb.sql*

```
/*  TPC-C Benchmark Kit                             */
/*                                                  */
/*  CREATEDB.SQL                                    */
/*                                                  */
/*  This script is used to create the 300 warehouse
database    */
use master
go
if exists ( select name from sysdatabases where name =
"tpcc" )
   drop database tpcc
go
create database tpcc on
   c_ordln1_dev   = 1170,
   c_ordln2_dev   = 1170,
   c_ordln3_dev   = 1170,
   c_ordln4_dev   = 1170,
   c_ordln1_dev   = 1170,
   c_ordln2_dev   = 1170,
   c_ordln3_dev   = 1170,
   c_ordln4_dev   = 1170,
   c_cs1_dev    = 1312,
   c_cs2_dev    = 1312,
   c_cs3_dev    = 1312,
   c_cs4_dev    = 1312,
   c_cs1_dev    = 1312,
   c_cs2_dev    = 1312,
   c_cs3_dev    = 1312,
   c_cs4_dev    = 1312,
   c_cs1_dev    = 1312,
   c_cs2_dev    = 1312,
   c_cs3_dev    = 1312,
   c_cs4_dev    = 1312,
   c_cs1_dev    = 1312,
   c_cs2_dev    = 1312,
   c_cs3_dev    = 1312,
   c_cs4_dev    = 1312,
   c_misc1_dev  = 396,
   c_misc2_dev  = 396,
   c_misc3_dev  = 1354,
   c_misc4_dev  = 1354
       log on c_log1_dev =  4000,
   c_log2_dev =  4000
go
```

*segment.sql*

```
/*  TPC-C Benchmark Kit                                  */
/*                                                       */
/*  SEGMENT.SQL                                          */
/*                                                       */
/*  This script is used to create the database segments  */
use tpcc
go
exec sp_addsegment misc_seg, c_misc1_dev
exec sp_extendsegment misc_seg, c_misc2_dev
exec sp_extendsegment misc_seg, c_misc3_dev
exec sp_extendsegment misc_seg, c_misc4_dev
exec sp_addsegment ordln_seg, c_ordln1_dev
exec sp_extendsegment ordln_seg, c_ordln2_dev
exec sp_extendsegment ordln_seg, c_ordln3_dev
exec sp_extendsegment ordln_seg, c_ordln4_dev
exec sp_addsegment cs_seg, c_cs1_dev
exec sp_extendsegment cs_seg, c_cs2_dev
exec sp_extendsegment cs_seg, c_cs3_dev
exec sp_extendsegment cs_seg, c_cs4_dev
go
```

*tables.sql*

```
/*  TPC-C Benchmark
Kit                                                  */
/*
             */
/*
TABLES.SQL
         */
/*
             */
/*  Creates TPC-C tables
(seg)                                           */
use tpcc
go
checkpoint
go
if exists ( select name from sysobjects where name =
'warehouse' )
   drop table warehouse
go
create table warehouse
(
   w_idsmallint,
   w_namechar(10),
   w_street_1char(20),
   w_street_2char(20),
   w_citychar(20),
   w_statechar(2),
   w_zipchar(9),
   w_taxnumeric(4,4),
   w_ytdnumeric(12,2)
) on misc_seg
go
if exists ( select name from sysobjects where name =
'district' )
   drop table district
go
create table district
(
   d_idtinyint,
   d_w_idsmallint,
   d_namechar(10),
   d_street_1char(20),
   d_street_2char(20),
   d_citychar(20),
   d_statechar(2),
   d_zipchar(9),
   d_taxnumeric(4,4),
   d_ytdnumeric(12,2),
   d_next_o_idint
) on misc_seg
go
if exists ( select name from sysobjects where name =
'customer' )
   drop table customer
go
create table customer
(
   c_idint,
   c_d_idtinyint,
   c_w_idsmallint,
   c_firstchar(16),
   c_middlechar(2),
   c_lastchar(16),
   c_street_1char(20),
   c_street_2char(20),
   c_citychar(20),
   c_statechar(2),
   c_zipchar(9),
   c_phonechar(16),
   c_sincedatetime,
   c_creditchar(2),
   c_credit_limnumeric(12,2),
   c_discountnumeric(4,4),
   c_balancenumeric(12,2),
   c_ytd_paymentnumeric(12,2),
   c_payment_cntsmallint,
   c_delivery_cntsmallint,
   c_data_1char(250),
   c_data_2char(250)
) on cs_seg
go
if exists ( select name from sysobjects where name =
'history' )
   drop table history
go
```

```
create table history
(
    h_c_idint,
    h_c_d_idtinyint,
    h_c_w_idsmallint,
    h_d_idtinyint,
    h_w_idsmallint,
    h_datedatetime,
    h_amountnumeric(6,2),
    h_datachar(24)
) on misc_seg
go
if exists ( select name from sysobjects where name =
'new_order' )
    drop table new_order
go
create table new_order
(
    no_o_idint,
    no_d_idtinyint,
    no_w_idsmallint
) on misc_seg
go
if exists ( select name from sysobjects where name =
'orders' )
    drop table orders
go
create table orders
(
    o_idint,
    o_d_idtinyint,
    o_w_idsmallint,
    o_c_idint,
    o_entry_ddatetime,
    o_carrier_idtinyint,
    o_ol_cnttinyint,
    o_all_localtinyint
) on misc_seg
go
if exists ( select name from sysobjects where name =
'order_line' )
    drop table order_line
go
create table order_line
(
    ol_o_idint,
    ol_d_idtinyint,
    ol_w_idsmallint,
    ol_numbertinyint,
    ol_i_idint,
    ol_supply_w_idsmallint,
    ol_delivery_ddatetime,
    ol_quantitysmallint,
    ol_amountnumeric(6,2),
    ol_dist_infochar(24)
) on ordln_seg
go
if exists ( select name from sysobjects where name = 'item'
)
    drop table item
go
create table item
(
    i_idint,
    i_im_idint,
    i_namechar(24),
    i_pricenumeric(5,2),
    i_datachar(50)
) on misc_seg
go
if exists ( select name from sysobjects where name =
'stock' )
    drop table stock
go
create table stock
(
    s_i_idint,
    s_w_idsmallint,
    s_quantitysmallint,
    s_dist_01char(24),
    s_dist_02char(24),
    s_dist_03char(24),
    s_dist_04char(24),
    s_dist_05char(24),
        s_dist_06char(24),
        s_dist_07char(24),
        s_dist_08char(24),
        s_dist_09char(24),
        s_dist_10char(24),
        s_ytdint,
        s_order_cntsmallint,
        s_remote_cntsmallint,
        s_datachar(50)
    ) on cs_seg
    go
```

*idxwarcl.sql*

```
    /*  TPC-C Benchmark
    Kit                                                    */
    /*
                 */
    /*
    IDXWARCL.SQL
             */
    /*
                 */
    /*  Creates clustered index on warehouse
    (seg)                          */
    use tpcc
    go
    if exists ( select name from sysindexes where name =
    'warehouse_c1' )
       drop index warehouse.warehouse_c1
    go
    select getdate()
    go
    create unique clustered index warehouse_c1 on
    warehouse(w_id)
       with fillfactor=1 on misc_seg
    go
    select getdate()
    go
```

*idxdiscl.sql*

```
    /*  TPC-C Benchmark
    Kit                                                    */
    /*
                 */
    /*
    IDXDISCL.SQL
             */
    /*
                 */
    /*  Creates clustered index on district
    (seg)                          */
    use tpcc
    go
    if exists ( select name from sysindexes where name =
    'district_c1' )
       drop index district.district_c1
    go
    select getdate()
    go
    create unique clustered index  district_c1 on
    district(d_w_id, d_id)
       with fillfactor=1 on misc_seg
    go
    select getdate()
    go
```

*idxcuscl.sql*

```
    /*  TPC-C Benchmark
    Kit                                                    */
    /*
                 */
    /*
    IDXCUSCL.SQL
             */
    /*
                 */
    /*  Creates clustered index on customer
    (seg)                          */
    use tpcc
    go
```

```
if exists ( select name from sysindexes where name =
'customer_c1' )
   drop index customer.customer_c1
go
select getdate()
go
create unique clustered index customer_c1 on
customer(c_w_id, c_d_id, c_id)
   with sorted_data on cs_seg
go
select getdate()
go
```

*idxodlcl.sql*

```
/*  TPC-C Benchmark
Kit                                                   */
/*
             */
/*
IDXODLCL.SQL
         */
/*
             */
/*  Creates clustered index on order-line
(seg)                         */
use tpcc
go
if exists ( select name from sysindexes where  name =
'order_line_c1' )
   drop index order_line.order_line_c1
go
select getdate()
go
create unique clustered index order_line_c1 on
order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
   with sorted_data on ordln_seg
go
select getdate()
go
```

*idxordcl.sql*

```
/*  TPC-C Benchmark
Kit                                                   */
/*
             */
/*
IDXORDCL.SQL
         */
/*
             */
/*  Creates clustered index on orders
(seg)                              */
use tpcc
go
if exists ( select name from sysindexes where  name =
'orders_c1' )
   drop index orders.orders_c1
go
select getdate()
go
create unique clustered index orders_c1 on orders(o_w_id,
o_d_id, o_id)
   with sorted_data on misc_seg
go
select getdate()
go
```

*idxnodcl.sql*

```
/*  TPC-C Benchmark
Kit                                                   */
/*
             */
/*
IDXNODCL.SQL
         */
/*
             */
/*  Creates clustered index on new-order
(seg)                           */
use tpcc
go
```

```
if exists ( select name from sysindexes where  name =
'new_order_c1' )
   drop index new_order.new_order_c1
go
select getdate()
go
create unique clustered index new_order_c1 on
new_order(no_w_id, no_d_id, no_o_id)
   with sorted_data on misc_seg
go
select getdate()
go
```

*idxstkcl.sql*

```
/*  TPC-C Benchmark
Kit                                                  */
/*
             */
/*
IDXSTKCL.SQL
         */
/*
             */
/*  Creates clustered index on stock
(seg)                                */
use tpcc
go
if exists ( select name from sysindexes where  name =
'stock_c1' )
   drop index stock.stock_c1
go
select getdate()
go
create unique clustered index stock_c1 on stock(s_i_id,
s_w_id)
   with sorted_data on cs_seg
go
select getdate()
go
```

*idxitmcl.sql*

```
/*  TPC-C Benchmark
Kit                                                  */
/*
             */
/*
IDXITMCL.SQL
         */
/*
             */
/*  Creates clustered index on item
(seg)                                */
use tpcc
go
if exists ( select name from sysindexes where  name =
'item_c1' )
   drop index item.item_c1
go
select getdate()
go
create unique clustered index item_c1 on item(i_id)
   with sorted_data on misc_seg
go
select getdate()
go
```

*idxcusnc.sql*

```
/*  TPC-C Benchmark
Kit                                                  */
/*
             */
/*
IDXCUSNC.SQL
         */
/*
             */
/*  Creates non-clustered index on customer
(seg)                          */
use tpcc
go
```

```
if exists ( select name from sysindexes where  name =
'customer_nc1' )
   drop index customer.customer_nc1
go
select getdate()
go
create unique nonclustered index customer_nc1 on
customer(c_w_id, c_d_id, c_last, c_first, c_id)
   on cs_seg
go
select getdate()
go
```

*dbopt1.sql*

```
/*  TPC-C Benchmark
Kit                                                   */
/*
             */
/*
DBOPT1.SQL
         */
/*
             */
/*  Set database options for database
load                             */
use master
go
sp_dboption tpcc,'select into/bulkcopy',true
go
sp_dboption tpcc,'trunc. log on chkpt.',true
go
use tpcc
go
checkpoint
go
use tpcc_admin
go
sp_dboption tpcc,'trunc. log on chkpt.',true
go
```

*tpccirl.sql*

```
/*  TPC-C Benchmark
Kit                                                           */
/*
                  */
/*
TPCCIRL.SQL
              */
/*
                  */
/*  This script file sets the insert row lock option on
selected tables       */
use tpcc
go
exec sp_tableoption "history","insert row lock",true
exec sp_tableoption "new_order","insert row lock",true
exec sp_tableoption "orders","insert row lock",true
exec sp_tableoption "order_line","insert row lock",true
go
```

*neword.sql*

```
/*  File:
NEWORD.SQL
*/
/*              Microsoft TPC-C Kit Ver.
3.00.000                           */
/*              Audited 08/23/96, By Francois
Raab                          */
/*
               */
/*              Copyright Microsoft,
1996                                */
/*
               */
/*  Purpose:    New-Order transaction for Microsoft TPC-C
Benchmark Kit    */
/*  Author:     Damien
Lindauer                                          */
/*
damienl@Microsoft.com
*/
use tpcc
go
/* new-order transaction stored procedure */
```

```
if exists ( select name from sysobjects where name =
"tpcc_neworder" )
    drop procedure tpcc_neworder
go
create proc tpcc_neworder
    @w_id           smallint,
    @d_id           tinyint,
    @c_id           int,
    @o_ol_cnt       tinyint,
    @o_all_local    tinyint,
    @i_id1  int = 0, @s_w_id1 smallint = 0, @ol_qty1
smallint = 0,
    @i_id2  int = 0, @s_w_id2 smallint = 0, @ol_qty2
smallint = 0,
    @i_id3  int = 0, @s_w_id3 smallint = 0, @ol_qty3
smallint = 0,
    @i_id4  int = 0, @s_w_id4 smallint = 0, @ol_qty4
smallint = 0,
    @i_id5  int = 0, @s_w_id5 smallint = 0, @ol_qty5
smallint = 0,
    @i_id6  int = 0, @s_w_id6 smallint = 0, @ol_qty6
smallint = 0,
    @i_id7  int = 0, @s_w_id7 smallint = 0, @ol_qty7
smallint = 0,
    @i_id8  int = 0, @s_w_id8 smallint = 0, @ol_qty8
smallint = 0,
    @i_id9  int = 0, @s_w_id9 smallint = 0, @ol_qty9
smallint = 0,
    @i_id10 int = 0, @s_w_id10 smallint = 0, @ol_qty10
smallint = 0,
    @i_id11 int = 0, @s_w_id11 smallint = 0, @ol_qty11
smallint = 0,
    @i_id12 int = 0, @s_w_id12 smallint = 0, @ol_qty12
smallint = 0,
    @i_id13 int = 0, @s_w_id13 smallint = 0, @ol_qty13
smallint = 0,
    @i_id14 int = 0, @s_w_id14 smallint = 0, @ol_qty14
smallint = 0,
    @i_id15 int = 0, @s_w_id15 smallint = 0, @ol_qty15
smallint = 0

as
declare @w_tax          numeric(4,4),
        @d_tax          numeric(4,4),
            @c_last         char(16),
            @c_credit       char(2),
            @c_discount     numeric(4,4),
            @i_price        numeric(5,2),
            @i_name         char(24),
            @i_data         char(50),
            @o_entry_d      datetime,
            @remote_flag    int,
            @s_quantity     smallint,
            @s_data         char(50),
            @s_dist         char(24),
    @li_no          int,
    @o_idint,
    @commit_flag    tinyint,
            @li_id          int,
            @li_s_w_id      smallint,
            @li_qty         smallint,
    @ol_numberint,
    @c_id_localint
begin
    begin transaction n
    /* get order date */
    select @o_entry_d = getdate()
    /* get district tax and next availible order id and
update */
    update district
    set @d_tax      = d_tax,
        @o_id       = d_next_o_id,
    d_next_o_id = d_next_o_id + 1
        where d_w_id = @w_id and
              d_id   = @d_id
    /*  process orderlines */
    select  @li_no = 0
    /*  set commit flag */
    select @commit_flag = 1
    while (@li_no < @o_ol_cnt)
    begin
        select @li_no = @li_no + 1
        /*  Set i_id, s_w_id, and qty for this lineitem */
        select @li_id = case @li_no
                when 1 then @i_id1
    when 2 then @i_id2
    when 3 then @i_id3
    when 4 then @i_id4
    when 5 then @i_id5
    when 6 then @i_id6
    when 7 then @i_id7
    when 8 then @i_id8
    when 9 then @i_id9
    when 10 then @i_id10
    when 11 then @i_id11
    when 12 then @i_id12
    when 13 then @i_id13
    when 14 then @i_id14
    when 15 then @i_id15
    end
select @li_s_w_id = case @li_no
    when 1 then @s_w_id1
    when 2 then @s_w_id2
    when 3 then @s_w_id3
    when 4 then @s_w_id4
    when 5 then @s_w_id5
    when 6 then @s_w_id6
    when 7 then @s_w_id7
    when 8 then @s_w_id8
    when 9 then @s_w_id9
    when 10 then @s_w_id10
    when 11 then @s_w_id11
    when 12 then @s_w_id12
    when 13 then @s_w_id13
    when 14 then @s_w_id14
    when 15 then @s_w_id15
    end
select @li_qty = case  @li_no
    when 1 then @ol_qty1
    when 2 then @ol_qty2
    when 3 then @ol_qty3
    when 4 then @ol_qty4
    when 5 then @ol_qty5
    when 6 then @ol_qty6
    when 7 then @ol_qty7
    when 8 then @ol_qty8
```

```
        when 9 then @ol_qty9
        when 10 then @ol_qty10
        when 11 then @ol_qty11
        when 12 then @ol_qty12
        when 13 then @ol_qty13
        when 14 then @ol_qty14
        when 15 then @ol_qty15
        end
        /* get item data (no one updates item) */
        select @i_price = i_price,
               @i_name  = i_name,
               @i_data  = i_data
        from item (tablock holdlock)
        where i_id = @li_id
       /* if there actually is an item with this id, go to
work */

      if (@@rowcount > 0)
       begin
            update stock set s_ytd        = s_ytd + @li_qty,
                             @s_quantity  = s_quantity,
                             s_quantity   = s_quantity -
@li_qty +
                                 case when (s_quantity -
@li_qty < 10) then 91 else 0 end,
                             s_order_cnt  = s_order_cnt + 1,
                             s_remote_cnt = s_remote_cnt +
case
                                 when (@li_s_w_id = @w_id)
then 0 else 1 end,
                             @s_data      = s_data,
                             @s_dist      = case @d_id
                        when 1  then s_dist_01
                            when 2  then s_dist_02
                            when 3  then s_dist_03
                            when 4  then s_dist_04
                            when 5  then s_dist_05
                            when 6  then s_dist_06
                            when 7  then s_dist_07
                            when 8  then s_dist_08
                            when 9  then s_dist_09
                            when 10 then s_dist_10
```

```
                                    end
            where s_i_id = @li_id and
          s_w_id = @li_s_w_id
    /* insert order_line data (using data from item and
stock) */
            insert into order_line
values(@o_id,              /* from district update */
                                          @d_id,      /*
input param          */

@w_id,              /* input param          */

@li_no,             /* orderline number     */

@li_id,             /* lineitem id          */

@li_s_w_id,         /* lineitem warehouse   */
                                          "jan 1,
1900",      /* constant             */

@li_qty,            /* lineitem qty         */
                                          @i_price *
@li_qty, /* ol_amount            */

@s_dist)            /* from stock           */

            /* send line-item data to client */

        select @i_name,
                   @s_quantity,
                   b_g = case when (
(patindex("%ORIGINAL%",@i_data) > 0) and

(patindex("%ORIGINAL%",@s_data) > 0) )
                         then "B" else "G" end,
                   @i_price,
                   @i_price * @li_qty

        end
        else
    begin

        /*  no item found - triggers rollback condition
*/
```

```
    select "",0,"",0,0
    select @commit_flag = 0

    end
    end
    /* get customer last name, discount, and credit rating
*/
    select @c_last     = c_last,
           @c_discount = c_discount,
           @c_credit   = c_credit,
       @c_id_local = c_id
    from customer holdlock
    where c_id   = @c_id and
          c_w_id = @w_id and
          c_d_id = @d_id
    /* insert fresh row into orders table */
    insert into orders values (@o_id,
                           @d_id,
                    @w_id,
              @c_id_local,
              @o_entry_d,
              0,
              @o_ol_cnt,
              @o_all_local)
    /* insert corresponding row into new-order table */
    insert into new_order values (@o_id,
      @d_id,
      @w_id)
    /*  select warehouse tax */
    select @w_tax = w_tax
    from warehouse holdlock
    where w_id = @w_id
    if (@commit_flag = 1)
    commit transaction n
    else
    /* all that work for nuthin!!! */
    rollback transaction n
    /* return order data to client */
    select @w_tax,
           @d_tax,
        @o_id,
        @c_last,
        @c_discount,
        @c_credit,
        @o_entry_d,
        @commit_flag
end
go
```

*payment.sql*

```
/*  File:
PAYMENT.SQL
*/
/*              Microsoft TPC-C Kit Ver.
3.00.000                          */
/*              Audited 08/23/96, By Francois
Raab                         */
/*
               */
/*              Copyright Microsoft,
1996                                  */
/*
               */
/*  Purpose:    Payment transaction for Microsoft TPC-C
Benchmark Kit      */
/*  Author:     Damien
Lindauer                                          */
/*
damienl@Microsoft.com
*/
use tpcc
go
if exists (select name from sysobjects where name =
"tpcc_payment" )
   drop procedure tpcc_payment
go
create proc tpcc_payment @w_id           smallint,
     @c_w_id         smallint,
     @h_amount       numeric(6,2),
     @d_id           tinyint,
     @c_d_id         tinyint,
     @c_id           int,
     @c_last         char(16) = ""


as
declare  @w_street_1     char(20),
         @w_street_2     char(20),
         @w_city         char(20),
         @w_state        char(2),
         @w_zip          char(9),
         @w_name         char(10),
         @d_street_1     char(20),
         @d_street_2     char(20),
         @d_city         char(20),
         @d_state        char(2),
         @d_zip          char(9),
         @d_name         char(10),
         @c_first        char(16),
         @c_middle       char(2),
         @c_street_1     char(20),
         @c_street_2     char(20),
         @c_city         char(20),
         @c_state        char(2),
         @c_zip          char(9),
         @c_phone        char(16),
         @c_since        datetime,
         @c_credit       char(2),
         @c_credit_lim   numeric(12,2),
         @c_balance      numeric(12,2),
         @c_discount     numeric(4,4),
         @data1          char(250),
         @data2          char(250),
         @c_data_1       char(250),
         @c_data_2       char(250),
         @datetime       datetime,
         @w_ytd          numeric(12,2),
         @d_ytd          numeric(12,2),
         @cnt            smallint,
         @val            smallint,
         @screen_data char(200),
     @d_id_local tinyint,
     @w_id_local smallint,
     @c_id_local int
select @screen_data = ""
begin tran p

    /* get payment date */
    select @datetime = getdate()

    if (@c_id = 0)
    begin
    /* get customer id and info using last name */
    select @cnt = count(*)
    from customer holdlock
    where c_last = @c_last and
          c_w_id = @c_w_id and
          c_d_id = @c_d_id

    select @val = (@cnt + 1) / 2
    set rowcount @val
    select @c_id = c_id
    from customer holdlock
    where c_last = @c_last and
          c_w_id = @c_w_id and
          c_d_id = @c_d_id
    order by c_w_id, c_d_id, c_last, c_first
    set rowcount 0
    end

    /* get customer info and update balances */

    update customer set
    @c_balance     = c_balance = c_balance - @h_amount,
    c_payment_cnt  = c_payment_cnt + 1,
    c_ytd_payment  = c_ytd_payment + @h_amount,
    @c_first       = c_first,
    @c_middle      = c_middle,
        @c_last        = c_last,
        @c_street_1    = c_street_1,
    @c_street_2    = c_street_2,
    @c_city        = c_city,
    @c_state       = c_state,
    @c_zip         = c_zip,
```

```
@c_phone       = c_phone,
@c_credit      = c_credit,
@c_credit_lim  = c_credit_lim,
@c_discount    = c_discount,
@c_since       = c_since,
@data1         = c_data_1,
@data2         = c_data_2,
@c_id_local    = c_id
where c_id   = @c_id and
      c_w_id = @c_w_id and
      c_d_id = @c_d_id

/* if customer has bad credit get some more info */
if (@c_credit = "BC")
begin
/* compute new info */
select @c_data_2 = substring(@data1,209,42) +
   substring(@data2, 1, 208)
select @c_data_1 = convert(char(5),@c_id) +
   convert(char(4),@c_d_id) +
           convert(char(5),@c_w_id) +
           convert(char(4),@d_id) +
           convert(char(5),@w_id) +
           convert(char(19),@h_amount) +
           substring(@data1, 1, 208)
/* update customer info */
update customer set
       c_data_1 = @c_data_1,
   c_data_2 = @c_data_2
where c_id   = @c_id and
  c_w_id = @c_w_id and
      c_d_id = @c_d_id
select @screen_data = substring (@c_data_1,1,200)
end



/* get district data and update year-to-date */

update district
set d_ytd       = d_ytd + @h_amount,
    @d_street_1 = d_street_1,
    @d_street_2 = d_street_2,
    @d_city     = d_city,
    @d_state    = d_state,
    @d_zip      = d_zip,
    @d_name     = d_name,
    @d_id_local = d_id
    where d_w_id = @w_id and
          d_id   = @d_id
    /* get warehouse data and update year-to-date */
    update warehouse
    set w_ytd       = w_ytd + @h_amount,
    @w_street_1 = w_street_1,
        @w_street_2 = w_street_2,
        @w_city     = w_city,
        @w_state    = w_state,
        @w_zip      = w_zip,
        @w_name     = w_name,
    @w_id_local = w_id
    where w_id = @w_id

    /* create history record */

    insert into history values  (@c_id_local,
         @c_d_id,
     @c_w_id,
     @d_id_local,
     @w_id_local,
     @datetime,
     @h_amount,
     @w_name + "    " + @d_name)
commit tran p
/* return data to client */
select  @c_id,
    @c_last,
    @datetime,
    @w_street_1,
    @w_street_2,
    @w_city,
    @w_state,
    @w_zip,
        @d_street_1,
        @d_street_2,
        @d_city,
        @d_state,
        @d_zip,
        @c_first,
        @c_middle,
        @c_street_1,
        @c_street_2,
        @c_city,
        @c_state,
        @c_zip,
        @c_phone,
        @c_since,
        @c_credit,
        @c_credit_lim,
        @c_discount,
        @c_balance,
        @screen_data
    go
```

*ordstat.sql*

```
/*  TPC-C Benchmark
Kit                                                  */
/*
             */
/*  Module:
TRAN_3.SQL
*/
/*  Modifed by:
DamienL                                                  */
/*
             */
/*  Description: Order-Status
Transaction                                  */


use tpcc
go
```

```
if exists ( select name from sysobjects where name =
"tpcc_orderstatus" )
   drop procedure   tpcc_orderstatus
go

create proc tpcc_orderstatus @w_idsmallint,
     @d_idtinyint,
     @c_idint,
     @c_lastchar(16) = ""

as

declare @c_balancenumeric(12,2),
        @c_firstchar(16),
        @c_middlechar(2),
        @o_idint,
        @o_entry_ddatetime,
        @o_carrier_idsmallint,
        @valsmallint,
        @cntsmallint

begin tran o

   if (@c_id = 0)
   begin
   /* get customer id and info using last name */

   select @cnt = count(*)
   from customer holdlock
   where c_last = @c_last and
         c_w_id = @w_id and
           c_d_id = @d_id

   select @val = (@cnt + 1) / 2
   set rowcount @val

   select @c_id = c_id,
      @c_balance = c_balance,
      @c_first   = c_first,
      @c_last    = c_last,
      @c_middle  = c_middle
from customer holdlock
where c_last = @c_last and
      c_w_id = @w_id and
        c_d_id = @d_id
order by c_w_id, c_d_id, c_last, c_first

set rowcount 0
end

else
begin

/*  get customer info if by id*/

select @c_balance = c_balance,
       @c_first   = c_first,
       @c_middle  = c_middle,
       @c_last    = c_last
from customer holdlock
where c_id   = @c_id and
      c_d_id = @d_id and
      c_w_id = @w_id
end


/*  get order info */

select @o_id = o_id,
   @o_entry_d    = o_entry_d,
      @o_carrier_id = o_carrier_id
from orders holdlock
where o_c_id = @c_id and
      o_d_id = @d_id and
      o_w_id = @w_id

/*  select order lines for the current order */

select ol_supply_w_id,
      ol_i_id,
       ol_quantity,
       ol_amount,
          ol_delivery_d
    from order_line holdlock
    where ol_o_id = @o_id and
          ol_d_id = @d_id and
          ol_w_id = @w_id

commit tran o

/*  return data to client  */

select @c_id,
       @c_last,
       @c_first,
       @c_middle,
       @o_entry_d,
       @o_carrier_id,
       @c_balance,
       @o_id

go
```

*delivery.sql*

```
/*  File:
DELIVERY.SQL
*/
/*              Microsoft TPC-C Kit Ver.
3.00.000                          */
/*              Audited 08/23/96, By Francois
Raab                         */
/*
               */
/*              Copyright Microsoft,
1996                                  */
/*
               */
/*  Purpose:    Delivery transaction for Microsoft TPC-C
Benchmark Kit     */
/*  Author:     Damien
Lindauer                                          */
/*
damienl@Microsoft.com
*/
use tpcc
go
/* delivery transaction */
if exists (select name from sysobjects where name =
"tpcc_delivery" )
    drop procedure tpcc_delivery
go
create proc tpcc_delivery@w_id           smallint,
    @o_carrier_id   smallint
as
declare @d_id tinyint,
        @o_id int,
        @c_id int,
        @total numeric(12,2),
        @oid1 int,
        @oid2 int,
        @oid3 int,
        @oid4 int,
        @oid5 int,
        @oid6 int,
        @oid7 int,
        @oid8 int,
        @oid9 int,
        @oid10 int
select @d_id = 0
begin tran d
    while (@d_id < 10)
    begin
        select @d_id  = @d_id + 1,
               @total = 0,
               @o_id  = 0
        select @o_id = min(no_o_id)
        from new_order holdlock
        where no_w_id = @w_id and
          no_d_id = @d_id
        if (@@rowcount <> 0)
        begin
            /* claim the order for this district */
            delete new_order
            where no_w_id = @w_id and
              no_d_id = @d_id and
          no_o_id = @o_id
            /* set carrier_id on this order (and get
customer id) */
            update orders
        set o_carrier_id = @o_carrier_id,
                @c_id        = o_c_id
            where o_w_id = @w_id and
              o_d_id = @d_id and
          o_id   = @o_id
            /* set date in all lineitems for this order
(and sum amounts) */
            update order_line
        set ol_delivery_d = getdate(),
                @total        = @total + ol_amount
            where ol_w_id = @w_id and
                  ol_d_id = @d_id and
          ol_o_id = @o_id
            /* accummulate lineitem amounts for this order
into customer */

        update customer
    set c_balance      = c_balance + @total,
        c_delivery_cnt = c_delivery_cnt + 1
            where c_w_id = @w_id and
              c_d_id = @d_id and
          c_id   = @c_id
        end
        select @oid1 = case @d_id when  1  then @o_id else
@oid1 end,
               @oid2 = case @d_id when  2  then @o_id else
@oid2 end,
               @oid3 = case @d_id when  3  then @o_id else
@oid3 end,
               @oid4 = case @d_id when  4  then @o_id else
@oid4 end,
               @oid5 = case @d_id when  5  then @o_id else
@oid5 end,
               @oid6 = case @d_id when  6  then @o_id else
@oid6 end,
               @oid7 = case @d_id when  7  then @o_id else
@oid7 end,
               @oid8 = case @d_id when  8  then @o_id else
@oid8 end,
               @oid9 = case @d_id when  9  then @o_id else
@oid9 end,
               @oid10 = case @d_id when 10 then @o_id else
@oid10 end
    end
commit tran d

select @oid1,
       @oid2,
       @oid3,
       @oid4,
       @oid5,
       @oid6,
       @oid7,
       @oid8,
       @oid9,
       @oid10
go
```

*stocklev.sql*

```
/*  File:
STOCKLEV.SQL
*/
/*              Microsoft TPC-C Kit Ver.
3.00.000                          */
/*              Audited 08/23/96, By Francois
Raab                         */
/*
               */
/*              Copyright Microsoft,
1996                                  */
/*
               */
/*  Purpose:    Stock-Level transaction for Microsoft TPC-C
Benchmark Kit  */
/*  Author:     Damien
Lindauer                                          */
/*
damienl@Microsoft.com
*/
use tpcc
go
/* stock-level transaction stored procedure */
```

```
if exists (select name from sysobjects where name =
"tpcc_stocklevel" )
   drop procedure tpcc_stocklevel
go
create proc tpcc_stocklevel@w_id        smallint,
    @d_id        tinyint,
    @threshhold smallint
as
declare @o_id_low int,
        @o_id_high int
    select @o_id_low  = (d_next_o_id - 20),
           @o_id_high = (d_next_o_id - 1)
    from district
    where d_w_id = @w_id and
      d_id   = @d_id
    select count(distinct(s_i_id))
    from  stock, order_line
    where ol_w_id    =  @w_id and
          ol_d_id    =  @d_id and
          ol_o_id between @o_id_low and @o_id_high and
          s_w_id     =  ol_w_id and
          s_i_id     =  ol_i_id and
          s_quantity <  @threshhold
go
```

*dbopt2.sql*

```
/*  TPC-C Benchmark
Kit                                                   */
/*
             */
/*
DBOPT2.SQL
         */
/*
             */
/*  Reset database options after database
load                          */
use master
go
sp_dboption tpcc,'select ',false
go
sp_dboption tpcc,'trunc. ',false
go
use tpcc
go
checkpoint
go
```

*pintable.sql*

```
/*  TPC-C Benchmark
Kit                                                         */
/*
                  */
/*
PINTABLE.SQL
              */
/*
                  */
/*  This script file is used to 'pin' certain tables in the
data cache        */
use tpcc
go
exec sp_tableoption "district","pintable",true
exec sp_tableoption "warehouse","pintable",true
exec sp_tableoption "new_order","pintable",true
exec sp_tableoption "item","pintable",true
go
```

*tmakefile.x86*

```
!include $(TPC_DIR)\build\ntintel\tpc.inc

CUR_DIR = $(TPC_DIR)\src

CLIENT_EXE     = $(EXE_DIR)\client.exe
MASTER_EXE     = $(EXE_DIR)\master.exe
TPCCLDR_EXE    = $(EXE_DIR)\tpccldr.exe
DELIVERY_EXE   = $(EXE_DIR)\delivery.exe
```

```
sqlstat_EXE    = $(EXE_DIR)\sqlstat.exe

all : $(CLIENT_EXE) $(MASTER_EXE) $(TPCCLDR_EXE)
$(DELIVERY_EXE) $(sqlstat_EXE)

$(OBJ_DIR)\client.obj : $(CUR_DIR)\client.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\client.obj
$(CUR_DIR)\client.c

$(OBJ_DIR)\master.obj : $(CUR_DIR)\master.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\master.obj
$(CUR_DIR)\master.c

$(OBJ_DIR)\tpccldr.obj : $(CUR_DIR)\tpccldr.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\tpccldr.obj
$(CUR_DIR)\tpccldr.c

$(OBJ_DIR)\stats.obj : $(CUR_DIR)\stats.c $(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\stats.obj
$(CUR_DIR)\stats.c

$(OBJ_DIR)\getargs.obj : $(CUR_DIR)\getargs.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\getargs.obj
$(CUR_DIR)\getargs.c

$(OBJ_DIR)\util.obj : $(CUR_DIR)\util.c  $(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\util.obj $(CUR_DIR)\util.c

$(OBJ_DIR)\time.obj : $(CUR_DIR)\time.c $(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\time.obj $(CUR_DIR)\time.c

$(OBJ_DIR)\random.obj : $(CUR_DIR)\random.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\random.obj
$(CUR_DIR)\random.c

$(OBJ_DIR)\strings.obj : $(CUR_DIR)\strings.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\strings.obj
$(CUR_DIR)\strings.c

$(OBJ_DIR)\sqlfuncs.obj : $(CUR_DIR)\sqlfuncs.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\sqlfuncs.obj
$(CUR_DIR)\sqlfuncs.c

$(OBJ_DIR)\tran.obj : $(CUR_DIR)\tran.c  $(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\tran.obj $(CUR_DIR)\tran.c

$(OBJ_DIR)\data.obj : $(CUR_DIR)\data.c  $(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\data.obj $(CUR_DIR)\data.c

$(OBJ_DIR)\delivery.obj : $(CUR_DIR)\delivery.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\delivery.obj
$(CUR_DIR)\delivery.c

$(OBJ_DIR)\sqlstat.obj : $(CUR_DIR)\sqlstat.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\sqlstat.obj
$(CUR_DIR)\sqlstat.c

$(EXE_DIR)\client.exe : $(OBJ_DIR)\client.obj
$(OBJ_DIR)\tran.obj $(OBJ_DIR)\sqlfuncs.obj
$(OBJ_DIR)\random.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\data.obj $(OBJ_DIR)\getargs.obj
$(OBJ_DIR)\time.obj $(OBJ_DIR)\stats.obj
$(OBJ_DIR)\strings.obj
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\client.exe                \
    $(OBJ_DIR)\client.obj $(OBJ_DIR)\tran.obj
$(OBJ_DIR)\sqlfuncs.obj   \
    $(OBJ_DIR)\random.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\data.obj       \
    $(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj      \

$(OBJ_DIR)\strings.obj
          \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)

$(EXE_DIR)\master.exe : $(OBJ_DIR)\master.obj
$(OBJ_DIR)\sqlfuncs.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj
```

```
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\master.exe              \
    $(OBJ_DIR)\master.obj $(OBJ_DIR)\sqlfuncs.obj
$(OBJ_DIR)\util.obj   \
    $(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj   \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)

$(EXE_DIR)\tpccldr.exe : $(OBJ_DIR)\tpccldr.obj
$(OBJ_DIR)\getargs.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\time.obj $(OBJ_DIR)\random.obj
$(OBJ_DIR)\strings.obj
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\tpccldr.exe              \
    $(OBJ_DIR)\tpccldr.obj $(OBJ_DIR)\getargs.obj
$(OBJ_DIR)\strings.obj \
    $(OBJ_DIR)\util.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\random.obj        \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)

$(EXE_DIR)\delivery.exe : $(OBJ_DIR)\delivery.obj
$(OBJ_DIR)\sqlfuncs.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\delivery.exe                 \
    $(OBJ_DIR)\delivery.obj $(OBJ_DIR)\sqlfuncs.obj
$(OBJ_DIR)\util.obj      \
    $(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj      \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)

$(EXE_DIR)\sqlstat.exe : $(OBJ_DIR)\sqlstat.obj
$(OBJ_DIR)\sqlfuncs.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\sqlstat.exe                \
    $(OBJ_DIR)\sqlstat.obj $(OBJ_DIR)\sqlfuncs.obj
$(OBJ_DIR)\util.obj      \
    $(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj      \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)
```

*random.c*

```
/*  FILE:RANDOM.C
 *  Microsoft TPC-C Kit Ver. 3.00.000
 *  Audited 08/23/96, By Francois Raab
 *
 *  Copyright Microsoft, 1996
 *
 *  PURPOSE:Random number generation functions for
Microsoft TPC-C Benchmark Kit
 *  Author:Damien Lindauer
 *  damienl@Microsoft.com
 */
// Includes
#include "tpcc.h"
#include "math.h"
// Defines
#define A            16807
#define M       2147483647
#define Q           127773      /* M div A */
#define R             2836      /* M mod A */
#define Thread __declspec(thread)
// Globals
long Thread Seed = 0;       /* thread local seed */
/************************************************************
*******************
*
                  *
* random
-
         *
*       Implements a GOOD pseudo random number generator.
This generator     *
*       will/should? run the complete period before
repeating.               *
*
                  *
* Copied
from:
         *
*       Random Numbers Generators: Good Ones Are Hard to
Find.                *
*       Communications of the ACM - October 1988 Volume 31
Number 10          *
*
                  *
```

```
* Machine
Dependencies:
        *
*       long must be 2 ^ 31 - 1 or
greater.                                    *
*
                  *
************************************************************
*****************/
/***********************************************************
******************
* seed - load the Seed value used in irand and drand.
Should be used before  *
*       first call to irand or
drand.                                         *
************************************************************
******************/
void seed(long val)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering seed()...\n", (int)
GetCurrentThreadId());
   printf("Old Seed %ld New Seed %ld\n",Seed, val);
#endif
    if ( val < 0 )
        val = abs(val);
    Seed = val;
}
/***********************************************************
*****************
*
                 *
* irand - returns a 32 bit integer pseudo random number
with a period of     *
*       1 to 2 ^ 32 -
1.                                                     *
*
                 *
*
parameters:
               *
*
none.
         *
```

```
*
                 *
*
returns:
               *
*       32 bit integer - defined as long ( see above
).                       *
*
                 *
* side
effects:
          *
*       seed get
recomputed.
*
************************************************************
****************/
long irand()
{
    register long    s;      /* copy of seed */
    register long    test;   /* test flag */
    register long    hi;     /* tmp value for speed */
    register long    lo;     /* tmp value for speed */
#ifdef DEBUG
    printf("[%ld]DBG: Entering irand()...\n", (int)
GetCurrentThreadId());
#endif
    s = Seed;
    hi = s / Q;
    lo = s % Q;
    test = A * lo - R * hi;
    if ( test > 0 )
   Seed = test;
    else
   Seed = test + M;
    return( Seed );
}
/***********************************************************
*****************
*
                 *
* drand - returns a double pseudo random number between 0.0
and 1.0.         *
```

```
*       See
irand.
     *
************************************************************
*****************/
double drand()
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering drand()...\n", (int)
GetCurrentThreadId());
#endif
    return( (double)irand() / 2147483647.0);
}
//============================================================
=============
// Function   : RandomNumber
//
// Description:
//============================================================
=============
long RandomNumber(long lower, long upper)
{
    long rand_num;
#ifdef DEBUG
    printf("[%ld]DBG: Entering RandomNumber()...\n", (int)
GetCurrentThreadId());
#endif
    if ( upper == lower )/* pgd 08-13-96 perf enhancement */
    return lower;
    upper++;
    if ( upper <= lower )
    rand_num = upper;
    else
    rand_num = lower + irand() % (upper - lower); /* pgd 08-
13-96 perf enhancement */
#ifdef DEBUG
    printf("[%ld]DBG: RandomNumber between %ld & %ld ==>
%ld\n",
    (int) GetCurrentThreadId(), lower, upper, rand_num);
#endif
    return rand_num;
}
#if 0
//Orginal code pgd 08/13/96
long RandomNumber(long lower,
      long upper)
{
    long rand_num;
#ifdef DEBUG
    printf("[%ld]DBG: Entering RandomNumber()...\n", (int)
GetCurrentThreadId());
#endif
    upper++;
    if ((upper <= lower))
    rand_num = upper;
    else
    rand_num = lower + irand() % ((upper > lower) ? upper -
lower : upper);
#ifdef DEBUG
    printf("[%ld]DBG: RandomNumber between %ld & %ld ==>
%ld\n",
    (int) GetCurrentThreadId(), lower, upper, rand_num);
#endif
    return rand_num;
}
#endif
//============================================================
=============
// Function   : NURand
//
// Description:
//============================================================
=============
long NURand(int iConst,
        long x,
        long y,
        long C)
{
    long rand_num;
#ifdef DEBUG
    printf("[%ld]DBG: Entering NURand()...\n", (int)
GetCurrentThreadId());
#endif

    rand_num = (((RandomNumber(0,iConst) |
RandomNumber(x,y)) + C) % (y-x+1))+x;
#ifdef DEBUG
    printf("[%ld]DBG: NURand: num = %d\n", (int)
GetCurrentThreadId(), rand_num);
#endif
    return rand_num;
}
```

*strings.c*

```
/*  FILE:STRINGS.C
 *  Microsoft TPC-C Kit Ver. 3.00.000
 *  Audited 08/23/96, By Francois Raab
 *
 *  Copyright Microsoft, 1996
 *
 *  PURPOSE:String generation functions for Microsoft TPC-C
Benchmark Kit
 *  Author:Damien Lindauer
 *  damienl@Microsoft.com
 */
// Includes
#include "tpcc.h"
#include <string.h>
#include <ctype.h>
//=========================================================
=============
//
// Function name: MakeAddress
//
//=========================================================
=============
void MakeAddress(char *street_1,
     char *street_2,
     char *city,
      char *state,
      char *zip)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeAddress()\n", (int)
GetCurrentThreadId());
#endif
    MakeAlphaString (10, 20, ADDRESS_LEN, street_1);
    MakeAlphaString (10, 20, ADDRESS_LEN, street_2);
    MakeAlphaString (10, 20, ADDRESS_LEN, city);
    MakeAlphaString ( 2,  2, STATE_LEN, state);
    MakeZipNumberString( 9,  9, ZIP_LEN, zip);
#ifdef DEBUG
    printf("[%ld]DBG: MakeAddress: street_1: %s, street_2:
%s, city: %s, state: %s, zip: %s\n",
    (int) GetCurrentThreadId(), street_1, street_2, city,
state, zip);
#endif
    return;
}
//=========================================================
=============
//
// Function name: LastName
//
//=========================================================
=============
void LastName(int  num,
         char *name)
{
    inti;
    intlen;
    static char *n[] =
    {
    "BAR" , "OUGHT", "ABLE" , "PRI"  , "PRES",
    "ESE" , "ANTI" , "CALLY", "ATION", "EING"
    };
#ifdef DEBUG
    printf("[%ld]DBG: Entering LastName()\n", (int)
GetCurrentThreadId());
#endif
    if ((num >= 0) && (num < 1000))
    {
    strcpy(name, n[(num/100)%10]);
    strcat(name, n[(num/10)%10]);
    strcat(name, n[(num/1)%10]);

    if (strlen(name) < LAST_NAME_LEN)
    {
    PaddString(LAST_NAME_LEN, name);
    }
    }
    else
    {
    printf("\nError in LastName()... num <%ld> out of range
(0,999)\n", num);
    exit(-1);
    }

#ifdef DEBUG
    printf("[%ld]DBG: LastName: num = [%d] ==>
[%d][%d][%d]\n",
    (int) GetCurrentThreadId(), num, num/100, (num/10)%10,
num%10);
    printf("[%ld]DBG: LastName: String = %s\n", (int)
GetCurrentThreadId(), name);
#endif
    return;
}
//=========================================================
============
//
// Function name: MakeAlphaString
//
//=========================================================
============
//philipdu 08/13/96 Changed MakeAlphaString to use A-Z, a-
z, and 0-9 in
//accordance with spec see below:
//The spec says:
//4.3.2.2The notation random a-string [x .. y]
//(respectively, n-string [x .. y]) represents a string of
random alphanumeric
//(respectively, numeric) characters of a random length of
minimum x, maximum y,
//and mean (y+x)/2. Alphanumerics are A..Z, a..z, and
0..9.  The only other
//requirement is that the character set used "must be able
to represent a minimum
//of 128 different characters".  We are using 8-bit chars,
so this is a non issue.
//It is completely unreasonable to stuff non-printing chars
into the text fields.
//-CLevine 08/13/96
```

```
int MakeAlphaString( int  x, int  y, int  z, char *str)
{
    intlen;
    inti;
    staticchar chArray[] =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvw
xyz";
    staticintchArrayMax = 61;
#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeAlphaString()\n", (int)
GetCurrentThreadId());
#endif
    len= RandomNumber(x, y);
    for (i=0; i<len; i++)
    str[i] = chArray[RandomNumber(0, chArrayMax)];
    if ( len < z )
    memset(str+len, ' ', z - len);
    str[len] = 0;

    return len;
}
#if 0
//philipdu 08/13/96 Orginal MakeAlphaString
int MakeAlphaString( int  x,
        int  y,
    int  z,
        char *str)
{

    intlen;
    inti;
#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeAlphaString()\n", (int)
GetCurrentThreadId());
#endif
    len= RandomNumber(x, y);
    for (i=0; i<len; i++)
    {
    str[i] = RandomNumber(MINPRINTASCII, MAXPRINTASCII);
    }
    str[len] = '\0';
    if (len < z)
    {
    PaddString(z, str);
    }
    return (len);
}
#endif
//=========================================================
=============
//
// Function name: MakeOriginalAlphaString
//
//=========================================================
=============
int MakeOriginalAlphaString(int  x,
        int  y,
    int  z,
        char *str,
        int  percent)
{
    intlen;
    intval;
    intstart;
#ifdef DEBUG
    printf("[%ld]DBG: Entering
MakeOriginalAlphaString()\n", (int) GetCurrentThreadId());
#endif
    // verify prercentage is valid
    if ((percent < 0) || (percent > 100))
    {
    printf("MakeOrigianlAlphaString: Invalid percentage:
%d\n", percent);
    exit(-1);
    }
    // verify string is at least 8 chars in length
    if ((x + y) <= 8)
    {
    printf("MakeOriginalAlphaString: string length must be
>= 8\n");
    exit(-1);
    }
    // Make Alpha String
    len = MakeAlphaString(x,y, z, str);
    val = RandomNumber(1,100);
    if (val <= percent)
    {
    start = RandomNumber(0, len - 8);
    strncpy(str + start, "ORIGINAL", 8);
    }

#ifdef DEBUG
    printf("[%ld]DBG: MakeOriginalAlphaString: : %s\n",
    (int) GetCurrentThreadId(), str);
#endif
    return strlen(str);
}
//=========================================================
=============
//
// Function name: MakeNumberString
//
//=========================================================
=============
int MakeNumberString(int  x, int  y, int  z, char *str)
{
    char tmp[16];
    //MakeNumberString is always called
MakeZipNumberString(16, 16, 16, string)
    memset(str, '0', 16);
    itoa(RandomNumber(0, 99999999), tmp, 10);
    memcpy(str, tmp, strlen(tmp));
    itoa(RandomNumber(0, 99999999), tmp, 10);
    memcpy(str+8, tmp, strlen(tmp));
    str[16] = 0;
    return 16;
}
#if 0
int MakeNumberString(int  x,
        int  y,
    int  z,
        char *str)
{
    intlen;
    inti;
```

```
#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeNumberString()\n", (int)
GetCurrentThreadId());
#endif
    len = RandomNumber(x,y);
    for (i=0; i < len; i++)
    {
   str[i] = (char) (RandomNumber(48,57));
    }

    str[len] = '\0';
   PaddString(z, str);
    return strlen(str);
}
#endif
//============================================================
============
//
// Function name: MakeZipNumberString
//
//============================================================
============
int MakeZipNumberString(int  x, int  y, int  z, char *str)
{
   char tmp[16];
   //MakeZipNumberString is always called
MakeZipNumberString(9, 9, 9, string)
   strcpy(str, "000011111");
   itoa(RandomNumber(0, 9999), tmp, 10);
   memcpy(str, tmp, strlen(tmp));
    return 9;
}
#if 0
//pgd 08/14/96 Orginal Code Below
int MakeZipNumberString(int  x,
        int  y,
     int  z,
         char *str)
{
    intlen;
    inti;
#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeZipNumberString()\n",
(int) GetCurrentThreadId());
#endif
    len = RandomNumber(x-5,y-5);
    for (i=0; i < len; i++)
    {
   str[i] = (char) (RandomNumber(48,57));
    }

    str[len] = '\0';
   strcat(str, "11111");
   PaddString(z, str);
    return strlen(str);
}
#endif
//============================================================
============
//
// Function name: InitString
//
//============================================================
============
void InitString(char *str, int len)
{
    int i;
#ifdef DEBUG
    printf("[%ld]DBG: Entering InitString()\n", (int)
GetCurrentThreadId());
#endif
   memset(str, ' ', len);
    str[len] = 0;
}
#if 0
//Orginal pgd 08/14/96
void InitString(char *str, int len)
{
    int i;
#ifdef DEBUG
    printf("[%ld]DBG: Entering InitString()\n", (int)
GetCurrentThreadId());
#endif
    for (i=0; i< len; i++)
    str[i] = ' ';
    str[len] = '\0';
}
#endif
//=========================================================
=============
// Function name: InitAddress
//
// Description:
//
//=========================================================
=============
void InitAddress(char *street_1, char *street_2, char
*city, char *state, char *zip)
{
    int i;
    memset(street_1, ' ', ADDRESS_LEN+1);
    memset(street_2, ' ', ADDRESS_LEN+1);
    memset(city, ' ', ADDRESS_LEN+1);
    street_1[ADDRESS_LEN+1] = 0;
    street_2[ADDRESS_LEN+1] = 0;
    city[ADDRESS_LEN+1]     = 0;
    memset(state, ' ', STATE_LEN+1);
    state[STATE_LEN+1] = 0;
    memset(zip, ' ', ZIP_LEN+1);
    zip[ZIP_LEN+1] = 0;
}
#if 0
//Orginal pgd 08/14/96
void InitAddress(char *street_1,
     char *street_2,
      char *city,
      char *state,
      char *zip)
{
    int i;
#ifdef DEBUG
    printf("[%ld]DBG: Entering InitAddress()\n", (int)
GetCurrentThreadId());
#endif
    for (i=0; i< ADDRESS_LEN+1; i++)
    {
    street_1[i] = ' ';
    street_2[i] = ' ';
    city[i]     = ' ';
    }
    street_1[ADDRESS_LEN+1] = '\0';
    street_2[ADDRESS_LEN+1] = '\0';
    city[ADDRESS_LEN+1]     = '\0';
    for (i=0; i< STATE_LEN+1; i++)
    state[i] = ' ';
    state[STATE_LEN+1] = '\0';
    for (i=0; i< ZIP_LEN+1; i++)
    zip[i] = ' ';
    zip[ZIP_LEN+1] = '\0';
}
#endif
//=========================================================
=============
//
// Function name: PaddString
//
//=========================================================
=============
void PaddString(int max, char *name)
{
    inti;
    intlen;
    len = strlen(name);
    if ( len < max )
    memset(name+len, ' ', max - len);
    name[max] = 0;
    return;
}
#if 0
    //pgd 08/14/96 Orginal code below
    void PaddString(intmax,
    char*name)
    {
    inti;
    intlen;
    #ifdef DEBUG
    printf("[%ld]DBG: Entering PaddString()\n", (int)
GetCurrentThreadId());
```

```
    #endif
    len = strlen(name);
    for (i=1;i<=(max - len);i++)
    {
    strcat(name, " ");
    }
    }
#endif
```

*time.c*

```
// TPC-C Benchmark Kit
//
// Module:  TIME.C
// Author:  DamienL
// Includes
#include "tpcc.h"
// Globals
static long start_sec;
//=========================================================
============
//
// Function name: TimeNow
//
//=========================================================
============
long TimeNow()
{
   longtime_now;
   struct_timeb el_time;
#ifdef DEBUG
    printf("[%ld]DBG: Entering TimeNow()\n", (int)
GetCurrentThreadId());
#endif
   _ftime(&el_time);
   time_now = ((el_time.time - start_sec) * 1000) +
el_time.millitm;
   return time_now;
}
//=========================================================
============
//
// Function name: TimeInit
//
// This function is used to normalize the seconds component
of
// elapsed time so that it will not overflow, when
converted to milli seconds
//
//=========================================================
============
void TimeInit()
{
    struct  _timeb  norm_time;
#ifdef DEBUG
    printf("[%ld]DBG: Entering TimeInit()\n", (int)
GetCurrentThreadId());
#endif
    _ftime(&norm_time);
    start_sec = norm_time.time;
}
//=========================================================
============
//
// Function name: TimeKeying
//
//=========================================================
============
void TimeKeying(intTranType,
   doubleload_multiplier)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering TimeKeying()\n", (int)
GetCurrentThreadId());
#endif
   switch (TranType)
   {
       case NEW_ORDER_TRAN:
   UtilSleepMs( (long) ((load_multiplier * 18)*1000) );
   break;
       case PAYMENT_TRAN:
   UtilSleepMs( (long) ((load_multiplier * 3)*1000) );
   break;
       case ORDER_STATUS_TRAN:
       case DELIVERY_TRAN:
       case STOCK_LEVEL_TRAN:
```

```
    UtilSleepMs( (long) ((load_multiplier * 2)*1000) );
    break;
        default:
    printf("TimeKeying: Error - default reached!\n");
    }
}
//=========================================================
============
//
// Function name: TimeThink
//
//=========================================================
============
void TimeThink(intTranType,
       doubleload_multiplier)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering TimeThink()\n", (int)
GetCurrentThreadId());
#endif
    switch (TranType)
    {
        case NEW_ORDER_TRAN:
        case PAYMENT_TRAN:
    UtilSleepMs( (long) ((load_multiplier * 12)*1000) );
    break;
        case ORDER_STATUS_TRAN:
    UtilSleepMs( (long) ((load_multiplier * 10)*1000) );
    break;
        case DELIVERY_TRAN:
        case STOCK_LEVEL_TRAN:
    UtilSleepMs( (long) ((load_multiplier * 5)*1000) );
    break;
        default:
    printf("TimeThink: Error - default reached!\n");
    }
}
```

*tpcc.h*

```
/*  FILE:TPCC.H
 *  Microsoft TPC-C Kit Ver. 3.00.000
 *  Audited 08/23/96, By Francois Raab
 *
 *  Copyright Microsoft, 1996
 *
 *  PURPOSE:Header file for Microsoft TPC-C Benchmark Kit
 *  Author:Damien Lindauer
 *  damienl@Microsoft.com
 */
// Build number of TPC Benchmark Kit
#define TPCKIT_VER "3.00.00"
// General headers
#include <windows.h>
#include <winbase.h>
#include <stdlib.h>
#include <stdio.h>
#include <process.h>
#include <stddef.h>
#include <stdarg.h>
#include <string.h>
#include <signal.h>
#include <time.h>
#include <timeb.h>
#include <types.h>
#include <wincon.h>
#ifdef USE_ODBC
// ODBC headers
#include <sql.h>
#include <sqlext.h>
HENV henv;
#endif
// DB-Library headers
#include <sqlfront.h>
#include <sqldb.h>
#include "trans.h"//pgd 5-6-96 split transaction structs
definations into own header
    //for tpcform.c i.e. telnet application
// Critical section declarations
CRITICAL_SECTIONConsoleCritSec;
CRITICAL_SECTIONQueuedDeliveryCritSec;
CRITICAL_SECTIONWriteDeliveryCritSec;
```

```
CRITICAL_SECTIONDroppedConnectionsCritSec;
CRITICAL_SECTIONClientErrorLogCritSec;
// General constants
#define SQLCONN DBPROCESS
#define DUMB_MESSAGE     5701
#define ABORT_ERROR      6104
#define INVALID_ITEM_ID 0
#define MILLI            1000
#define MAX_THREADS      2510
#define STATS_MSG_LOW   3600
#define STATS_MSG_HIGH  3700
#define SHOWPLAN_MSG_LOW   6200
#define SHOWPLAN_MSG_HIGH  6300
#define FALSE           0
#define TRUE            1
#define UNDEF   -1
#define MINPRINTASCII   32
#define MAXPRINTASCII  126
// Default environment constants
#define SERVER          ""
#define DATABASE        "tpcc"
#define USER            "sa"
#define PASSWORD            ""
#define SYNCH_SERVERNAME""
// Statistic constants
#define INTERVAL          20   // Total interval of buckets, in
sec
#define UNIT               .1    // Time period of each bucket
#define HIST_MAX          200    // Num of histogram buckets =
INTERVAL/UNIT
#define BUCKET           100 //Division factor for response time
// Default master arguments
#define ADMIN_DATABASE        "tpcc_admin"
#define RAMP_UP               600
#define STEADY_STATE         1200
#define RAMP_DOWN             120
#define NUM_USERS              10
#define NUM_WAREHOUSES      1
#define THINK_TIMES0
#define DISPLAY_DATA        0
#define DEFMSPACKSIZE     4096
#define TRANSACTION        0
#define CLIENT_MODE          1
#define DEF_WW_T  120
#define DEF_WW_a1
#define DEADLOCK_RETRY         4
#define DELIVERY_BACKOFF2
#define DELIVERY_MODE0
#define NEWORDER_MODE0
#define DEF_LOAD_MULTIPLIER   1.0
#define DEF_CHECKPOINT_INTERVAL   960
#define DEF_FIRST_CHECKPOINT  240
#define DISABLE_90TH0
#defineRESFILENAME"results.txt"
#define SQLSTAT_FILENAME"sqlstats.txt"
#define ENABLE_SQLSTAT0
#define SQLSTAT_PERIOD  100
#define SHUTDOWN_SERVER0
#define AUTO_RUN0
#define DISABLE_SQLPERF0
// Default client arguments
#define NUM_THREADS       10
#define X_FLAG                      0
#define Y_FLAG                      1
#define NUM_DELIVERIES2
#define CLIENT_NURAND      223
#define DISABLE_DELIVERY_RESFILES   1
#define ENABLE_QJ0
// Globals for queued delivery handling
typedef struct delivery_node *DELIVERY_PTR;
DELIVERY_PTRdelivery_head, delivery_tail;
shortqueued_delivery_cnt;
HANDLEhDeliveryMonPipe;
struct delivery_node
{
    shortw_id;
    shorto_carrier_id;
    SYSTEMTIMEqueue_time;
    longtran_start_time;
    structdelivery_node *next_delivery;
};
// Default loader arguments
#define BATCH           10000
```

```
#define DEFLDPACKSIZE 4096
#define ORDERS_PER_DIST  3000
#define LOADER_RES_FILE     "load.out"
#define LOADER_NURAND_C  123
#define DEF_STARTING_WAREHOUSE1
#define BUILD_INDEX1
#define INDEX_SCRIPT_PATH"scripts"
// Transaction types
#define EMPTY               0
#define NEW_ORDER_TRAN      1
#define PAYMENT_TRAN        2
#define ORDER_STATUS_TRAN   3
#define DELIVERY_TRAN       4
#define STOCK_LEVEL_TRAN    5
// Statistic structures
typedef struct
{
    long   tran_count;
    long   total_time;
    long   resp_time;
    long   resp_min;
    long   resp_max;
    long   rolled_back;
    long   tran_2sec;
    long   tran_5sec;
    long   tran_sqr;
    long   num_deadlocks;
    long   resp_hist[HIST_MAX];
} TRAN_STATS;
typedef struct
{
    TRAN_STATS NewOrderStats;
    TRAN_STATS PaymentStats;
    TRAN_STATS OrderStatusStats;
    TRAN_STATS QueuedDeliveryStats;
    TRAN_STATSDeliveryStats;
    TRAN_STATS StockLevelStats;
} CLIENT_STATS;
// driver structures
typedef struct
{
    char  *server;
    char  *database;
    char  *user;
    char  *password;
    char  *table;
    long   num_warehouses;
    long   batch;
    long   verbose;
    long   pack_size;
    char*loader_res_file;
    char*synch_servername;
    longcase_sensitivity;
    longstarting_warehouse;
    longbuild_index;
    char*index_script_path;
} TPCCLDR_ARGS;
typedef struct
{
    char  *server;
    char  *user;
    char  *password;
    char  *admin_database;
    char  *sqlstat_filename;
    longrun_id;
} SQLSTAT_ARGS;
typedef struct
{
    SQLCONN  *sqlconn;
    char    *server;
    char     *database;
    char*admin_database;
    char     *user;
    char     *password;
    long      ramp_up;
    long      steady_state;
    long      ramp_down;
    long      num_users;
    long      num_warehouses;
    long      think_times;
    long      display_data;
    long     client_mode;
    long     tran;
    longdeadlock_retry;
    longdelivery_backoff;
    longnum_deliveries;
    char *comment;
    doubleload_multiplier;
    longcheckpoint_interval;
    longfirst_checkpoint;
    longdisable_90th;
    char*resfilename;
    char*sqlstat_filename;
    longenable_sqlstat;
    longsqlstat_period;
    longshutdown_server;
    longauto_run;
    longdropped_connections;
    short     spid;
    longdisable_sqlperf;
} MASTER_DATA;
typedef struct
{
    long  num_threads;
    char      *server;
    char      *database;
    char*admin_database;
    char      *user;
    char      *password;
    long      pack_size;
    shortx_flag;
    char*synch_servername;
    longdisable_delivery_resfiles;
    longenable_qj;
#ifdef USE_CONMON
    HANDLE  hConMon;
    short  con_id;
    short  con_x;
    short  con_y;
#endif
} GLOBAL_CLIENT_DATA;
typedef struct
{
#ifdef USE_ODBC
    HDBChdbc;
    HSTMThstmt;
#else
    SQLCONN   *sqlconn;
#endif
    short  threadid;
    char      *server;
    char      *database;
    char*admin_database;
    char      *user;
    char      *password;
    long      ramp_up;
    long      steady_state;
    long      ramp_down;
    long     num_warehouses;
    long     client_mode;
    long     tran;
    longdeadlock_retry;
    long      think_times;
    long      pack_size;
    long      tran_start_time;
    long      tran_end_time;
    long      display_data;
    long     id;
    short     w_id;
    short     spid;
    longdisable_90th;
    doubleload_multiplier;
    longnum_deliveries;
    longenable_qj;
#ifdef USE_CONMON
    HANDLE  hConMon;
    short  con_id;
    short  con_x;
    short  con_y;
    shortfTimerStat;
#endif
} CLIENT_DATA;
typedef struct
{
```

```
#ifdef USE_ODBC
   HDBChdbc;
   HSTMThstmt;
#else
   SQLCONN   *sqlconn;
#endif
   SYSTEMTIMEqueue_time;
   SYSTEMTIMEcompletion_time;
   long      tran_start_time;
   long      tran_end_time;
   short  threadid;
   FILE      *fDelivery;
   short     spid;
   short     w_id;
   shortd_id;
   short     o_carrier_id;
   DEL_ITEM  DelItems[10];
   char      *server;
   char      *database;
   char*admin_database;
   char      *user;
   char      *password;
   long      ramp_up;
   long      steady_state;
   long      ramp_down;
   long      pack_size;
   long     id;
   longdisable_90th;
   longdelivery_backoff;
   longdisable_delivery_resfiles;
   longenable_qj;
} DELIVERY;
typedef struct
{
   longpipe_num;
} DELIVERY_ARGS;
// For client synchronization
#define LINE_LEN   80
#define NAME_SIZE  25
#define IN_BUF_SIZE        1000
#define OUT_BUF_SIZE       1000
#define TIME_OUT           0
#define PLEASE_READ        1000
#define PLEASE_WRITE       1000
typedef struct _WRTHANDLE
{   HANDLEhPipe;
    DWORDthreadID;
    CHARName[NAME_SIZE];
    struct _WRTHANDLE *next;
}WRTHANDLE;
// For client console monitor
#ifdef USE_CONMON
#defineCON_LINE_SIZE40
#defineDEADLOCK_X17
#define DEADLOCK_Y4
#define CUR_STATE_X15
#define CUR_STATE_Y3
#defineYELLOW0
#defineRED1
#defineGREEN2
int total_deadlocks;
#endif
// Functions in random.c
void   seed();
long   irand();
doubledrand();
voidWUCreate();
shortWURand();
// Functions in getargs.c;
void GetArgsLoader();
void GetArgsLoaderUsage();
void GetArgsMaster();
void GetArgsMasterUsage();
void GetArgsClient();
void GetArgsClientUsage();
void GetArgsDelivery();
void GetArgsDeliveryUsage();
void GetArgsSQLStat();
void GetArgsSQLStatUsage();
// Functions in master.c
void ReadClientDone();
BOOL    CtrlHandler();
```

```
// Functions in client.c
void  ClientMain();
voidDeliveryMain();
voidDelivery();
void  ClientEmulate();
short ClientSelectTransaction();
void  ClientShuffleDeck();
//Functions in tran.c
BOOL TranNewOrder();
BOOL TranPayment();
BOOL TranOrderStatus();
BOOL TranDelivery();
BOOL TranStockLevel();
// Functions in data.c
void DataNewOrder();
void DataPayment();
void DataOrderStatus();
void DataDelivery();
void DataStockLevel();
short DataRemoteWarehouse();
// Functions in time.c
long TimeNow();
void TimeInit();
void TimeKeying();
void TimeThink();
// Functions in stats.c
void StatsInit();
void StatsInitTran();
void StatsGeneral();
void StatsDelivery();
// Functions in sqlfuncs.c
BOOL  SQLExec();
BOOL  SQLExecCmd();
BOOL  SQLOpenConnection();
void  SQLClientInit();
int   SQLMasterInit();
voidSQLDeliveryInit();
int   SQLClientStats();
int   SQLDeliveryStats();
void  SQLTranStats();
void  SQLMasterStats();
void  SQLMasterTranStats();
void  SQLIOStats();
void  SQLCheckpointStats();
voidSQLInitResFile();
void  SQLGetRunId();
BOOL  SQLNewOrder();
BOOL  SQLPayment();
BOOL  SQLOrderStatus();
BOOLSQLStockLevel();
void    SQLDelivery();
int   SQLGetCustId();
void   SQLExit();
void  SQLInit();
void  SQLInitPrivate();
void  SQLClientInitPrivate();
void  SQLDeliveryInitPrivate();
int   SQLMsgHandler();
int   SQLErrHandler();
int   SQLClientMsgHandler();
int   SQLClientErrHandler();
int   SQLDeliveryMsgHandler();
int   SQLDeliveryErrHandler();
void  SQLInitDate();
voidSQLShutdown();
#ifdef USE_ODBC
void ODBCOpenConnection();
void ODBCOpenDeliveryConnection();
BOOLODBCError();
voidODBCExit();
#endif
// Functions in util.c
void UtilSleep();
void UtilPrintNewOrder();
void UtilPrintPayment();
void UtilPrintOrderStatus();
void UtilPrintDelivery();
void UtilPrintStockLevel();
void UtilPrintOlTable();
void UtilError();
void UtilFatalError();
void UtilStrCpy();
```

```
#ifdef USE_CONMON
void WriteConsoleString();
#endif
voidWriteDeliveryString();
BOOLAddDeliveryQueueNode();
BOOLGetDeliveryQueueNode();
// Functions in strings.c
void MakeAddress();
void LastName();
int  MakeAlphaString();
int  MakeOriginalAlphaString();
int  MakeNumberString();
int  MakeZipNumberString();
void InitString();
void InitAddress();
void PaddString();
// Functions in delivery.c
void  DeliveryHMain();
void DeliveryH();
```

*tpccldr.c*

```
/*  FILE:TPCC.H
 *  Microsoft TPC-C Kit Ver. 3.00.000
 *  Audited 08/23/96, By Francois Raab
 *
 *  Copyright Microsoft, 1996
 *
 *  PURPOSE:Header file for Microsoft TPC-C Benchmark Kit
 *  Author:Damien Lindauer
 *  damienl@Microsoft.com
 */
// Build number of TPC Benchmark Kit
#define TPCKIT_VER "3.00.00"
// General headers
#include <windows.h>
#include <winbase.h>
#include <stdlib.h>
#include <stdio.h>
#include <process.h>
#include <stddef.h>
#include <stdarg.h>
#include <string.h>
#include <signal.h>
#include <time.h>
#include <timeb.h>
#include <types.h>
#include <wincon.h>
#ifdef USE_ODBC
// ODBC headers
#include <sql.h>
#include <sqlext.h>
HENV henv;
#endif
// DB-Library headers
#include <sqlfront.h>
#include <sqldb.h>
#include "trans.h"//pgd 5-6-96 split transaction structs
defination into own header
    //for tpcform.c i.e. telnet application
// Critical section declarations
CRITICAL_SECTIONConsoleCritSec;
CRITICAL_SECTIONQueuedDeliveryCritSec;
CRITICAL_SECTIONWriteDeliveryCritSec;
CRITICAL_SECTIONDroppedConnectionsCritSec;
CRITICAL_SECTIONClientErrorLogCritSec;
// General constants
#define SQLCONN DBPROCESS
#define DUMB_MESSAGE     5701
#define ABORT_ERROR      6104
#define INVALID_ITEM_ID 0
#define MILLI            1000
#define MAX_THREADS      2510
#define STATS_MSG_LOW   3600
#define STATS_MSG_HIGH  3700
#define SHOWPLAN_MSG_LOW   6200
#define SHOWPLAN_MSG_HIGH  6300
#define FALSE           0
#define TRUE            1
#define UNDEF   -1
#define MINPRINTASCII   32
#define MAXPRINTASCII  126
```

```
// Default environment constants
#define SERVER          ""
#define DATABASE        "tpcc"
#define USER            "sa"
#define PASSWORD            ""
#define SYNCH_SERVERNAME""
// Statistic constants
#define INTERVAL          20  // Total interval of buckets, in
sec
#define UNIT               .1   // Time period of each bucket
#define HIST_MAX          200   // Num of histogram buckets =
INTERVAL/UNIT
#define BUCKET           100 //Division factor for response time
// Default master arguments
#define ADMIN_DATABASE        "tpcc_admin"
#define RAMP_UP               600
#define STEADY_STATE         1200
#define RAMP_DOWN             120
#define NUM_USERS              10
#define NUM_WAREHOUSES      1
#define THINK_TIMES0
#define DISPLAY_DATA        0
#define DEFMSPACKSIZE     4096
#define TRANSACTION        0
#define CLIENT_MODE          1
#define DEF_WW_T  120
#define DEF_WW_a1
#define DEADLOCK_RETRY         4
#define DELIVERY_BACKOFF2
#define DELIVERY_MODE0
#define NEWORDER_MODE0
#define DEF_LOAD_MULTIPLIER   1.0
#define DEF_CHECKPOINT_INTERVAL   960
#define DEF_FIRST_CHECKPOINT  240
#define DISABLE_90TH0
#defineRESFILENAME"results.txt"
#define SQLSTAT_FILENAME"sqlstats.txt"
#define ENABLE_SQLSTAT0
#define SQLSTAT_PERIOD  100
#define SHUTDOWN_SERVER0
#define AUTO_RUN0
#define DISABLE_SQLPERF0
// Default client arguments
#define NUM_THREADS       10
#define X_FLAG                       0
#define Y_FLAG                       1
#define NUM_DELIVERIES2
#define CLIENT_NURAND      223
#define DISABLE_DELIVERY_RESFILES   1
#define ENABLE_QJ0
// Globals for queued delivery handling
typedef struct delivery_node *DELIVERY_PTR;
DELIVERY_PTRdelivery_head, delivery_tail;
shortqueued_delivery_cnt;
HANDLEhDeliveryMonPipe;
struct delivery_node
{
   shortw_id;
   shorto_carrier_id;
   SYSTEMTIMEqueue_time;
   longtran_start_time;
   structdelivery_node *next_delivery;
};
// Default loader arguments
#define BATCH           10000
#define DEFLDPACKSIZE 4096
#define ORDERS_PER_DIST  3000
#define LOADER_RES_FILE     "load.out"
#define LOADER_NURAND_C  123
#define DEF_STARTING_WAREHOUSE1
#define BUILD_INDEX1
#define INDEX_SCRIPT_PATH"scripts"
// Transaction types
#define EMPTY               0
#define NEW_ORDER_TRAN      1
#define PAYMENT_TRAN        2
#define ORDER_STATUS_TRAN   3
#define DELIVERY_TRAN       4
#define STOCK_LEVEL_TRAN    5
// Statistic structures
typedef struct
{
    long   tran_count;
    long   total_time;
    long   resp_time;
    long   resp_min;
    long   resp_max;
    long   rolled_back;
    long   tran_2sec;
    long   tran_5sec;
    long   tran_sqr;
    long   num_deadlocks;
    long   resp_hist[HIST_MAX];
} TRAN_STATS;
typedef struct
{
    TRAN_STATS NewOrderStats;
    TRAN_STATS PaymentStats;
    TRAN_STATS OrderStatusStats;
    TRAN_STATS QueuedDeliveryStats;
    TRAN_STATSDeliveryStats;
    TRAN_STATS StockLevelStats;
} CLIENT_STATS;
// driver structures
typedef struct
{
    char  *server;
    char  *database;
    char  *user;
    char  *password;
    char  *table;
    long   num_warehouses;
    long   batch;
    long   verbose;
    long   pack_size;
    char*loader_res_file;
    char*synch_servername;
    longcase_sensitivity;
    longstarting_warehouse;
    longbuild_index;
    char*index_script_path;
} TPCCLDR_ARGS;
typedef struct
{
    char  *server;
    char  *user;
    char  *password;
    char  *admin_database;
    char  *sqlstat_filename;
    longrun_id;
} SQLSTAT_ARGS;
typedef struct
{
    SQLCONN  *sqlconn;
    char    *server;
    char     *database;
    char*admin_database;
    char     *user;
    char     *password;
    long      ramp_up;
    long      steady_state;
    long      ramp_down;
    long      num_users;
    long      num_warehouses;
    long      think_times;
    long      display_data;
    long     client_mode;
    long     tran;
    longdeadlock_retry;
    longdelivery_backoff;
    longnum_deliveries;
    char *comment;
    doubleload_multiplier;
    longcheckpoint_interval;
    longfirst_checkpoint;
    longdisable_90th;
    char*resfilename;
    char*sqlstat_filename;
    longenable_sqlstat;
    longsqlstat_period;
    longshutdown_server;
    longauto_run;
    longdropped_connections;
    short     spid;
    longdisable_sqlperf;
```

```
} MASTER_DATA;
typedef struct
{
    long  num_threads;
    char      *server;
    char      *database;
    char*admin_database;
    char      *user;
    char      *password;
    long      pack_size;
    shortx_flag;
    char*synch_servername;
    longdisable_delivery_resfiles;
    longenable_qj;
#ifdef USE_CONMON
    HANDLE  hConMon;
    short  con_id;
    short  con_x;
    short  con_y;
#endif
} GLOBAL_CLIENT_DATA;
typedef struct
{
#ifdef USE_ODBC
    HDBChdbc;
    HSTMThstmt;
#else
    SQLCONN   *sqlconn;
#endif
    short  threadid;
    char      *server;
    char      *database;
    char*admin_database;
    char      *user;
    char      *password;
    long      ramp_up;
    long      steady_state;
    long      ramp_down;
    long     num_warehouses;
    long     client_mode;
    long     tran;
    longdeadlock_retry;
    long      think_times;
    long      pack_size;
    long      tran_start_time;
    long      tran_end_time;
    long      display_data;
    long     id;
    short     w_id;
    short     spid;
    longdisable_90th;
    doubleload_multiplier;
    longnum_deliveries;
    longenable_qj;
#ifdef USE_CONMON
    HANDLE  hConMon;
    short  con_id;
    short  con_x;
    short  con_y;
    shortfTimerStat;
#endif
} CLIENT_DATA;
typedef struct
{
#ifdef USE_ODBC
    HDBChdbc;
    HSTMThstmt;
#else
    SQLCONN   *sqlconn;
#endif
    SYSTEMTIMEqueue_time;
    SYSTEMTIMEcompletion_time;
    long      tran_start_time;
    long      tran_end_time;
    short  threadid;
    FILE      *fDelivery;
    short     spid;
    short     w_id;
    shortd_id;
    short     o_carrier_id;
    DEL_ITEM  DelItems[10];
    char      *server;
    char      *database;
    char*admin_database;
    char      *user;
    char      *password;
    long      ramp_up;
    long      steady_state;
    long      ramp_down;
    long      pack_size;
    long     id;
    longdisable_90th;
    longdelivery_backoff;
    longdisable_delivery_resfiles;
    longenable_qj;
} DELIVERY;
typedef struct
{
    longpipe_num;
} DELIVERY_ARGS;
// For client synchronization
#define LINE_LEN   80
#define NAME_SIZE  25
#define IN_BUF_SIZE        1000
#define OUT_BUF_SIZE       1000
#define TIME_OUT           0
#define PLEASE_READ        1000
#define PLEASE_WRITE       1000
typedef struct _WRTHANDLE
{   HANDLEhPipe;
    DWORDthreadID;
    CHARName[NAME_SIZE];
    struct _WRTHANDLE *next;
}WRTHANDLE;
// For client console monitor
#ifdef USE_CONMON
#defineCON_LINE_SIZE40
#defineDEADLOCK_X17
#define DEADLOCK_Y4
#define CUR_STATE_X15
#define CUR_STATE_Y3
#defineYELLOW0
#defineRED1
#defineGREEN2
int total_deadlocks;
#endif
// Functions in random.c
void   seed();
long   irand();
doubledrand();
voidWUCreate();
shortWURand();
// Functions in getargs.c;
void GetArgsLoader();
void GetArgsLoaderUsage();
void GetArgsMaster();
void GetArgsMasterUsage();
void GetArgsClient();
void GetArgsClientUsage();
void GetArgsDelivery();
void GetArgsDeliveryUsage();
void GetArgsSQLStat();
void GetArgsSQLStatUsage();
// Functions in master.c
void ReadClientDone();
BOOL    CtrlHandler();
// Functions in client.c
void  ClientMain();
voidDeliveryMain();
voidDelivery();
void  ClientEmulate();
short ClientSelectTransaction();
void  ClientShuffleDeck();
//Functions in tran.c
BOOL TranNewOrder();
BOOL TranPayment();
BOOL TranOrderStatus();
BOOL TranDelivery();
BOOL TranStockLevel();
// Functions in data.c
void DataNewOrder();
void DataPayment();
void DataOrderStatus();
void DataDelivery();
void DataStockLevel();
short DataRemoteWarehouse();
// Functions in time.c
long TimeNow();
void TimeInit();
void TimeKeying();
void TimeThink();
// Functions in stats.c
void StatsInit();
void StatsInitTran();
void StatsGeneral();
void StatsDelivery();
// Functions in sqlfuncs.c
BOOL  SQLExec();
BOOL  SQLExecCmd();
BOOL  SQLOpenConnection();
void  SQLClientInit();
int   SQLMasterInit();
voidSQLDeliveryInit();
int   SQLClientStats();
int   SQLDeliveryStats();
void  SQLTranStats();
void  SQLMasterStats();
void  SQLMasterTranStats();
void  SQLIOStats();
void  SQLCheckpointStats();
voidSQLInitResFile();
void  SQLGetRunId();
BOOL  SQLNewOrder();
BOOL  SQLPayment();
BOOL  SQLOrderStatus();
BOOLSQLStockLevel();
void    SQLDelivery();
int   SQLGetCustId();
void   SQLExit();
void  SQLInit();
void  SQLInitPrivate();
void  SQLClientInitPrivate();
void  SQLDeliveryInitPrivate();
int   SQLMsgHandler();
int   SQLErrHandler();
int   SQLClientMsgHandler();
int   SQLClientErrHandler();
int   SQLDeliveryMsgHandler();
int   SQLDeliveryErrHandler();
void  SQLInitDate();
voidSQLShutdown();
#ifdef USE_ODBC
void ODBCOpenConnection();
void ODBCOpenDeliveryConnection();
BOOLODBCError();
voidODBCExit();
#endif
// Functions in util.c
void UtilSleep();
void UtilPrintNewOrder();
void UtilPrintPayment();
void UtilPrintOrderStatus();
void UtilPrintDelivery();
void UtilPrintStockLevel();
void UtilPrintOlTable();
void UtilError();
void UtilFatalError();
void UtilStrCpy();
#ifdef USE_CONMON
void WriteConsoleString();
#endif
voidWriteDeliveryString();
BOOLAddDeliveryQueueNode();
BOOLGetDeliveryQueueNode();
// Functions in strings.c
void MakeAddress();
void LastName();
int  MakeAlphaString();
int  MakeOriginalAlphaString();
int  MakeNumberString();
int  MakeZipNumberString();
void InitString();
void InitAddress();
void PaddString();
// Functions in delivery.c
void  DeliveryHMain();
```

```
    void DeliveryH();
```

*util.c*

```
    // TPC-C Benchmark Kit
    //
    // Module:  UTIL.C
    // Author:  DamienL
    // Includes
    #include "tpcc.h"
    //=========================================================
    =============
    //
    // Function name: UtilSleep
    //
    //=========================================================
    =============
    void UtilSleep(long  delay)
    {
    #ifdef DEBUG
        printf("[%ld]DBG: Entering UtilSleep()\n", (int)
    GetCurrentThreadId());
    #endif
    #ifdef DEBUG
        printf("[%ld]DBG: Sleeping for %ld seconds...\n", (int)
    GetCurrentThreadId(), delay);
    #endif
        Sleep(delay * 1000);
    }
    //=========================================================
    =============
    //
    // Function name: UtilSleep
    //
    //=========================================================
    =============
    void UtilSleepMs(long  delay)
    {
    #ifdef DEBUG
        printf("[%ld]DBG: Entering UtilSleepMs()\n", (int)
    GetCurrentThreadId());
    #endif
#ifdef DEBUG
    printf("[%ld]DBG: Sleeping for %ld milliseconds...\n",
(int) GetCurrentThreadId(), delay);
#endif
    Sleep(delay);
}
//=========================================================
=============
//
// Function name: UtilPrintNewOrder
//
//=========================================================
=============
void UtilPrintNewOrder(NEW_ORDER_DATA *pNewOrder)
{
    int i;
#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintNewOrder()\n",
(int) GetCurrentThreadId());
#endif
    EnterCriticalSection(&ConsoleCritSec);
    printf("\n[%04ld]\tNewOrder Transaction\n\n", (int)
GetCurrentThreadId());
    printf("Warehouse: %ld\n"
       "District: %ld\n"
       "Date: %02ld/%02ld/%04ld %02ld:%02ld:%02ld\n\n"
       "Customer Number:  %ld\n"
       "Customer Name: %s\n"
       "Customer Credit: %s\n"
       "Cusotmer Discount: %02.2f%%\n\n"
       "Order Number: %ld\n"
       "Warehouse Tax: %02.2f%%\n"
       "District Tax: %02.2f%%\n\n"
       "Number of Order Lines: %ld\n\n",
    (int)    pNewOrder->w_id,
    (int)    pNewOrder->d_id,
    (char *) pNewOrder->o_entry_d.month,
    (char *) pNewOrder->o_entry_d.day,
    (char *) pNewOrder->o_entry_d.year,
    (char *) pNewOrder->o_entry_d.hour,
    (char *) pNewOrder->o_entry_d.minute,
    (char *) pNewOrder->o_entry_d.second,
    (int)    pNewOrder->c_id,
    (char *) pNewOrder->c_last,
    (char *) pNewOrder->c_credit,
    (float)  pNewOrder->c_discount,
    (int)    pNewOrder->o_id,
    (float)  pNewOrder->w_tax,
    (float)  pNewOrder->d_tax,
    (int)    pNewOrder->o_ol_cnt);

    printf("Supp_W Item_Id Item Name                  Qty
Stock B/G Price    Amount   \n");
    printf("------ ------- ------------------------ --- --
--- --- -------- ---------\n");
    for (i=0;i < pNewOrder->o_ol_cnt;i++)
    {
    printf("%04ld   %06ld  %24s   %02ld   %03ld   %1s
%8.2f %9.2f\n",
          (int)    pNewOrder->Ol[i].ol_supply_w_id,
          (int)    pNewOrder->Ol[i].ol_i_id,
      (char *) pNewOrder->Ol[i].ol_i_name,
        (int)  pNewOrder->Ol[i].ol_quantity,
          (int)    pNewOrder->Ol[i].ol_stock,
          (char *) pNewOrder->Ol[i].ol_brand_generic,
          (float)  pNewOrder->Ol[i].ol_i_price,
          (float)  pNewOrder->Ol[i].ol_amount);
    }
    printf("\nTotal: $%05.2f\n\n",
    (float)  pNewOrder->total_amount);

    printf("Execution Status: %s\n\n",
    (char *) pNewOrder->execution_status);

    LeaveCriticalSection(&ConsoleCritSec);

}
//============================================================
============
//
// Function name: UtilPrintPayment
//
//============================================================
============
void UtilPrintPayment(PAYMENT_DATA *pPayment)
{
    char    tmp_data[201];
    char    data_line_1[51];
    char    data_line_2[51];
    char    data_line_3[51];
    char    data_line_4[51];
#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintPayment()\n", (int)
GetCurrentThreadId());
#endif

    EnterCriticalSection(&ConsoleCritSec);

    printf("\n[%04ld]\tPayment Transaction\n\n", (int)
GetCurrentThreadId());
    printf("Date: %02ld/%02ld/%04ld %02ld:%02ld:%02ld\n\n",
        (int)    pPayment->h_date.month,
    (int)    pPayment->h_date.day,
        (int)    pPayment->h_date.year,
    (int)    pPayment->h_date.hour,
    (int)    pPayment->h_date.minute,
    (int)    pPayment->h_date.second);
    printf("Warehouse: %ld\n"
       "District: %ld\n\n",
    (int)    pPayment->w_id,
    (int)    pPayment->d_id);
    printf("Warehouse Address Street 1: %s\n"
       "Warehouse Address Street 2: %s\n",
    (char *) pPayment->w_street_1,
    (char *) pPayment->w_street_2);
    printf("Warehouse Address City: %s\n"
           "Warehouse Address State: %s\n"
           "Warehouse Address Zip: %s\n\n",
    (char *) pPayment->w_city,
    (char *) pPayment->w_state,
    (char *) pPayment->w_zip);
    printf("District Address Street 1: %s\n"
       "District Address Street 2: %s\n",
    (char *) pPayment->d_street_1,
        (char *) pPayment->d_street_2);
    printf("District Address City: %s\n"
       "District Address State: %s\n"
       "District Address Zip: %s\n\n",
    (char *) pPayment->d_city,
(char *) pPayment->d_state,
(char *) pPayment->d_zip);

printf("Customer Number: %ld\n"
   "Customer Warehouse: %ld\n"
   "Customer District: %ld\n",
(int)    pPayment->c_id,
(int)    pPayment->c_w_id,
(int)    pPayment->c_d_id);
printf("Customer Name:   %s %s %s\n"
   "Customer Since:  %02ld-%02ld-%04ld\n",
(char *) pPayment->c_first,
(char *) pPayment->c_middle,
(char *) pPayment->c_last,
(int)    pPayment->c_since.month,
(int)    pPayment->c_since.day,
(int)    pPayment->c_since.year);
printf("Customer Address Street 1: %s\n"
       "Customer Address Street 2: %s\n"
       "Customer Address City: %s\n"
       "Customer Address State: %s\n"
       "Customer Address Zip: %s\n"
   "Customer Phone Number: %s\n\n"
       "Customer Credit: %s\n"
    "Customer Discount: %02.2f%%\n",
(char *) pPayment->c_street_1,
(char *) pPayment->c_street_2,
(char *) pPayment->c_city,
(char *) pPayment->c_state,
(char *) pPayment->c_zip,
(char *) pPayment->c_phone,
(char *) pPayment->c_credit,
(double) pPayment->c_discount);
printf("Amount Paid: $%04.2f\n"
       "New Customer Balance: $%10.2f\n",
(float)  pPayment->h_amount,
(double) pPayment->c_balance);
```

```
    printf("Credit Limit: $%10.2f\n\n",
    (double) pPayment->c_credit_lim);

    if (strcmp(pPayment->c_data," ") != 0)
    {
    strcpy(tmp_data, pPayment->c_data);
    strncpy(data_line_1, tmp_data, 50);
data_line_1[50] = '\0';
    strncpy(data_line_2, &tmp_data[50], 50);
data_line_2[50] = '\0';
    strncpy(data_line_3, &tmp_data[100], 50);
data_line_3[50] = '\0';
    strncpy(data_line_4, &tmp_data[150], 50);
data_line_4[50] = '\0';

    }
    else
    {
    strcpy(data_line_1, " "); strcpy(data_line_2, " ");
    strcpy(data_line_3, " "); strcpy(data_line_4, " ");
    }
    printf("                -------------------------------
------------------\n");
    printf("Customer Data: |%50s|\n", data_line_1);
    printf("               |%50s|\n", data_line_2);
    printf("               |%50s|\n", data_line_3);
    printf("               |%50s|\n", data_line_4);
    printf("                -------------------------------
------------------\n\n");
    printf("Execution Status: %s\n\n",
    (char *) pPayment->execution_status);
    LeaveCriticalSection(&ConsoleCritSec);
}
//=========================================================
============
//
// Function name: UtilPrintOrderStatus
//
//=========================================================
============
void UtilPrintOrderStatus(ORDER_STATUS_DATA *pOrderStatus)
{
    int i;
```

```
#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintOrderStatus()\n",
(int) GetCurrentThreadId());
#endif
   EnterCriticalSection(&ConsoleCritSec);
   printf("\n[%04ld]\tOrder-Status Transaction\n\n", (int)
GetCurrentThreadId());
   printf("Warehouse: %ld\n"
          "District: %ld\n\n",
    (int) pOrderStatus->w_id,
    (int) pOrderStatus->d_id);
   printf("Customer Number: %ld\n"
          "Customer Name: %s %s %s\n\n",
    (int) pOrderStatus->c_id,
    (char *) pOrderStatus->c_first,
    (char *) pOrderStatus->c_middle,
    (char *) pOrderStatus->c_last);
   printf("Customer Balance: $%5.2f\n\n",
    (double) pOrderStatus->c_balance);
   printf("Order Number: %ld\n"
          "Entry Date: %02ld/%02ld/%04ld
%02ld:%02ld:%02ld\n"
          "Carrier Number: %ld\n\n"
          "Number of order lines: %ld\n\n",
    (int) pOrderStatus->o_id,
    (int) pOrderStatus->o_entry_d.month,
    (int) pOrderStatus->o_entry_d.day,
    (int) pOrderStatus->o_entry_d.year,
    (int) pOrderStatus->o_entry_d.hour,
    (int) pOrderStatus->o_entry_d.minute,
    (int) pOrderStatus->o_entry_d.second,
    (int) pOrderStatus->o_carrier_id,
    (int) pOrderStatus->o_ol_cnt);

    printf ("Supply-W    Item-Id    Delivery-Date    Qty
Amount    \n");
   printf ("--------    -------    -------------    ---   -
---------\n");
    for (i=0;i < pOrderStatus->o_ol_cnt; i++)
    {
   printf("%04ld        %06ld     %02ld/%02ld/%04ld
%02ld     %9.2f\n",
    (int) pOrderStatus->OlOrderStatusData[i].ol_supply_w_id,
    (int) pOrderStatus->OlOrderStatusData[i].ol_i_id,
    (int) pOrderStatus-
>OlOrderStatusData[i].ol_delivery_d.month,
    (int) pOrderStatus-
>OlOrderStatusData[i].ol_delivery_d.day,
    (int) pOrderStatus-
>OlOrderStatusData[i].ol_delivery_d.year,
    (int) pOrderStatus->OlOrderStatusData[i].ol_quantity,
    (double) pOrderStatus->OlOrderStatusData[i].ol_amount);
    }
   if (pOrderStatus->o_ol_cnt == 0)
   printf("\nNo Order-Status items.\n\n");
   printf("\nExecution Status: %s\n\n",
   (char *) pOrderStatus->execution_status);
   LeaveCriticalSection(&ConsoleCritSec);
}
//==========================================================
============
//
// Function name: UtilPrintDelivery
//
//==========================================================
============
void UtilPrintDelivery(DELIVERY_DATA *pQueuedDelivery)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintDelivery()\n",
(int) GetCurrentThreadId());
#endif
   EnterCriticalSection(&ConsoleCritSec);
   printf("\n[%04ld]\tDelivery Transaction\n\n", (int)
GetCurrentThreadId());
    printf("Warehouse: %ld\n", (int) pQueuedDelivery->w_id);

   printf("Carrier Number: %ld\n\n", (int) pQueuedDelivery-
>o_carrier_id);
    printf("Execution Status: %s\n\n", (char *)
pQueuedDelivery->execution_status);
   LeaveCriticalSection(&ConsoleCritSec);
}
//==========================================================
============
//
// Function name: UtilPrintStockLevel
//
//=========================================================
============
void UtilPrintStockLevel(STOCK_LEVEL_DATA *pStockLevel)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintStockLevel()\n",
(int) GetCurrentThreadId());
#endif
    EnterCriticalSection(&ConsoleCritSec);
    printf("\n[%04ld]\tStock-Level Transaction\n\n", (int)
GetCurrentThreadId());
    printf("Warehouse: %ld\nDistrict: %ld\n",
    (int) pStockLevel->w_id,
    (int) pStockLevel->d_id);

    printf("Stock Level Threshold: %ld\n\n", (int)
pStockLevel->thresh_hold);
    printf("Low Stock Count: %ld\n\n", (int) pStockLevel-
>low_stock);
    printf("Execution Status: %s\n\n", (char *) pStockLevel-
>execution_status);

    LeaveCriticalSection(&ConsoleCritSec);
}
//=========================================================
============
//
// Function name: UtilError
//
//=========================================================
============
void UtilError(long threadid, char * header, char *msg)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilError()\n", (int)
GetCurrentThreadId());
#endif
    printf("[%ld] %s: %s\n", (int) threadid, header, msg);
}
//=========================================================
============
//
// Function name: UtilFatalError
//
//=========================================================
============
void UtilFatalError(long threadid, char * header, char *msg)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilFatalError()\n", (int)
GetCurrentThreadId());
#endif
    printf("[Thread: %ld]... %s: %s\n", (int) threadid,
header, msg);
    exit(-1);
}
//=========================================================
============
//
// Function name: UtilStrCpy
//
//=========================================================
===========
void UtilStrCpy(char * pDest, char * pSrc, int n)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilStrCpy()\n", (int)
GetCurrentThreadId());
#endif
    strncpy(pDest, pSrc, n);
    pDest[n] = '\0';
}
#ifdef USE_CONMON
//=========================================================
============
//
// Function name: WriteConsoleString
//
//=========================================================
===========
void WriteConsoleString(HANDLE hConMon, char *str, short x,
short y, short color, BOOL pad)
```

```
{
    COORD   dwWriteCoord = {0, 0};
    DWORD   cCharsWritten;
    LPVOID  dummy;
    int     len, i;
#ifdef DEBUG
    printf("[%ld]DBG: Entering WriteConsoleString()\n",
(int) GetCurrentThreadId());
#endif

    dwWriteCoord.X = x;
    dwWriteCoord.Y = y;
    if (pad)
    {
    len = strlen(str);
    if (len < CON_LINE_SIZE)
    {
    for(i=1;i<CON_LINE_SIZE-len;i++)
    {
    strcat(str," ");
    }
    }
    }
    EnterCriticalSection(&ConsoleCritSec);
    switch (color)
    {
    case YELLOW:
    SetConsoleTextAttribute(hConMon,
    FOREGROUND_INTENSITY | FOREGROUND_GREEN |
FOREGROUND_RED | BACKGROUND_BLUE);
        break;
    case RED:
    SetConsoleTextAttribute(hConMon,
    FOREGROUND_INTENSITY | FOREGROUND_RED |
BACKGROUND_BLUE);
        break;
    case GREEN:
    SetConsoleTextAttribute(hConMon,
    FOREGROUND_INTENSITY | FOREGROUND_GREEN |
BACKGROUND_BLUE);
    break;
    }
    SetConsoleCursorPosition(hConMon, dwWriteCoord);
    WriteConsole(hConMon, str, strlen(str), &cCharsWritten,
dummy);
    LeaveCriticalSection(&ConsoleCritSec);
}
#endif
//==========================================================
============
//
// Function name: AddDeliveryQueueNode
//
//==========================================================
============
BOOL AddDeliveryQueueNode(DELIVERY_PTR node_to_add)
{

    DELIVERY_PTRlocal_node;
#ifdef DEBUG
    DELIVERY_PTRptrtmp;
    short           i;
#endif
    EnterCriticalSection(&QueuedDeliveryCritSec);

    if ((local_node = malloc(sizeof(struct delivery_node))
) == NULL)
    {
    printf("ERROR:  problem allocating memory for delivery
queue.\n");
    exit(-1);
    }
    else
    {
    memcpy(local_node, node_to_add, sizeof (struct
delivery_node));

    if (queued_delivery_cnt == 0)
    {
    delivery_head = local_node;
    delivery_head->next_delivery = NULL;
    delivery_tail = delivery_head;
    }
    else
    {
    local_node->next_delivery = NULL;
    delivery_tail->next_delivery = local_node;
    delivery_tail = local_node;
    }
    }

    queued_delivery_cnt++;
#ifdef DEBUG
    i=0;
    printf("Add to delivery list:
%ld\n",queued_delivery_cnt);
    ptrtmp=delivery_head;
    while (ptrtmp != NULL)
    {
    i++;
    printf("%ld - w_id %ld - o_carrier_id %ld - queue_time
%d/%d/%d %d:%d:%d:%d\n",
    i, ptrtmp->w_id, ptrtmp->o_carrier_id,
    ptrtmp->queue_time.wMonth,
    ptrtmp->queue_time.wDay,
    ptrtmp->queue_time.wYear,
    ptrtmp->queue_time.wHour,
    ptrtmp->queue_time.wMinute,
    ptrtmp->queue_time.wSecond,
    ptrtmp->queue_time.wMilliseconds);
    ptrtmp=ptrtmp->next_delivery;
    }
#endif
    LeaveCriticalSection(&QueuedDeliveryCritSec);

    return TRUE;
}
//=========================================================
=============
//
// Function name: GetDeliveryQueueNode
//
//=========================================================
=============
BOOL GetDeliveryQueueNode(DELIVERY_PTR node_to_get)
{

    DELIVERY_PTRlocal_node;
    BOOLrc;
#ifdef DEBUG
    DELIVERY_PTRptrtmp;
    shorti;
#endif

    EnterCriticalSection(&QueuedDeliveryCritSec);

    if (queued_delivery_cnt == 0)
    {
#ifdef DEBUG
    printf("No delivery nodes found.\n");
#endif
    rc = FALSE;
    }
    else
    {
    memcpy(node_to_get, delivery_head, sizeof(struct
delivery_node));
    if (queued_delivery_cnt == 1)
    {
    free(delivery_head);
    delivery_head = NULL;
    queued_delivery_cnt = 0;
    }
    else
    {
    local_node = delivery_head;
    delivery_head = delivery_head->next_delivery;
    free(local_node);
    queued_delivery_cnt--;
    }
#ifdef DEBUG
    i=0;
    printf("Get from delivery list:
%ld\n",queued_delivery_cnt);
    ptrtmp=delivery_head;
    while (ptrtmp != NULL)
    {
```

```
    i++;
    printf("%ld - w_id %ld - o_carrier_id %ld - queue_time
%d/%d/%d %d:%d:%d:%d\n",
    i, ptrtmp->w_id, ptrtmp->o_carrier_id,
    ptrtmp->queue_time.wMonth,
    ptrtmp->queue_time.wDay,
    ptrtmp->queue_time.wYear,
    ptrtmp->queue_time.wHour,
    ptrtmp->queue_time.wMinute,
    ptrtmp->queue_time.wSecond,
    ptrtmp->queue_time.wMilliseconds);
    ptrtmp=ptrtmp->next_delivery;
    }
#endif
    rc = TRUE;

    }
    LeaveCriticalSection(&QueuedDeliveryCritSec);

    return rc;
}
//=========================================================
=============
//
// Function name: WriteDeliveryString
//
//=========================================================
=============
void WriteDeliveryString(char   buf[255])
{
    DWORD   bytesWritten;
    DWORD   retCode;
#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilDeliveryMsg()\n", (int)
GetCurrentThreadId());
#endif
    EnterCriticalSection(&WriteDeliveryCritSec);

    retCode = WriteFile (hDeliveryMonPipe, buf,
PLEASE_WRITE,
      &bytesWritten, NULL);
    LeaveCriticalSection(&WriteDeliveryCritSec);
    }
```

*tpc.inc*

```
    #
    ############################################################
    ################
    #       TPC.INC
    #
    #
    ############################################################
    ################
    # TYPE will be supplied as the type directory.
    EXE_DIR = $(TPC_DIR)\run\ntintel
    OBJ_DIR = $(TPC_DIR)\build\ntintel\obj
    INC_DIR = $(TPC_DIR)\src
    # C compiler flags.
    # NT_WIN32 is always small model.
    # OF will be supplied as the optimizing flag (/Od or /Ot).
    # ZF will be supplied as the debugging flag (none or /Zi).
    # DB will be supplied as a debugging flag.
    CDEFINES = -DWIN32  -DNTWIN32 -Di386 -DDBNTWIN32  -D_X86_ -
    DCONSOLE -D_WINDOWS -D_NTWIN
    CFLAGS = /c /G4 /Gs $(OF) /W2 $(ZF) $(DB) $(DBAPI)
    $(CDEFINES) /DLINT_ARGS=1
    CFLAGSOPT = $(CFLAGS)  /Ot
    CC = cl
    # Linker flags.
    # LF1 will be supplied as the link debugging flag (-
    debug:full)
    # LF2 will be supplied as the link debugging flag (-
    debugtype:cv)
    LFLAGS = -subsystem:console $(LF1) $(LF2) /NODEFAULTLIB:LIBC
    LL = link $(LFLAGS)
    # NTWIN32 libraries
    # BUGBUG: Can't load strings in console subsystem mode yet.
```

```
NTLIBS= $(NTLIB)\kernel32.lib \
    $(NTLIB)\advapi32.lib \
    $(NTLIB)\libcmt.lib
```

# *Appendix C – Tunable Parameters*

## Microsoft Windows NT Version 4.0 Configuration Parameters

There were no Windows NT registry parameters that were changed from their defaults.

## Microsoft SQL Server Version 6.5 Startup Parameters

c:\mssql\binn\sqlservr -c -x -t1081 -t3502 -t812

where

| | |
|---|---|
| -c | start SQL Server independently of the Windows NT Service Control Manager |
| -x | disables the keeping of CPU time and cache-hit ratio statistics |
| -t1081 | allows the index pages a “second” trip through the cache |
| -t3052 | prints a message to the log at the start and end of each checkpoint |
| -t812 | disables checkpoint io buffer sorting |

## Microsoft SQL Server Version 6.5 Configuration Parameters

sp_configure:

```
name                      minimum    maximum    config_value
run_value
--------------------------------- ----------- ----------- ------------ -----------
affinity mask              0        2147483647  0          0
allow updates              0         1          1          1
backup buffer size         1         32         1          1
backup threads             0         32         0          0
cursor threshold          -1        2147483647  -1         -1
```

| | | | | |
|---|---|---|---|---|
| database size | 2 | 10000 | 2 | 2 |
| default language | 0 | 9999 | 0 | 0 |
| default sortorder id | 0 | 255 | 50 | 50 |
| fill factor | 0 | 100 | 0 | 0 |
| free buffers | 20 | 524288 | 23000 | 23000 |
| hash buckets | 4999 | 265003 | 265003 | 265003 |
| language in cache | 3 | 100 | 3 | 3 |
| LE threshold maximum | 2 | 500000 | 200 | 200 |
| LE threshold minimum | 2 | 500000 | 20 | 20 |
| LE threshold percent | 1 | 100 | 0 | 0 |
| locks | 5000 | 2147483647 | 5000 | 5000 |
| LogLRU buffers | 0 | 2147483647 | 1000 | 1000 |
| logwrite sleep (ms) | -1 | 500 | -1 | -1 |
| max async IO | 1 | 1024 | 150 | 150 |
| max lazywrite IO | 1 | 1024 | 45 | 45 |
| max text repl size | 0 | 2147483647 | 65536 | 65536 |
| max worker threads | 10 | 1024 | 255 | 255 |
| media retention | 0 | 365 | 0 | 0 |
| memory | 2800 | 1048576 | 450000 | 450000 |
| nested triggers | 0 | 1 | 1 | 1 |
| network packet size | 512 | 32767 | 4096 | 4096 |
| open databases | 5 | 32767 | 8 | 8 |
| open objects | 100 | 2147483647 | 200 | 200 |
| priority boost | 0 | 1 | 0 | 0 |
| procedure cache | 1 | 99 | 1 | 1 |
| Protection cache size | 1 | 8192 | 15 | 15 |
| RA cache hit limit | 1 | 255 | 4 | 4 |
| RA cache miss limit | 1 | 255 | 3 | 3 |
| RA delay | 0 | 500 | 15 | 15 |
| RA pre-fetches | 1 | 1000 | 3 | 3 |
| RA slots per thread | 1 | 255 | 5 | 5 |
| RA worker threads | 0 | 255 | 0 | 0 |
| recovery flags | 0 | 1 | 0 | 0 |
| recovery interval | 1 | 32767 | 32767 | 32767 |
| remote access | 0 | 1 | 0 | 0 |
| remote conn timeout | -1 | 32767 | 10 | 10 |
| remote login timeout | 0 | 2147483647 | 5 | 5 |
| remote proc trans | 0 | 1 | 0 | 0 |
| remote query timeout | 0 | 2147483647 | 0 | 0 |
| remote sites | 0 | 256 | 0 | 0 |

```
resource timeout          5       2147483647  10      10
set working set size        0       1       0       0
show advanced options         0       1       1       1
SMP concurrency              -1      64      -1      -1
sort pages               64      511      64      64
spin counter             1      2147483647  10000    10000
tempdb in ram (MB)          0       2044     5       5
time slice               50      1000     100     100
user connections           5       32767    100     100
user options              0      4095     0       0
(1 row(s) affected)
```

## Server System Configuration Parameters

```
Board Information
 HP NetServer LH Pro System Board (CFG 1.20)
      System
 Manufacturer ................. Hewlett-Packard Co.
 ID ........................... HWPC141
 Category ..................... SYS
Board slot type ............. Embedded
Readable ID ................. Yes
Amperage ..................... 19000 milliamps
Overlay name ................. HWPC141.OVL
Overlay version ............. 1.20
General System Information ..................... Press 'Enter' to view
System Date and Time ........................... Press 'Enter' to set Date
                         and Time
System Language
  System Language .............................. English
Flexible Disk Drives
  Embedded Controller .......................... Primary
  Flexible Drive A ............................ 1.44MB (3.5-inch)
  Flexible Drive B ............................ Not Installed
Fixed Disk Drives
  Embedded IDE Hard Disk Controller ............ Primary
  Drive 1 ..................................... Not Installed or SCSI
  Drive 2 ..................................... Not Installed or SCSI
  Enhanced Performance - Disk 1 ................ Disabled
  Enhanced Performance - Disk 2 ................ Disabled
```

Security Options

Power-On Password ............................ Press 'Enter' to Set

Network Server Mode .......................... Disabled

Keyboard Lock ................................ Disabled

Video Blanking ............................... Disabled

Power Switch Lock ............................ Disabled

Start from Flexible Disk ..................... Enabled

Writes to Flexible Disk ...................... Enabled

Cache Configuration

Internal Cache Control ....................... Enabled

Secondary Cache Control ...................... Enabled

Video Configuration

Embedded Video Adapter Display ............... Single Display

Primary Video Adapter Type ................... VGA/EGA

System Memory

Base Memory .................................. 640 KB

Total Memory Installed ....................... 1024 MB

Keyboard Options

Numlock On at Boot ........................... Enabled at Startup

Key Click .................................... Off

Keyboard Auto Repeat Control

Keyboard Auto Repeat Delay ................... 1/4 sec

Keyboard Auto Repeat Speed ................... 30.0/sec

Mini-DIN Mouse ................................ Enabled

Dual Serial Ports

Serial Port A ................................ COM1

Serial Port B ................................ COM2

Parallel Port Configuration

Parallel Port ................................ Enabled as LPT1

Parallel Port Mode ........................... Uni-directional mode

Miscellaneous Features

Fast I/O Recovery ............................ Enabled

Alternate A20 Control ........................ Port 92h On, Compatible A20

Extended Memory Report Limit ................ Special - Full Amount

15-16 MB Memory Control ...................... Enabled

Advanced Chipset Configuration

INTEL_82C440FX Registers ..................... Press 'Enter' to edit

Advanced Chipset Configuration

PCEB Registers ............................... Press 'Enter' to edit

EISA NMI Control

Software Generated NMI ....................... Disabled

Bus Timeout NMI .............................. Disabled

Fail-Safe Timer Timeout NMI .................. Disabled

PCI Bridge Timers .............................. 16 PCI Clock Cycle

PCI Busmaster Control During Boot

Slot 9 Busmaster ............................. Enabled

SCSI A Busmaster ............................. Enabled

SCSI B Busmaster ............................. Enabled

Virtual-Wire Mode .............................. Local APIC

MPS Specification .............................. MPS 1.1

RAM Cache Option

RAM Cache Option (00000-C7FFF)

Cache Option for 0-512K .................... Write Back

Cache Option for 512K-640K ................. Write Back

Cache Option for A0000-BFFFF ............... Disabled

Cache Option for C0000-C7FFF ............... Disabled

RAM Cache Option (C8000-FFFFF)

Cache Option for C8000-CFFFF ............... Disabled

Cache Option for D0000-D7000 ............... Disabled

Cache Option for D8000-DFFFF ............... Disabled

Cache Option for E0000-FFFFF ............... Disabled

Extended Memory Cache Option ................. Write Back

PCI SCSI Boot Priority Order ................... SCSI A, B before PCI adapters

Board Information

PCI SCSI Controller

SCSI A

Manufacturer ................. PCI

ID ........................... 90048078

Category ..................... MSD

Board slot type .............. PCI

Readable ID .................. No

Skirt ........................ No

PCI Function 1 ................................. Enabled

Board Information

Device Configuration for Embedded SCSI

SCSI A

Manufacturer ................. Hewlett-Packard Co.

ID ........................... ADP7880

Category ..................... MSD

Board slot type .............. Embedded

Readable ID .................. No

Overlay name ................. ADP7880.OVL

Overlay version .............. 1.00

Wide SCSI Channel Configuration

SCSI Channel Interface ....................... Wide Channel, Single-Ended

SCSI

Host Adapter SCSI ID ......................... 7

SCSI Bus Parity Check ........................ Enabled

BIOS configuration ............................. Press <Enter> to configure

Device configuration ........................... Press <Enter> to configure

Boot Device option ............................. Press <Enter> to configure

Utilities ..................................... Press <Enter> to access

Board Information

PCI SCSI Controller

SCSI B

Manufacturer ................. PCI

ID ........................... 90048078

Category ..................... MSD

Board slot type .............. PCI

Readable ID .................. No

Skirt ........................ No

PCI Function 1 ................................. Enabled

Board Information

Device Configuration for Embedded SCSI

SCSI B

Manufacturer ................. Hewlett-Packard Co.

ID ........................... ADP7880

Category ..................... MSD

Board slot type .............. Embedded

Readable ID .................. No

Overlay name ................. ADP7880.OVL

Overlay version .............. 1.00

Wide SCSI Channel Configuration

SCSI Channel Interface ....................... Wide Channel, Single-Ended

SCSI

Host Adapter SCSI ID ......................... 7

SCSI Bus Parity Check ........................ Enabled

BIOS configuration ............................. Press <Enter> to configure

Device configuration ........................... Press <Enter> to configure

Boot Device option ............................. Press <Enter> to configure

Utilities ...................................... Press <Enter> to access

Board Information

PCI Mass Storage Controller

PCI 9

Manufacturer ................. PCI

ID ........................... 101e9010

Category ..................... MSD

Board slot type .............. PCI

Readable ID .................. No

Skirt ........................ No

PCI Function 1 ................................ Enabled

Board Information

PCI Mass Storage Controller

PCI 8

Manufacturer ................. PCI

ID ........................... 101e9010

Category ..................... MSD

Board slot type .............. PCI

Readable ID .................. No

Skirt ........................ No

PCI Function 1 ................................. Enabled

Board Information

PCI Mass Storage Controller

PCI 7

Manufacturer ................. PCI

ID ........................... 101e9010

Category ..................... MSD

Board slot type .............. PCI

Readable ID .................. No

Skirt ........................ No

PCI Function 1 ................................. Enabled

Board Information

PCI Mass Storage Controller

PCI 6

Manufacturer ................. PCI

ID ........................... 101e9010

Category ..................... MSD

Board slot type .............. PCI

Readable ID .................. No

Skirt ........................ No

PCI Function 1 ................................. Enabled

Board Information

PCI Ethernet Controller

SHARED 5

Manufacturer ................. PCI

ID ........................... 80861229

Category ..................... NET

Board slot type .............. PCI

Readable ID .................. No

Skirt ........................ No

PCI Function 1 ................................. Enabled

Used Resources

| Resource | Slot | Function |
|---|---|---|
| IRQ 0.................. | System | Timer |
| IRQ 1.................. | System | Keyboard |
| IRQ 3.................. | System | Dual Serial Ports |
| IRQ 4.................. | System | Dual Serial Ports |
| IRQ 6.................. | System | Flexible Disk Drives |
| IRQ 7.................. | System | Parallel Port Configuration |
| IRQ 8.................. | System | Real-time Clock |
| IRQ 2(9)............... | PCI 6 | PCI Function 1 |
| IRQ 10.................. | PCI 7 | PCI Function 1 |
| IRQ 10.................. | SHARED 5 | PCI Function 1 |
| IRQ 11.................. | PCI 9 | PCI Function 1 |
| IRQ 12.................. | System | Mini-DIN Mouse |
| IRQ 13.................. | System | Math Coprocessor |
| IRQ 14.................. | System | Fixed Disk Drives |
| IRQ 15.................. | SCSI A | PCI Function 1 |
| IRQ 15.................. | SCSI B | PCI Function 1 |
| IRQ 15.................. | PCI 8 | PCI Function 1 |
| DMA 2.................. | System | Flexible Disk Drives |

| | | |
|---|---|---|
| Port 22h - 23h.......... | System | ESC chip |
| Port 60h................ | System | Keyboard |
| Port 64h................ | System | Keyboard |
| Port 70h - 71h.......... | System | Real-time Clock |
| Port 1F0h - 1F7h........ | System | Fixed Disk Drives |
| Port 2F8h - 2FFh........ | System | Dual Serial Ports |
| Port 378h - 37Fh........ | System | Parallel Port Configuration |
| Port 398h - 399h........ | System | SuperIO |
| Port 3B0h - 3BBh........ | System | Video Configuration |
| Port 3BFh - 3DFh........ | System | Video Configuration |
| Port 3F0h - 3F5h........ | System | Flexible Disk Drives |
| Port 3F6h - 3F7h........ | System | Flexible Disk Drives |
| Port 3F8h - 3FFh........ | System | Dual Serial Ports |
| Port 0CF8h - 0CFFh...... | System | Chip Set |
| Port 0E800h - 0E87Fh.... | PCI 6 | PCI Function 1 |
| Port 0E880h - 0E8FFh.... | PCI 7 | PCI Function 1 |
| Port 0EC00h - 0EC7Fh.... | PCI 8 | PCI Function 1 |
| Port 0ECE0h - 0ECFFh.... | SHARED 5 | PCI Function 1 |
| Port 0F400h - 0F4FFh.... | SCSI B | PCI Function 1 |

```
Port 0F800h - 0F8FFh.... SCSI A     PCI Function 1
Port 0FC00h - 0FC7Fh.... PCI 9      PCI Function 1


Memory
Address      Amount
0...............640K.... System     System Memory
0A0000h.........128K.... System     Video Configuration
0C0000h..........32K.... System     Video Configuration
0C8000h...........8K.... PCI 9      PCI Function 1
0E0000h.........128K.... System     System BIOS ROM



1M...............14M.... System     Ext Memory
15M..............1M.... System      Miscellaneous Features
16M.............48M.... System      Ext Memory
64M.............64M.... System      Ext Memory
128M............64M.... System      Ext Memory
192M............64M.... System      Ext Memory
256M............64M.... System      Ext Memory
320M............64M.... System      Ext Memory
384M............64M.... System      Ext Memory
448M............64M.... System      Ext Memory
512M............64M.... System      Ext Memory above 512M
576M............64M.... System      Ext Memory above 512M
640M............64M.... System      Ext Memory above 512M
704M............64M.... System      Ext Memory above 512M
768M............64M.... System      Ext Memory above 512M
832M............64M.... System      Ext Memory above 512M
896M............64M.... System      Ext Memory above 512M
960M............64M.... System      Ext Memory above 512M
0FE9FE000h........4K.... SHARED 5   PCI Function 1
4074M............1M.... SHARED 5    PCI Function 1
0FEDFE000h........4K.... SCSI B     PCI Function 1
0FEDFF000h........4K.... SCSI A     PCI Function 1
```

Available Resources

ÄÄÄIRQsÄÄÂÄÄÄDMAsÄÄÂÄÄÄÄÄISA I/O PortsÄÄÄÂÄMemory AmountÄÄÄAddressÄÄÄÄ

```
5     0      24h -   5Fh       24K    0CA000h
      1      61h -   63h       64K    0D0000h
      3      65h -   6Fh
      5      72h -  1EFh
      6     1F8h -  2F7h
      7     300h -  377h
            380h -  397h
            39Ah -  3AFh
            3BCh -  3BEh
            3E0h -  3EFh

                System Specifications
Slot      Slot      Board     Accept  Max     Bus-    Slot
Name      Type      ID        Skirted Length  master  Tag(s)
ÄÄÄÄÄÄÄÄÄÄ ÄÄÄÄÄÄÄÄÄÄ ÄÄÄÄÄÄÄ ÄÄÄÄÄÄÄ
ÄÄÄÄÄÄ ÄÄÄÄÄÄ ÄÄÄÄÄÄÄÄÄÄ
SCSI A    PCI       90048078  Yes     341mm   Yes
SCSI A    Other     ADP7880   Yes     341mm   No      ADP7880
SCSI B    PCI       90048078  Yes     341mm   Yes
SCSI B    Other     ADP7880   Yes     341mm   No      ADP7880
PCI 9     PCI       101e9010  Yes     341mm   Yes
PCI 8     PCI       101e9010  Yes     341mm   Yes
PCI 7     PCI       101e9010  Yes     341mm   Yes
PCI 6     PCI       101e9010  Yes     341mm   Yes
SHARED 5  EISA or PCI 80861229 Yes    341mm   Yes
EISA 4    EISA      (Empty)   Yes     341mm   Yes
EISA 3    EISA      (Empty)   Yes     341mm   Yes
EISA 2    EISA      (Empty)   Yes     341mm   Yes
EISA 1    EISA      (Empty)   Yes     341mm   Yes

Nonvolatile memory ................. 6K
```

## Disk Array Configuration Parameters

```
NetRAID DiskArray Display
Utility version 1.01

HA #2 FwVersion = Hi64,
BIOS Version = 1.37

HA #3 FwVersion = Hi64,
BIOS Version = 1.37

HA #4 FwVersion = Hi64,
BIOS Version = 1.37

HA #5 FwVersion = Hi64,
BIOS Version = 1.37

No of RAIDCard Adapters
Found = 4

ADAPTER MAPPINGS :
Adapter = 0  Device Location
= \\.\Scsi2:

Adapter = 1  Device Location
= \\.\Scsi3:

Adapter = 2  Device Location
= \\.\Scsi4:

Adapter = 3  Device Location
= \\.\Scsi5:
----------------------------------------------------------------
 NetRAID HA-0
LogicalDrives Found  = 1

Logical Drive : 1
Raid Level : 0                    Spans : 3       Read Ahead
: No
Stripe Size : 8K                  Status : Optimal Write Mode :
Write Through
DirectIO : Disabled               Num Stripes : 5
                                   Physical Drives ###
Channel  ID    ConfiguredSize
                                        0     1      4066
MB
                                        0     2      4066
MB
                                        0     3      4066
MB
                                        0     4      4066
MB
                                        0     5      4066
MB
                                        1     1      4066
MB
                                        1     2      4066
MB
                                        1     3      4066
MB
```

```
                                        1    4    4066
MB

                                        1    5    4066
MB

                                        2    1    4066
MB

                                        2    2    4066
MB

                                        2    3    4066
MB

                                        2    4    4066
MB

                                        2    5    4066
MB

----------------------------------------------------------------

                         NetRAID HA-1

                         LogicalDrives Found  = 1

Logical Drive : 1

Raid Level : 0                 Spans : 3       Read Ahead
: No

Stripe Size : 8K               Status : Optimal Write Mode :
Write Through

DirectIO : Disabled            Num Stripes : 5
```

```
                              Physical Drives ###
Channel  ID   ConfiguredSize

                                        0    1    4066
MB

                                        0    2    4066
MB

                                        0    3    4066
MB

                                        0    4    4066
MB

                                        0    5    4066
MB

                                        1    1    4066
MB

                                        1    2    4066
MB

                                        1    3    4066
MB

                                        1    4    4066
MB

                                        1    5    4066
MB

                                        2    1    4066
MB
```

```
                                       2    2    4066
MB
                                       2    3    4066
MB
                                       2    4    4066
MB
                                       2    5    4066
MB
---------------------------------------------------------------
                              NetRAID HA-2
                              LogicalDrives Found  = 1
Logical Drive : 1
Raid Level : 0                  Spans : 3      Read Ahead
: No
Stripe Size : 8K              Status : Optimal Write Mode :
Write Through
DirectIO : Disabled           Num Stripes : 5
                              Physical Drives ###
Channel  ID   ConfiguredSize
                                       0    1    4066
MB
                                       0    2    4066
MB
                                       0    4    4066
MB
                                       0    5    4066
MB
                                       0    6    4066
MB
                                       1    1    4066
MB
                                       1    2    4066
MB
                                       1    4    4066
MB
                                       1    5    4066
MB
                                       1    6    4066
MB
                                       2    1    4066
MB
                                       2    2    4066
MB
                                       2    4    4066
MB
                                       2    5    4066
MB
```

```
                                        2     6     4066
MB
---------------------------------------------------------------
                                NetRAID HA-3
                                LogicalDrives Found  = 1
Logical Drive : 1
Raid Level : 0                    Spans : 3     Read Ahead
: No
Stripe Size : 8K                Status : Optimal Write Mode :
Write Through
DirectIO : Disabled             Num Stripes : 5
                                Physical Drives ###
Channel  ID    ConfiguredSize
                                        0     1     8677
MB
                                        0     2     8677
MB
                                        0     4     8677
MB
                                        0     5     8677
MB
                                        0     6     8677
MB
                                        1     1     8677
MB
                                        1     2     8677
MB
                                        1     4     8677
MB
                                        1     5     8677
MB
                                        1     6     8677
MB
                                        2     1     8677
MB
                                        2     2     8677
MB
                                        2     4     8677
MB
                                        2     5     8677
MB
                                        2     6     8677
MB
```

## Tuxedo UBBconfig

```
## 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:

#                                       IPCKEYsome decent
IPCKEY, should be different for each config

#                                       ROOTDIR

#                                       TUXCONFIG

#                                       APPDIR

#                                       ULOGDIR

#


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

*RESOURCES

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

                                        IPCKEY2220001

                                        PERM0666

                                        MASTER client1


  MAXACCESSERS                          1700# 1024 or more

  MAXGTT                                1024

  MAXSERVERS                            45

  MAXSERVICES                           210   # 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).

                                     SCANUNIT60

                                     SANITYSCAN5

                                     DBBLWAIT1

                                     BBLQUERY30

                                     BLOCKTIME5

#

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

*MACHINES

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

DEFAULT:

          TUXCONFIG="/project/iti/confs/TUXconfig.client1"

                                        ROOTDIR="/project/iti"


APPDIR="/project/tpcc/bin"


ULOGPFX="/tmp/TUXEDO_LOG"
```

```
# for debugging, put both into the same log on the same machine

#
ULOGPFX="/home/iti/confs/tpcc/ULOG"

# but for a big run, need some space, and want them local to the

# machine rather than across the net.


# Leave TUXCONFIG alone on the MASTER machine; over-ride
for each

# other machine?

client1                              LMID=client1

     TUXCONFIG="/project/iti/confs/TUXconfig.client1"

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

*GROUPS

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

group1          LMID=client1

              GRPNO=1

group2          LMID=client1

              GRPNO=2

group3          LMID=client1

              GRPNO=3
```

```
group4      LMID=client1
          GRPNO=4
group5      LMID=client1
          GRPNO=5
group6      LMID=client1
          GRPNO=6
group7      LMID=client1
          GRPNO=7
group8      LMID=client1
          GRPNO=8

#---------------------------------------------------------------------
*NETWORK
#---------------------------------------------------------------------

#---------------------------------------------------------------------
*SERVERS
#---------------------------------------------------------------------
#
# "--" is application-specific arguments to be passed to server
# "-n" is designed to specify server-id

service SRVGRP=group1
                                CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n1"
    RQADDR=tpcc_1 SRVID=1

service SRVGRP=group1
                                CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n2"
    RQADDR=tpcc_2 SRVID=2

service SRVGRP=group1
                                CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n3"
    RQADDR=tpcc_3 SRVID=3

service SRVGRP=group1
                                CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n4"
    RQADDR=tpcc_4 SRVID=4
```

```
service SRVGRP=group1

                                                    CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n5"

       RQADDR=tpcc_5 SRVID=5


service SRVGRP=group2

                                                    CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n6"

       RQADDR=tpcc_6 SRVID=6


service SRVGRP=group2

                                                    CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n7"

       RQADDR=tpcc_7 SRVID=7


service SRVGRP=group2

                                                    CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n8"

       RQADDR=tpcc_8 SRVID=8


service SRVGRP=group2

                                                    CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n9"

       RQADDR=tpcc_9 SRVID=9


service SRVGRP=group2

                                                    CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n10"

       RQADDR=tpcc_10 SRVID=10


service SRVGRP=group3

                                                    CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n11"

       RQADDR=tpcc_11 SRVID=11


service SRVGRP=group3

                                                    CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n12"

       RQADDR=tpcc_12 SRVID=12


service SRVGRP=group3
```

```
                                          CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n13"

     RQADDR=tpcc_13 SRVID=13


service SRVGRP=group3

                                          CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n14"

     RQADDR=tpcc_14 SRVID=14


service SRVGRP=group3

                                          CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n15"

     RQADDR=tpcc_15 SRVID=15


service SRVGRP=group4

                                          CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n16"

     RQADDR=tpcc_16 SRVID=16


service SRVGRP=group4

                                          CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n17"

     RQADDR=tpcc_17 SRVID=17


service SRVGRP=group4

                                          CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n18"

     RQADDR=tpcc_18 SRVID=18


service SRVGRP=group4

                                          CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n19"

     RQADDR=tpcc_19 SRVID=19


service SRVGRP=group4

                                          CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n20"

     RQADDR=tpcc_20 SRVID=20


service SRVGRP=group5

                                          CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n21"

     RQADDR=tpcc_21 SRVID=21
```

```
service SRVGRP=group5
                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n22"
    RQADDR=tpcc_22 SRVID=22

service SRVGRP=group5
                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n23"
    RQADDR=tpcc_23 SRVID=23

service SRVGRP=group5
                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n24"
    RQADDR=tpcc_24 SRVID=24

service SRVGRP=group5
                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n25"
    RQADDR=tpcc_25 SRVID=25

service SRVGRP=group6
                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n26"
    RQADDR=tpcc_26 SRVID=26

service SRVGRP=group6
                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n27"
    RQADDR=tpcc_27 SRVID=27

service SRVGRP=group6
                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n28"
    RQADDR=tpcc_28 SRVID=28

service SRVGRP=group6
                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n29"
    RQADDR=tpcc_29 SRVID=29

service SRVGRP=group6
```

```
                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n30"

    RQADDR=tpcc_30 SRVID=30


service SRVGRP=group7

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n31"

    RQADDR=tpcc_31 SRVID=31


service SRVGRP=group7

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n32"

    RQADDR=tpcc_32 SRVID=32


service SRVGRP=group7

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n33"

    RQADDR=tpcc_33 SRVID=33


service SRVGRP=group7

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n34"

    RQADDR=tpcc_34 SRVID=34


service SRVGRP=group7

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n35"

    RQADDR=tpcc_35 SRVID=35


service SRVGRP=group8

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n36"

    RQADDR=tpcc_36 SRVID=36


service SRVGRP=group8

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n37"

    RQADDR=tpcc_37 SRVID=37


service SRVGRP=group8

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n38"

    RQADDR=tpcc_38 SRVID=38
```

```
service SRVGRP=group8

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n39"

     RQADDR=tpcc_39 SRVID=39


service SRVGRP=group8

                                        CLOPT="-s NEWO_SVC -s
PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n40"

     RQADDR=tpcc_40 SRVID=40

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

*SERVICES

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

*ROUTING

#---------------------------------------------------------------------
```

## HP-UX Configuration - Clients

```
****************************************

* $Source: /usr/local/kcs/sys.SSR10_800/filesets.info/CORE-
KRN/RCS/generic,v $

* $Revision: 1.2.71.3 $                    $Author: craig $

* $State: Exp $                            $Locker: CRT $

* $Date: 94/05/23 14:52:23 $

*


******************************************************
**************************

* SCSI drivers

******************************************************
**************************

scsi1

target

tape2

disc3


******************************************************
**************************

* Networking

******************************************************
**************************
```

```
lan2

lan3

dlpi

inet

uipc

nm

ni


****************************************************************************
* Misc drivers
****************************************************************************

mux2


****************************************************************************
* Subsystems
****************************************************************************

cdfs

nfs

prf


****************************************************************************
* Streams, DLIP, and Streams-based PTY Drivers/Modules
* Note: To remove the Streams PTY driver from the dfile, you need to
* yank out the following items:
* ptm, pts. ldterm, ptem, pckt, and nstrpty 60
****************************************************************************

hpstreams

clone

strlog

sad

echo

sc

timod

tirdwr

pipedev
```

```
pipemod

ffs


*****************************************************
***********************

* Tunables

*****************************************************
***********************

swap default

default_disk_ir    0

*

bufpages        1024

maxusers        3100

maxswapchunks   2048

maxuprc         5000

nproc           5050

*

msgmap          (2*MSGMNI)

msgmax          4096

msgmnb          65536

msgmni          4096

msgseg          20000

msgssz          200

msgtql          (MSGMAP+2)

*

nfile           15000

nflocks         800

ninode          15000

npty            100

*

semaem          16384

semmap          2048

semmni          4096

semmns          4096

semmnu          4096

semume          32

semvmx          32767

*

shmmni          50

shmmax          0x7fffffff

shmseg          12
```

```
*
timezone        480
```

# Appendix D – Disk Storage

180-day and 8 hour Space Calculations are provided below:

# *Appendix E – Quotations*

All quotes can be found on the following pages.