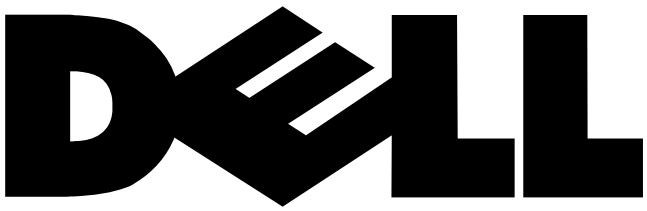


TPC Benchmark™ C
Full Disclosure Report
for
Dell Computer Corporation PowerEdge 6300
Using
Microsoft SQL Server 7.0 Enterprise Edition
and
Microsoft Windows NT Server 4.0 Enterprise Edition

Second Edition

May 10, 1999

First Printing, March 28, 1999
Second Edition (repricing only), May 10, 1999

The pricing information given in this FDR is accurate as of the publication date, March 28, 1999, and generally available.

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

All performance data contained in this report were obtained in a rigorously controlled environment. Actual performance experienced by a particular customer may vary due to differences in system layout and configuration, hardware and/or software revision levels, and background system activity. The content of this document is for informational purposes only.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark™ C test conducted on the Dell Computer Corporation PowerEdge 6300. The tests were run in a client/server configuration using three PowerEdge 2300's as clients. The operating system used for the benchmark was Microsoft NT Server 4.0 Enterprise Edition for the server and Microsoft NT Server 4.0 on the clients. The database was the Microsoft SQL Server Enterprise Edition 7.0. Tuxedo 6.3 Core Functional Services (CFS) provided the database connection queues. All tests were done in compliance with Revision 3.4 of the Transaction Processing Council's TPC Benchmark™ C Standard Specification. Two standard TPC Benchmark™ 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
Dell Computer Corporation PowerEdge 6300	Microsoft Windows NT 4.0 Enterprise Edtn. Microsoft SQL Server 7.0 EE Tuxedo 6.3 CFS	$404,886	23460.57	17.26	June 1, 1999

Auditor

The results of the benchmark and test methodology used to produce the results were audited by Tom Sawyer of Performance Metrics and have fully met the TPC-C rev 3.4 specifications.

Additional copies of this Full Disclosure Report can be obtained from either the Transaction Processing Performance Council or Dell Computer Corporation 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, fax 295-9768
www.tpc.org

or

Dell Computer Corporation
1 Dell Drive
Round Rock, TX 78682
Attention: Dave Jaffe

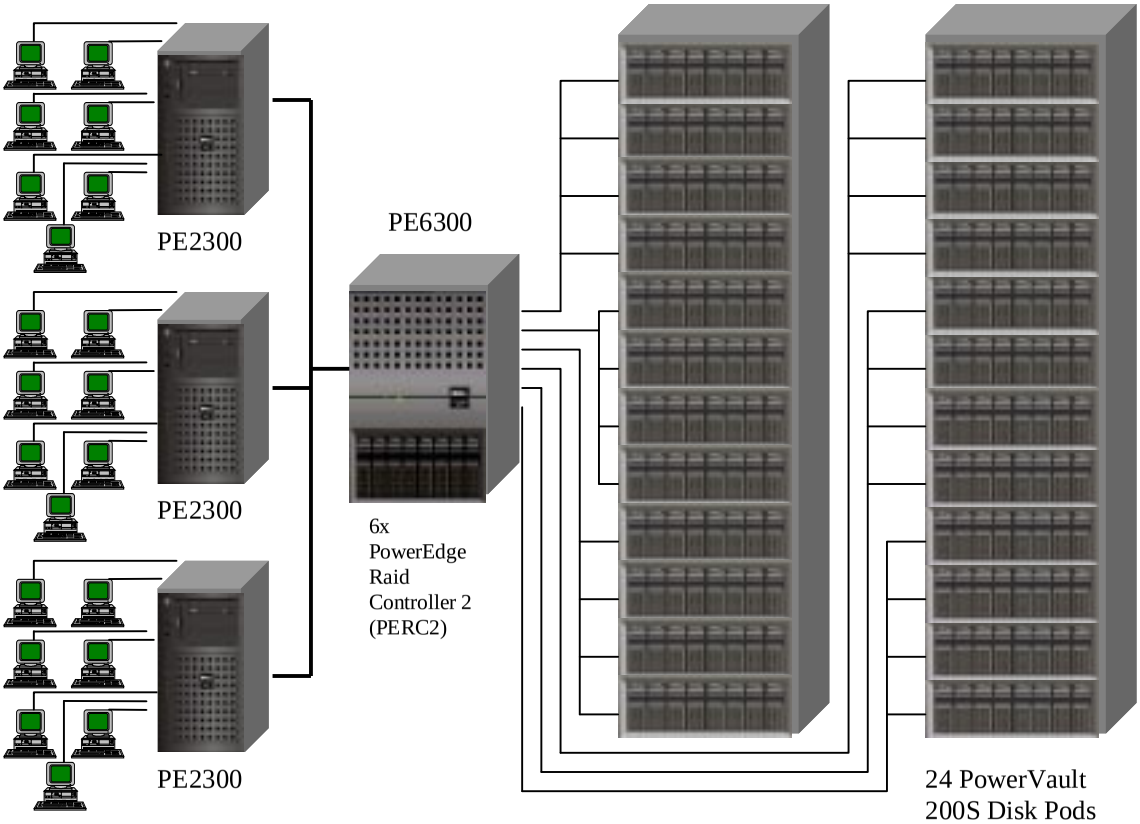
PowerEdge 6300

Client/Server w/3 PE2300 Front Ends

TPC-C Rev 3.4
Report Date
March 28, 1999
Repriced: May 10, 1999

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$404,886	**23,460.57 tpmC**	**$17.26 / tpmC**	June 1, 1999

Processors	Database Manager	OS	Other Software	Number of Users
4 x Pentium® III Xeon™ Processors 500 MHz 2MB L2 Cache	Microsoft SQL Server 7.0 Enterprise Edition	Microsoft WindowsNT Server4.0 Enterprise Edition	Tuxedo 6.3 CFS Microsoft Internet Information Server Microsoft Visual C++	18750



System Component	Server		Each Client	
Processors Cache	4	Pentium® III Xeon™ @ 500MHz 2MB	2	Pentium® II @ 400MHz 512 KB
Memory		4096 MB		512 MB
Disk Controllers	6 2	PowerEdge Raid Controller 2 Adaptec On-Board	1	Adaptec On-Board
Disk Drives	185 8	9 GB SCSI (8.474 GB useable) 18 GB SCSI (16.9 GB useable)	1	4.5 GB
Total Storage		1,703 GB		4.5 GB
Other	1 1 1	CD-ROM PCI Ethernet NIC Tape Drive	1 4 2	CD-ROM PCI Ethernet NIC PCI Ethernet Dual Port NIC

Dell Computer	PowerEdge 6300	TPC-C REV 3.4 EXECUTIVE SUMMARY PAGE 2 OF 2
Corporation	Client/Server	Report Date: 10-May-1999

Description	Part Number	Third Party		Unit Price	Qty	Extended Price	5 yr. Maint. Price
Server Hardware		**Brand**	**Pricing**				
Dell PowerEdge6300 w/1 PentiumIII Xeon500/2MB L2	220-3087		1	11,113	1	11,113	1,999
3 Pentium III Xeon 500 / 2MB L2	311-0875		1	12,999	1	12,999	0
4 GB RAM, 16 x 256 EDO DIMMs	311-0481		1	11,520	1	11,520	0
1x6 Backplane	311-0485		1	399	1	399	0
SMB Port	310-0268		1	20	1	20	0
9 GB Ultra-2 LVD SCSI 10K RPM Disk	340-0595		1	599	1	599	0
Intel Ether Express PRO 100+ PCI ENET Adapter	430-0111	Intel	1	69	1	69	0
Readyware fee for NIC	365-1234		1	15	1	15	0
12/24 GB DDS-3 DAT TBU	340-6121		1	749	1	749	0
Dell 800F Monitor	320-0071		1	199	1	199	0
PERC 2 w/32MB ECC Cache RAM	340-1029		1	1,899	4	7,596	0
PERC 2 w/128MB ECC Cache RAM	340-1032		1	2,149	2	4,298	0
					Subtotal	49,576	1,999
PowerVault Disk Subsystem							
200S Disk Pod, non-redundant PS	220-0644		1	2,584	24	62,016	36,336
SCSI Cables	310-0313		1	69	24	1,656	0
9 GB Ultra-2 LVD SCSI 10K RPM Disk *	340-1394		1	599	184	110,216	0
18 GB Ultra-2 LVD SCSI 10K RPM Disk *	340-1395		1	1,099	8	8,792	0
42U Rack	220-0605		1	2,199	2	4,398	0
					Subtotal	187,078	36,336
Server Software							
MS SQL Server, Enterprise Ed. 7.0, Unlimited Users **		Microsoft	2	28,999	1	28,999	10,475
MS Windows NT Server 4.0 Enterprise Ed. w/25CAL **		Microsoft	2	3,999	1	3,999	0
					Subtotal	32,998	10,475
Client Hardware							
Dell PowerEdge 2300, 400 MHz Pentium II w/ 512KB	220-3172		1	1,589	3	4,767	5,397
2nd Processor,400MHz Pentium II w/512K	311-0431		1	399	3	1,197	
512MB RAM, 4 DIMMs	311-0440		1	1,082	3	3,246	
1x6 Backplane	311-0446		1	199	3	597	0
4.5GB LVD SCSI Disk Drive,	340-0929		1	399	3	1,197	
Dell 800F Monitor	320-0071		1	199	3	597	
Intel Ether Express PRO 100/B PCI ENET Adapter	430-0008	Intel	1	69	12	828	0
Adaptec 6922 10/100 Dual Port Ethernet NIC	85162-55	Adaptec	1	304	6	1,824	0
Readyware fee for NIC (per machine)	365-1234		1	15	3	45	0
					Subtotal	14,298	5,397
Client Software							
MS Window NT Server 4.0 w/ 10 CAL **		Microsoft	2	809	3	2,427	0
MS Visual C++ 32 bit Edition **		Microsoft	2	499	1	499	0
Tuxedo 6.3 Core Functionality Services for NT		BEA	3	3,000	3	9,000	7,200
					Subtotal	11,926	7,200
User Connectivity							
5-Port 100BaseTX Hub ***	FEHUB05W	Linksys	1	73	3	219	0
8+1 Port 10BaseT Ethernet Hub ***	TP1008C	Compex	1	29.47	2,588	76,268	0
					Subtotal	76,487	0
					Other Discounts	($28,884)	0
					Total	**$343,479**	**$61,407**

Notes: * Maint. included in PowerVault 200S disk pod
** All Microsoft maintenance is covered by the maintenance costs of Microsoft SQL Server
*** 10% or minimum 2 spares are added in place of onsite service (products have a five year return-to-vendor warranty) Pricing: 1 - Dell 2 - Microsoft 3 - BEA

Audited by Tom Sawyer of Performance Metrics, Inc.

Five-Year Cost of Ownership:	**$404,886**
tpmC Rating:	**23460.57**
$ / tpmC:	**17.26**

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

MQTh, computed Maximum Qualified Throughput 23,460.57 tpmC
% throughput difference, reported & reproducibility runs 0.13%

Response Times (in seconds)	Average	90th	Max
- Neworder	0.36	0.6	2.89
- Payment	0.25	0.5	2.46
- Order Status	0.28	0.5	2.92
- Delivery (interactive portion)	0.11	0.12	0.21
- Delivery (deferred portion)	0.56	0.97	2.59
- Stock-Level	1.63	2.3	3.13
- Menu	0.11	0.12	0.74

Response time delay added for emulated components Menu 0.1
Resp 0.1

Transaction Mix, in percent of total transactions

- New-Order 44.88 %
- Payment 43.05 %
- Order-Status 4.02 %
- Delivery 4.02 %
- Stock-Level 4.00 %

Keying/Think Times (in seconds),	Min		Average		Max	
- New-Order	18.00	0.0	18.01	12.07	18.05	120.82
- Payment	3.00	0.0	3.01	12.08	3.06	120.82
- Order-Status	2.00	0.0	2.00	10.10	2.04	100.81
- Delivery	2.00	0.0	2.00	5.10	2.04	50.80
- Stock-Level	2.00	0.0	2.00	5.10	2.04	50.80

Test Duration
- Ramp-up time 21 minutes
- Measurement interval 30 minutes
- Number of checkpoints 1
- Checkpoint interval 30 minutes
- Number of transactions (all types) completed in measurement interval 1,568,124

Table of Contents

Introduction

Document Structure

The contents of this report are determined by the TPC Benchmark C Standard Specification Revision 3.4, written and approved by the Transaction Processing Performance Council (TPC). The format of this report is based on this specification. Most sections of this report begin with the specification requirements printed in italic type, immediately followed by the detail in plain type of how Dell Computer Corporation 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.

Benchmark Overview

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.

Although these specifications express implementation in terms of a relational data model with conventional locking scheme, the database may be implemented using any commercially available database management system (DBMS), database server, file system, or other data repository that provides a functionally equivalent implementation. The terms "table", "row", and "column" are used in this document only as examples of logical data structures.

TPC-C uses terminology and metrics that are similar to other benchmarks, originated by the TPC or others. Such similarity in terminology does not in any way imply that TPC-C results are comparable to other benchmarks. The only benchmark results comparable to TPC-C are other TPC-C results conformant with the same revision.

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 is a Dell PowerEdge 6300 server driven by three Dell PowerEdge 2300 clients. The clients and server are networked together via a 100BaseT hub. Five remote terminal emulator (RTE) systems (PowerEdge 2100's) emulate 18,750 users executing the standard TPC-C workload. The RTE's are connected to the three clients at 10Mbit/sec, half duplex. Microsoft NT4.0 Server was the operating system used - Enterprise Edition on the server and standard edition on the clients. Microsoft SQL Server 7.0, Enterprise Edition was the database on the server machine.

The PowerEdge 6300 motherboard is based on the Intel 82450NX chipset and can hold up to four Pentium® III Xeon™ processors (500MHz with 2 MB L2 cache on each). The system has 7 hot plug ready PCI I/O slots based on dual-peer 64-bit PCI buses. Four of the slots are 64-bit and three are 32-bit. The measured configuration used 4 Gbytes of RAM, which was achieved by using sixteen 256 Mbyte DIMMs on the memory board.

The PowerEdge 6300 has two integrated Ultra2 LVD SCSI controllers (based on the Adaptec AIC-7890 chip) which offer a peak transfer rate of 80 MB/s each. One of the integrated controllers was cabled to the internal hard drive backplane, which contained one 9 GB disk drive containing the operating system. In addition, six PowerEdge RAID Controller 2 (PERC2) 4-channel controllers were installed in six PCI slots and connected to twenty-four PowerVault 200S disk pods, which hold 8 disks each. Of these 192 external disks, 184 were 9 GB disks and contained the database data and the remaining 8 were 18 GB and contained the transaction log.

Each of the clients had two 400 MHz Pentium II processors with 512 Kbyte of L2 cache, 512 Mbyte of RAM, one 4.5 GB hard disk and six Intel Ether Express Pro 100/B ethernet adapters. On each client one of the Intel ethernet adapters was connected to the server through a 100BaseT hub and the other adapters provided 5 network segments to the RTE machines. The network interface cards between Clients and RTEs were fixed at 10 Mbit/sec, half duplex. In the priced configuration, to meet the requirement of maximum 1024 emulated user per network segment, two of the Intel ethernet adapters on each client are replaced with Adaptec 6922 Dual Port Ethernet adapters to give a total of seven segments to handle the 6250 emulated users per client.

General Items

Test Sponsor

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

Dell Computer Corporation was the test sponsor of this TPC Benchmark™ C.

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 application consists of a Remote Terminal Emulator (RTE) program emulating a set of users entering TPC-C transactions through web browsers, and communicating with Client machines running the Microsoft Internet Information Server (IIS) web server. The Client machines use the BEA Tuxedo™ transaction monitor to communicate with the database server machine.

The Remote Terminal Emulator program is a custom, multithreaded, C program split into an RTE master program and an RTE slave program. A single RTE master program runs on one PE2100 uniprocessor Pentium Pro 200MHz machine and drives 15 RTE slave processes (one per network segment), distributed across 5 other PE2100s. The RTE master consists of a main program that creates and controls one thread for each RTE slave process. These threads communicate through named pipes to the RTE slave processes. In turn, each RTE slave process consists of a main program which creates and controls one thread for each user it will emulate. Each thread opens up a persistent socket to the web server running on one of the Client machines, then issues transactions according to the specified parameters. Transaction data is stored in a data structure which is dumped to a binary file at the end of the run, or when selected by the test operator. The RTE parameters – login rate, ramp up rate, run time, screen update interval – as well as the transaction parameters – mix, key time, think time, etc. – are specified in two files. All data from the run in progress, including average tpmC over the last 1, 5, 10 and 30 minutes are shown on a single console.

On each Client machine IIS loads a custom Microsoft Internet Information Server Application Programming Interface dynamic link library (ISAPI DLL) program that communicates with the emulated web browsers through the HTTP protocol and with the database server through the Tuxedo transaction monitor and the Microsoft DBLIB interface. The ISAPI DLL, tpcc.dll, supplies fill-in screens to the user for each transaction, then parses the data in each request, makes a Tuxedo call and hands the data to the appropriate Tuxedo server. Tuxedo manages the request in its queue and then makes a DBLIB call upon the database server running MS SQL Server. The resulting data is passed back to the tpcc.dll where it is formatted into HTML and sent back to the user's browser. Tpcc.dll is a C program featuring efficient, robust input handling and user screen management.

The Client back end database connections are implemented with a multithreaded version of Tuxedo 6.3 Core Functionality Services for NT. Two TPC-C transaction servers are implemented as separate executables. One server (TMDL) handles only the Delivery transaction and the other (TMT4) handles the remaining four. Multiple instances of each transaction server are booted by Tuxedo prior to the start of the simulation. Deferred Delivery transactions are handled by asynchronous Tuxedo calls in which control is returned immediately to the ISAPI DLL while Tuxedo completes the Delivery database stored procedure. The other 4 transactions are handled by synchronous Tuxedo calls which don't return to tpcc.dll until the database access is complete.

The web Client and Tuxedo server code is listed in Appendix A.

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, Windows NT Server, Internet Information Service and Tuxedo parameters used in this benchmark.

Appendix D contains the 180 day space calculations.

Configuration Diagrams

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

Figures 1 and 2 respectively show the measured and priced full client/server configurations. The system under test (SUT) in the measured system was identical to the priced one with the exception that, to meet the requirement of maximum 1024 emulated user per network segment, two of the Intel ethernet adapters on each client are replaced with Adaptec 6922 Dual Port Ethernet adapters to give a total of seven segments to handle the 6250 emulated users per client.

Figure 1: Measured Configuration

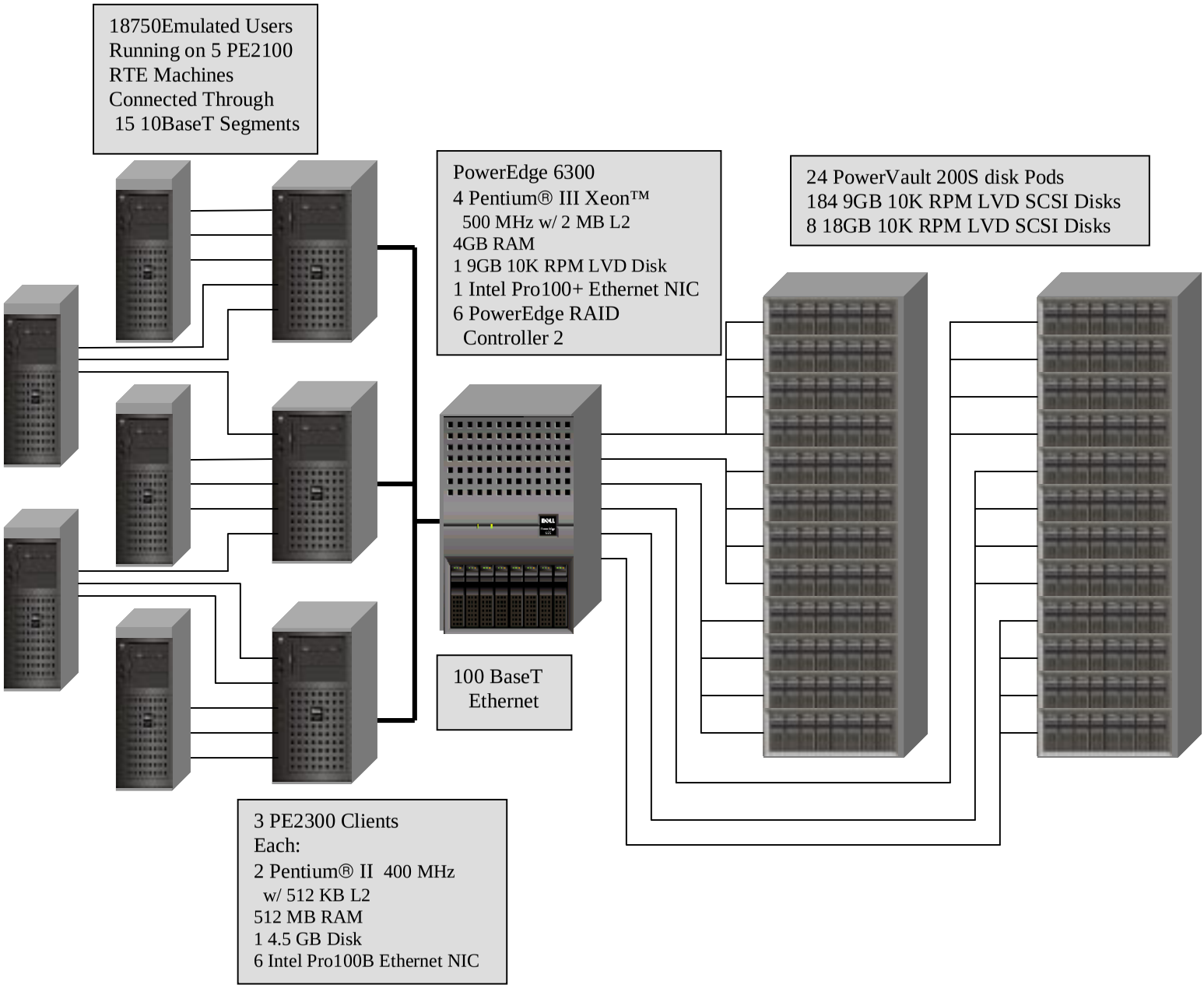
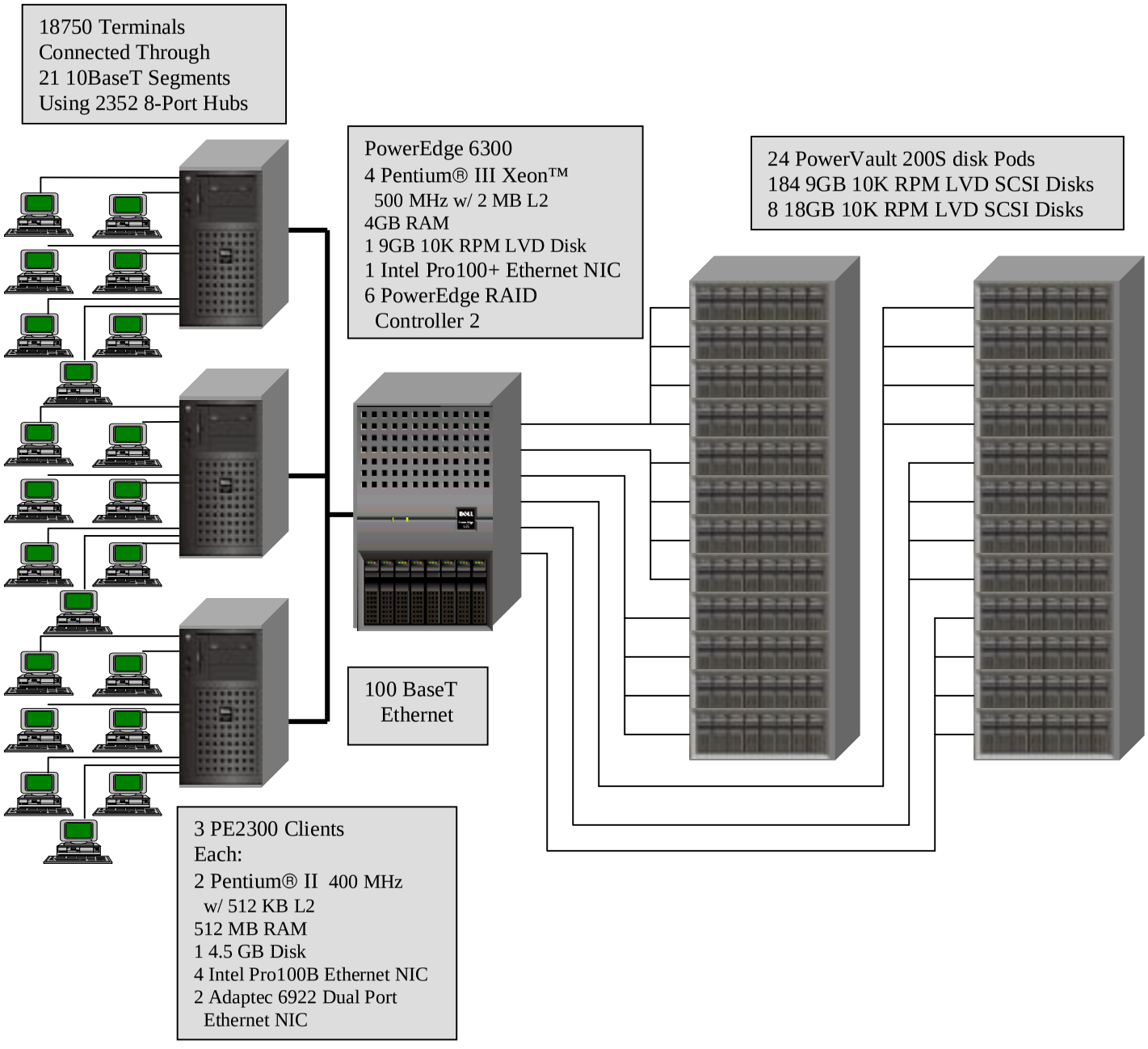


Figure 2: Priced Configuration



Clause 1 -- Logical Database Design Related Items

Table Definitions

Listings must be provided for all table definition statements and all other statements used to set-up the database. (8.1.2.1)

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

Physical Organization of the Database

The physical organization of tables and indices, within the database, must be disclosed. (8.1.2.2)

The measured configuration used 192 disk drives. The organization is shown in Table 5: Data Distribution and in Appendix B.

Insert and Delete Operations

It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C transaction mix. Furthermore, any restriction in the SUT database implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows. (8.1.2.3)

Insert and delete functionality was fully operational during the benchmark.

Horizontal and Vertical Partitioning

While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark (see Clause 1.6), any such partitioning must be disclosed. (8.1.2.4)

Partitioning was not used in this benchmark.

Replication

Replication of tables, if used, must be disclosed (see Clause 1.4.6). (8.1.2.5)

Replication was not used in this benchmark.

Table Attributes

Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance (see Clause 1.4.7). (8.1.2.6)

No additional attributes were used in this benchmark.

Clause 2 -- Transaction and Terminal Profiles Related Items

Random Number Generation

The method of verification for the random number generation must be described. (8.1.3.1)

Random numbers for transaction distributions, etc., were generated inside the RTE using a Lehmer random number generator which returns a pseudo-random real number uniformly distributed between 0.0 and 1.0. The method is described in "Random Number Generators: Good Ones Are Hard to Find", by Steve Park and Keith Miller in **Communications of the ACM,** October, 1988. Sample code is available from ftp://cs.wm.edu/pub/rngs.tar. During the RTE audit the auditor verified generated random numbers conformed to required distributions.

Screen Layout

The actual layouts of the terminal input/output screens must be disclosed. (8.1.3.2)

The screen layouts are based on 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. There are some very minor differences based on the fact that this is a web client implementation.

Terminal 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). (8.1.3.3)

The terminal features were verified by allowing the auditor to manually execute each of the five transaction types, using Microsoft Internet Explorer version 3.0.

Intelligent Terminals

Any usage of presentation managers or intelligent terminals must be explained. (8.1.3.4)
***Comment 1**: The intent of this clause is to describe any special manipulations performed by a local terminal or workstation to off-load work from the SUT. This includes, but is not limited to: screen presentations, message bundling, and local storage of TPC-C rows.*
***Comment 2**: This disclosure also requires that all data manipulation functions performed by the local terminal to provide navigational aids for transaction(s) must also be described. Within this disclosure, the purpose of such additional function(s) must be explained.*

Application code involved in the manipulation of data was run on the client. Screen manipulation commands in the form of HTML were downloaded to the web browser which handled input and output presentation graphics. A listing of this code is included in Appendix A. Microsoft Internet Information Service assisted in the processing and presentation of this data.

Transaction Profiles

The percentage of home and remote order-lines in the New-Order transactions must be disclosed. (8.1.3.5)
The percentage of New-Order transactions that were rolled back as a result of an unused item number must be disclosed. (8.1.3.6)
The number of items per orders entered by New-Order transactions must be disclosed. (8.1.3.7)
The percentage of home and remote Payment transactions must be disclosed. (8.1.3.8)
The percentage of Payment and Order-Status transactions that used non-primary key (C_LAST) access to

the database must be disclosed. (8.1.3.9)
The percentage of Delivery transactions that were skipped as a result of an insufficient number of rows in the NEW-ORDER table must be disclosed. (8.1.3.10)

Table 1: Transaction Statistics

Transaction	Function	Value
New Order	Home Warehouse Items	99.00%
	Remote Warehouse Items	1.00%
	Rolled Back Transactions	0.99%
	Average Lines Per Order	9.99
Payment	Home Warehouse	85.00%
	Remote Warehouse	15.00%
	Non-Primary Key Access	60.12%
Order Status	Non-Primary Key Access	59.83%
Delivery	Skipped Transactions	0

Transaction Mix

The mix (i.e., percentages) of transaction types seen by the SUT must be disclosed. (8.1.3.11)

Table 2: Transaction Mix

Transaction	Percentage
New Order	44.88%
Payment	43.05%
Order Status	4.02%
Delivery	4.02%
Stock Level	4.00%

Deferred Delivery Mechanism

The queuing mechanism used to defer the execution of the Delivery transaction must be disclosed. (8.1.3.12)

The client application submits delivery transactions to asynchronous Tuxedo queues running on the client machines. There were multiple delivery servers with single execution threads running on each client machine. These delivery servers were responsible for processing deliveries queued to Tuxedo and submitting them to the database server.

The source code is listed in Appendix A.

Clause 3 -- Transaction and System Properties Related Items

ACID Tests

The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosing which case was followed for the execution of Isolation Test 7. (8.1.4.1)

All ACID property tests were successful. The executions are described below.

Atomicity

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

Completed Transactions

A row was selected in a script from the warehouse, district and customer tables, and the balances noted. A payment transaction was started with the same warehouse, district and customer identifiers and a known amount. The payment transaction was committed and the rows were verified to contain correctly updated balances.

Aborted Transactions

A row was selected in a script from the warehouse, district and customer tables, and the balances noted. A payment transaction was started with the same warehouse, district and customer identifiers and a known amount. The payment transaction was rolled back and the rows were verified to contain the original balances.

Consistency

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

Consistency conditions one through four were tested using a shell script to issue queries to the database. The results of the queries verified that the database was consistent for all four tests. A run was executed under full load lasting over ten (10) minutes and included a checkpoint. The shell script was executed again. The result of the same queries verified that the database remained consistent after the run.

Isolation

Sufficient conditions must be enabled at either the system or application level to ensure the required isolation defined above (clause 3.4.1) is obtained.

Isolation tests one through seven were executed using shell scripts to issue queries to the database. Each script included timestamps to demonstrate the concurrency of operations. The results of the queries were captured to files. The captured files were verified by the auditor to demonstrate the required isolation had been met.

In addition, the phantom tests and the stock level tests were executed and verified.

For Isolation test seven, case A was followed.

Durability

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

Durable Media Failure

For convenience, durability from media failure was demonstrated on a 10 Warehouse database having similar characteristics to the fully scaled database. The standard driving mechanism was used to generate the transaction load of 100 users for the Loss of Data. The fully scaled database under full load would also have passed the following test.

Loss of Data

Loss of data was demonstrated on a 10 Warehouse database for convenience. The standard driving mechanism was used to generate the transaction load of 100 users for the test. To demonstrate recovery from a permanent failure of durable media containing TPC-C tables, the following steps were executed:

1. A 10 Warehouse database was built having similar characteristics to the large database.
2. The database was backed up using SQL Server backup facilities.
3. A sum of D_NEXT_O_ID was taken.
4. 100 users were logged in to the database and ran transactions.
5. One disk drive in the array was removed causing SQL Server to shut down.
6. SQL Server was restarted and a dump of the transaction log was taken.
7. The 10 Warehouse database was restored from backup.
8. The transaction log was restored and transactions rolled forward.
9. A new count of D_NEXT_O_ID was taken.
10. This number was compared with the number of new orders reported by the RTE.

Loss of Log

Loss of Log was demonstrated on a 10 Warehouse database for convenience. The standard driving mechanism was used to generate the transaction load of 100 users for the test. To demonstrate recovery from a permanent failure of durable medium containing SQL Server transaction log data, the following steps were executed:

1. A 10 Warehouse database was built having similar characteristics to the large database.
2. The database was backed up using SQL Server backup facilities.
3. A sum of D_NEXT_O_ID was taken.
4. 100 users were logged in to the database and ran transactions.
5. One disk drive holding the transaction log was removed causing an NT alert message, but causing no failures.
6. The RTE was shut down normally.
7. A new count of D_NEXT_O_ID was taken.
8. This number was compared with the number of new orders reported by the RTE.

Instantaneous Interruption and Loss of Memory

An instantaneous system interruption was caused by powering off the Server. This test was executed on a fully scaled database of 1900 warehouses under a full load of 18750 users. The following steps were executed:

1. A sum of D_NEXT_O_ID was taken.
2. 18750 users were logged in to the database and ran transactions for 5 minutes.

3. A checkpoint was initiated.
4. After completion of the checkpoint, the Server was powered off by normal means, causing instantaneous interruption.
5. The RTE was allowed to continue running. Completed transactions enroute from the clients were recorded. Error messages from the clients reporting loss of network to the server began appearing in the RTE log and screen.
6. The RTE was eventually shut down.
7. The server was powered on again and rebooted.
8. SQL Server was restarted and automatically recovered.
9. A new count of D_NEXT_O_ID was taken.
10. This number was compared with the number of new orders reported by the RTE.

Clause 4 -- Scaling and Database Population Related Items

Table Cardinality

The cardinality (e.g., the number of rows) of each table, as it existed at the start of the benchmark run (see Clause 4.2), must be disclosed. If the database was over-scaled and inactive rows of the WAREHOUSE table were deleted (see Clause 4.2.2), the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed. (8.1.5.1)

The database was originally built with 1900 warehouses. The performance run used 1875 warehouses and this is verified by runcheck

Table 3: Table Cardinality

Table	Cardinality as Benchmarked
Warehouse	1900
District	19000
Customer	57000000
History	57000000
NewOrder	17100000
Orders	57000000
OrderLine	570001928
Item	100000
Stock	190000000
Deleted Warehouses	0

Constant Values

The following values were used as constant value inputs to the NURand function for this benchmark.

Table 4: Constant Values

Function	Constant C Value
C_LAST (Build)	123
C_LAST (Run)	208

Data Distribution

The distribution of tables and logs across all media must be explicitly depicted for the tested and priced systems. (8.1.5.2)

The database was built using a total of 192 disks, 184 9GB for data and 8 18GB disk for transaction logs. The data drives were configured as hardware RAID 0. Logs and backups were hardware RAID 10. Controllers 0 thru 4 each have 3 containers with 32 disk stripes, with container 0 for Customer/Stock, container 1 for Miscellaneous, and container 2 for backups. Controller 5 has container 0 for the logs on 4 disk stripes on 4 mirrored disks , container 1 for Customer/Stock with 24 disk stripes, and container 2 for backup. The details are shown in Table 5.

Table 5: Data Distribution

Controller	Drives	Partition	Size	Use	RAID
Internal 0	9GB	C: & D:	1000/7470MB	OS & SQL	None
Dell PERC2 – 0	9GB	O:	19488MB	CS_6	0
	9GB	P:	16000MB	MISC_5	0
	9GB	Y:	106784MB	Backup_6	10
Dell PERC2 – 1	9GB	M:	19488MB	CS_5	0
	9GB	N:	16000MB	MISC_4	0
	9GB	X:	106784MB	Backup_5	10
Dell PERC2 – 2	9GB	K:	19488MB	CS_4	0
	9GB	L:	16000MB	MISC_3	0
	9GB	W:	106784MB	Backup_4	10
Dell PERC2 – 3	9GB	I:	19488MB	CS_3	0
	9GB	J:	16000MB	MISC_2	0
	9GB	V:	106784MB	Backup_3	10
Dell PERC2 – 4	9GB	G:	19488MB	CS_2	0
	9GB	H:	16000MB	MISC_1	0
	9GB	U:	106784MB	Backup_2	10
Dell PERC2 – 5	18GB	Q:	66400MB	Logs	10
	9GB	E:	19488MB	CS_1	0
	9GB	T:	106784MB	Backup_1	10

***Comment**: Detailed diagrams for layout of database files on disks can widely vary, and it is difficult to provide exact guideline suitable for all implementations. The intent is to provide sufficient detail to allow independent reconstruction of the test database. The two figures below are examples of database layout descriptions and are not intended to depict or imply any optimal layout for the TPC-C database.*
8.1.5.3 A statement must be provided that describes:

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

Microsoft SQL Server 7.0 is a relational DBMS.

The interface used was Microsoft SQL Server stored procedures accessed with Remote Procedure Calls embedded in C code using the Microsoft DBLIB interface.

Partition Mapping

The mapping of database partitions/replications must be explicitly described.
***Comment**: The intent is to provide sufficient detail about partitioning and replication to allow independent reconstruction of the test database. (8.1.5.4)*
An description of a database partitioning scheme is presented below as an example. The nomenclature of this example was outlined using the CUSTOMER table (in Clause 8.1.2.1), and has been extended to use the ORDER and ORDER_LINE tables as well.

The database was not replicated.

180 Day Space Calculation

Details of the 180 day space computations along with proof that the database is configured to sustain 8 hours of growth for the dynamic tables (Order, Order-Line, and History) must be disclosed (see Clause 4.2.3). (8.1.5.5)

1. To calculate the space required to sustain the database log for 8 hours of growth at steady state, the following steps were followed:
2. The free space on the log file was queried using *DBCC checktable(syslogs).*
3. Transactions were run against the database with a full load of users.
4. The free space was again queried using *DBCC checktable(syslogs).*
5. The space used was calculated as the difference between the first and second query.
6. The number of NEW-ORDERS was verified from an RTE report covering the entire run.
7. The space used was divided by the number of NEW-ORDERS giving a spaceused per NEW-ORDER transaction.
8. The space used per transaction was multiplied by the measured tpmC rate times 480 minutes.

The results of the above steps yielded a requirement of 56.78 GB (including mirror) to sustain the log for 8 hours. Space available on the transaction log volume was 66.4 GB (including mirror), indicating that enough storage was configured to sustain 8 hours of growth.

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

The details of the 180-day space requirement is shown in Appendix D.

Clause 5 -- Performance Metrics and Response Time Related Items

Measured TpmC

Measured tpmC must be reported. (8.1.6.1)

Measured TpmC	23,460.57
Price per TpmC	$17.26

Response Times

Ninetieth percentile, maximum and average response times must be reported for all transaction types as well as for the Menu response time. (8.1.6.2)

Table 6: Transaction Response Times

Transaction	Average	90%	Maximum
New Order	0.36	0.6	2.89
Payment	0.25	0.5	2.46
Order Status	0.28	0.5	2.92
Interactive Delivery	0.11	0.12	0.21
Deferred Delivery	0.56	0.97	2.59
Stock Level	1.63	2.3	3.13
Menu	0.11	0.12	0.74

Think Times & Key Times

The minimum, the average, and the maximum keying and think times must be reported for each transaction type. (8.1.6.3)

Table 7: Transaction Key Times

Transaction	Minimum	Average	Maximum
New Order	18.00	18.01	18.05
Payment	3.00	3.01	3.06
Order Status	2.00	2.00	2.04
Delivery	2.00	2.00	2.04
Stock Level	2.00	2.00	2.04

Table 8: Transaction Think Times

Transaction	Minimum	Average	Maximum
New Order	0.00	12.07	120.82
Payment	0.00	12.08	120.82
Order Status	0.00	10.10	100.81
Delivery	0.00	5.10	50.80
Stock Level	0.00	5.10	50.80

Response Time Distribution Curves

Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type. (8.1.6.4)

Figure 3: New Order Response Time Distribution

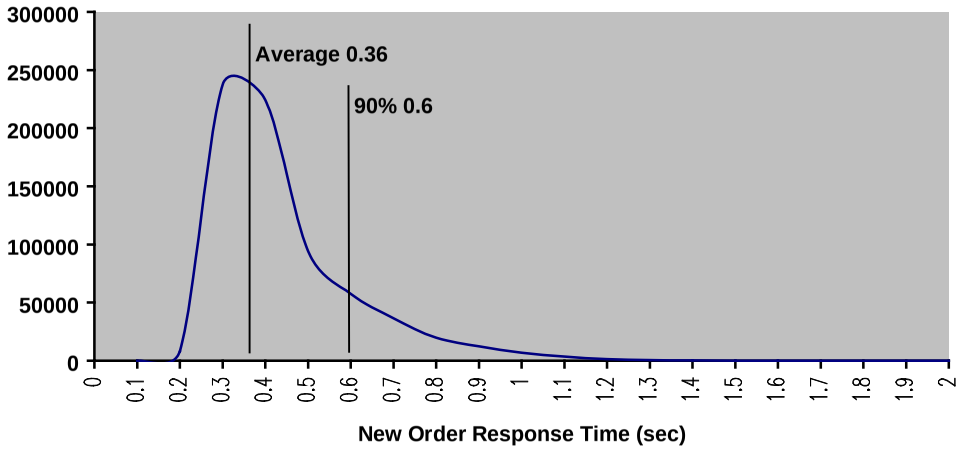


Figure 4: Payment Response Time Distribution

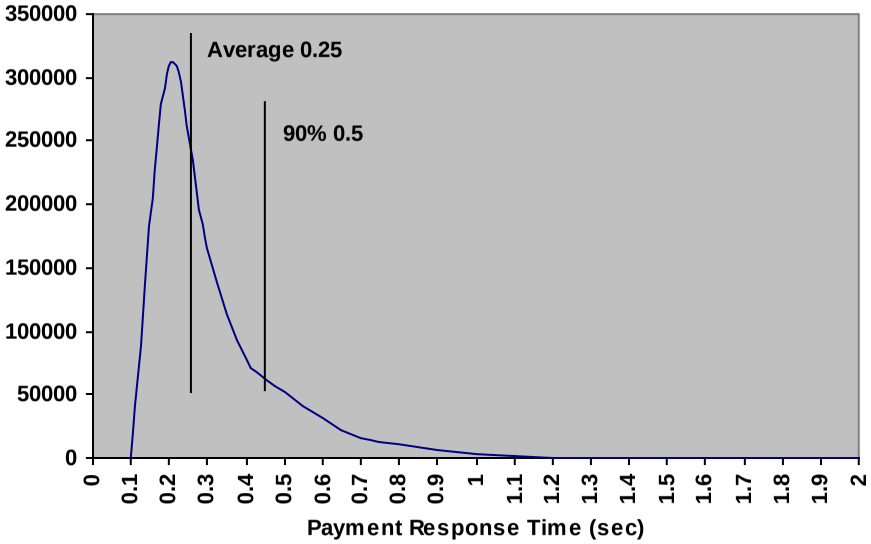


Figure 5: Order Status Response Time Distribution

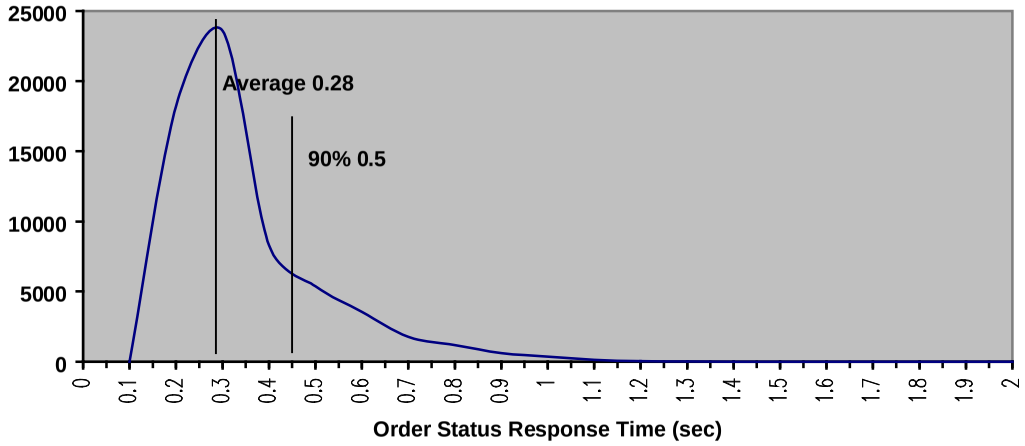


Figure 6: Delivery Response Time Distribution

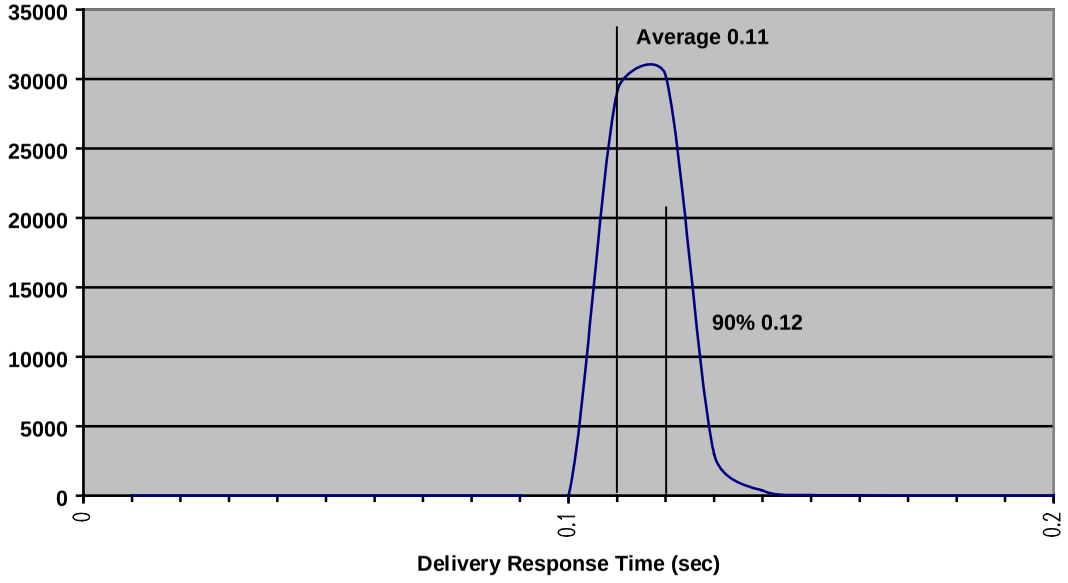
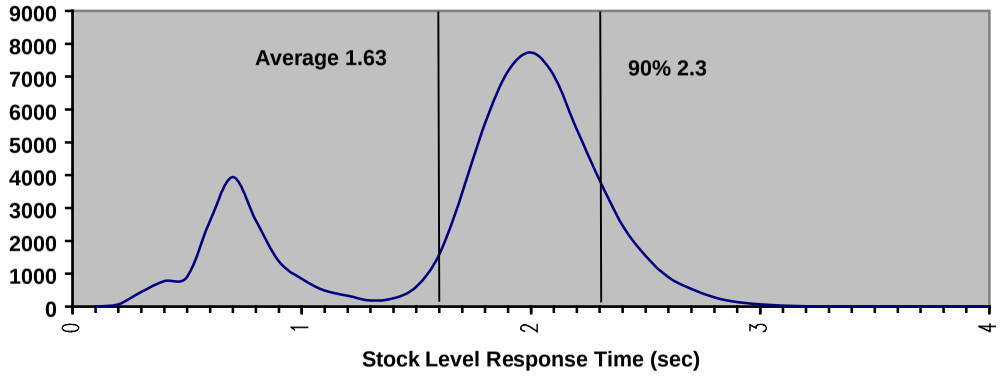


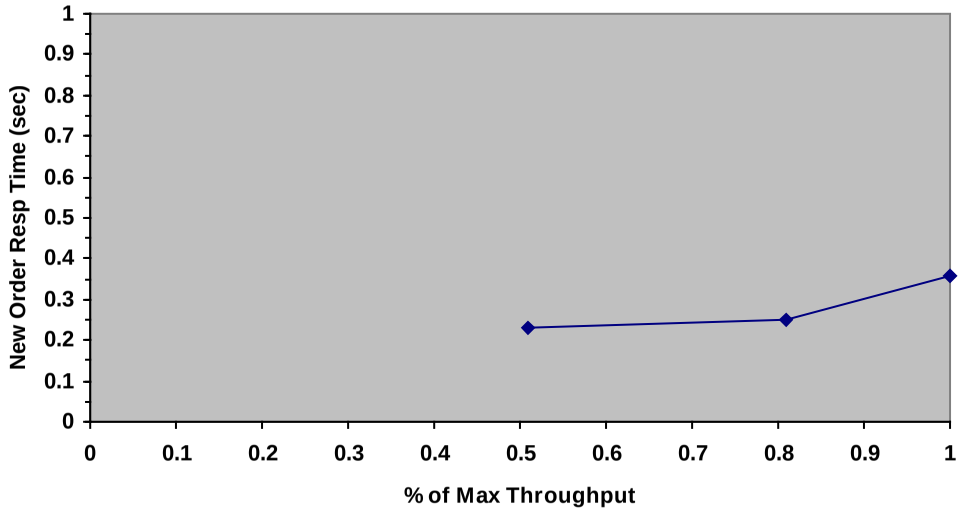
Figure 7: Stock Level Response Time Distribution



New-Order Response Time vs. Throughput Graph

The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction. (8.1.6.5)

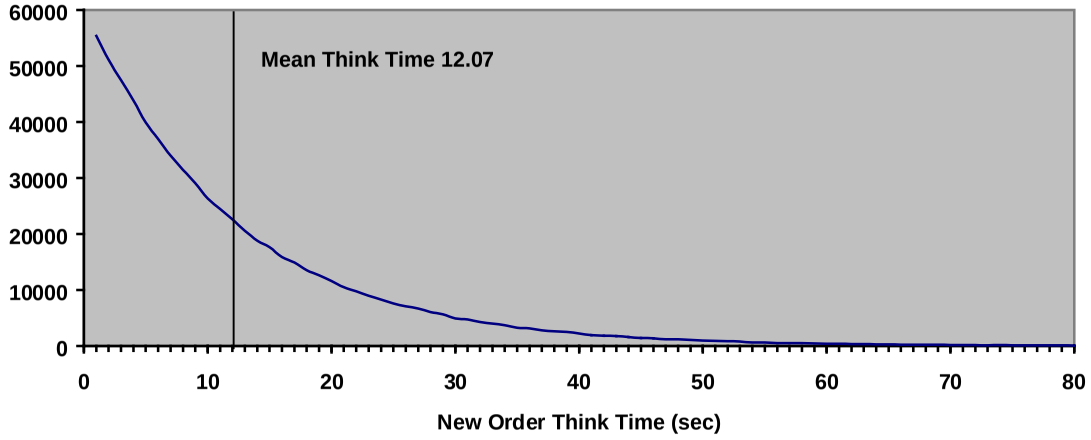
Figure 8: New Order Response Time vs. Throughput



New-Order Think Time Distribution Graph

Think Time frequency distribution curves (see Clause 5.6.3) must be reported for the New-Order transaction (8.1.6.6)

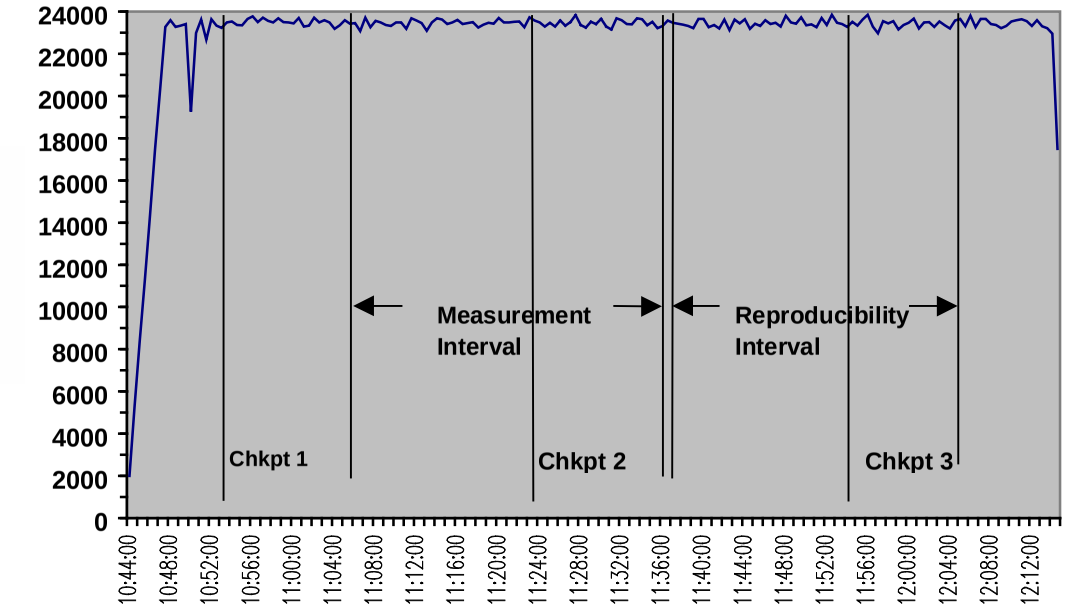
Figure 9: New Order Think Time Distribution



Steady-State Graph

A graph of throughput versus elapsed time (see Clause 5.6.5) must be reported for the New-Order transaction. (8.1.6.8)

Figure 10: New Order Throughput vs. Time



Steady-State Methodology

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval (see Clause 5.5) must be described. (8.1.6.9)

Steady state was determined using real time monitor utilities from both the operating system and the RTE. Steady state was further confirmed by the throughput data collected during the run and graphed in Figure 10.

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. (8.1.6.10)

The RTE generated the required input data to choose a transaction from the menu. This data was timestamped. The menu response for the requested transaction was verified and timestamped in the RTE log files.

The RTE generated the required input data for the chosen transaction. It waited to complete the minimum required key time before transmitting the HTTP request to the client. The transmission was timestamped. The return of the screen with the required response data was timestamped. The difference between these two timestamps was the response time for that transaction and was logged in the RTE log.

The RTE then waited the required think time interval before repeating the process starting at selecting another transaction from the menu.

The RTE transmissions were sent to the web-based application program running on the client machines through Ethernet LANs. These web clients managed the emulated web browser interface as well as all requests to the database on the server. The applications communicated with the database server over another Ethernet LAN using the Tuxedo transaction monitor and Microsoft SQL Server DBLIB library and RPC calls.

To perform checkpoints at specific intervals, we set SQL Server *recovery interval* to the maximum allowable value and wrote a script to schedule multiple checkpoints at specific intervals. By setting the TRACE FLAG #3502, SQL Server logged the checkpoint beginning and ending time in the ERRORLOG file. The script included a wait time between each checkpoint equal to the measurement interval which was 30 minutes. The checkpoint script was started manually after the RTE had all users logged in and sending transactions.

At each checkpoint, Microsoft SQL Server wrote to disk all memory pages that had been updated but not yet physically written to disk. Upon completion of the checkpoint, Microsoft SQL Server wrote a special record to the recovery log to indicate that all disk operations had been satisfied to this point. The positioning of the checkpoint was verified to be clear of the guard zones and is depicted on the graph in Figure 8.

Reproducibility Methodology

A description of the method used to determine the reproducibility of the measurement results must be reported. (8.1.6.11)

The RTE master program was invoked with the configuration file shown in Appendix C - Tunable Parameters and the number of users logged in and issuing transactions was ramped up from 0 to 18750 over 4.0 minutes. We allowed the database to warm up and to reach a steady state for an additional 17.3 minutes, including a checkpoint. The steady state was sustained for a 30-minute measurement interval, including checkpoint. One second after the conclusion of the reproducibility interval the reproducibility interval began. After 30 minutes of steady state, including a checkpoint, the command was issued to the RTE to stop issuing transactions and the users were logged out of the web server. The repeatable interval result was within 0.13% of the reported interval result.

Measurement Interval

A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC) must be included. (8.1.6.12)

The measurement interval was 30 minutes.

Transaction Mix

8.1.6.13 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. (8.1.6.13)

The RTE was given a weighted random distribution which was not be adjusted during the run.

The percentage of the total mix for each transaction type must be disclosed. (8.1.6.14)

Table 9: Transaction Mix

Transaction	Percentage
New Order	44.88%
Payment	43.05%
Order Status	4.02%
Delivery	4.02%
Stock Level	4.00%

Other Metrics

The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed. (8.1.6.15)
The average number of order-lines entered per New-Order transaction must be disclosed. (8.1.6.16)
The percentage of remote order-lines entered per New-Order transaction must be disclosed. (8.1.6.17)
The percentage of remote Payment transactions must be disclosed. (8.1.6.18)
The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed. (8.1.6.19)
The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed. (8.1.6.20)

Table 10: Transaction Statistics

Transaction	Function	Value
New Order	Home Warehouse Items	99.00%
	Remote Warehouse Items	1.00%
	Rolled Back Transactions	0.99%
	Average Lines Per Order	9.99
Payment	Home Warehouse	85.00%
	Remote Warehouse	15.00%
	Non-Primary Key Access	60.12%
Order Status	Non-Primary Key Access	59.83%
Delivery	Skipped Transactions	0

Checkpoints

The number of checkpoints in the Measurement Interval, the time in seconds from the start of the Measurement Interval to the first checkpoint and the Checkpoint Interval must be disclosed. (8.1.6.21)

There was 1 checkpoint in the measurement interval. It started 1091 sec after the start of the measurement interval. The checkpoint in the measurement interval lasted 224 seconds.

Clause 6 -- SUT, Driver, and Communication Definition Related Items

RTE Parameters

The RTE input parameters, code fragments, functions, etc. used to generate each transaction input field must be disclosed. (8.1.7.1)
Comment: *The intent is to demonstrate the RTE was configured to generate transaction input data as specified in Clause 2.*

The RTE input parameters are listed in Appendix C - Tunable Parameters.

Emulated Components

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system. The results of the test described in Clause 6.6.3.4 must be disclosed. (8.1.7.2)

No components were emulated.

Benchmarked and Targeted System Configuration Diagrams

A complete functional diagram of both the benchmark configuration and the configuration of the proposed (target) system must be disclosed. A detailed list of all software and hardware functionality being performed on the Driver System, and its interface to the SUT must be disclosed (see Clause 6.6.3.6). (8.1.7.3)

The driver system performed transaction data generation and communication to the client through the standard web browser (HTTP) protocol. It also captured and timestamped the SUT output data for post-processing of the reported metrics. No other functionality was included on the driver system.

Figures 1 & 2 of this report contain detailed diagrams of both the benchmark configuration and the priced configuration.

Network Configuration

The network configurations of both the tested services and the proposed (target) services which are being represented and a thorough explanation of exactly which parts of the proposed configuration are being replaced with the Driver System must be disclosed (see Clause 6.6.4). (8.1.7.4)

The network configurations of the benchmarked and priced configurations were identical with the exception that, to meet the requirement of maximum 1024 emulated user per network segment, two of the Intel ethernet adapters on each client as measured are replaced with Adaptec 6922 Dual Port Ethernet adapters to give a total of seven segments to handle the 6250 emulated users per client.

Network Bandwidth

The bandwidth of the network(s) used in the tested/priced configuration must be disclosed. (8.1.7.5)

The bandwidth of the tested and priced networks were as follows:

- 10 BaseT (10 Mbit/sec) network segments between the RTE/Emulated Users and the Clients.
- 100 BaseT (100 Mbit/sec) between the Clients and Server.

Operator Intervention

If the configuration requires operator intervention (see Clause 6.6.6), the mechanism and the frequency of this intervention must be disclosed. (8.1.7.6)

This configuration does not require any operator intervention to sustain eight hours of the reported throughput.

Clause 7 -- Pricing Related Items

Hardware and Software List

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 date. If package-pricing is used, vendor part number of the package and a description uniquely identifying each of the components of the package must be disclosed.

Pricing source(s) and effective date(s) of price(s) must also be reported. (8.1.8.1)

The total 5-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed. (8.1.8.2)

The details of the hardware and software are reported in the front of this report as part of the executive summary. All third party quotations are included at the end of this report as Appendix E.

Availability Date

The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available. (8.1.8.3)

Hardware Availability Date: June 1, 1999
Software Availability Date: June 1, 1999

Measured TpmC

A statement of the measured tpmC, as well as the respective calculations for 5-year pricing, price/performance (price/tpmC), and the availability date must be included. (8.1.8.4)

Maximum Qualified Throughput: 23,460.57 tpmC
Price Performance Metric: $17.26/tpmC

Country Specific Pricing

Additional Clause 7 related items may be included in the Full Disclosure Report for each country specific priced configuration. Country specific pricing is subject to Clause 7.1.7. (8.1.8.5)

This system is priced for the United States of America.

Usage Pricing

For any usage pricing, the sponsor must disclose (8.1.8.6):

- *Usage level at which the component was priced.*
- *A statement of the company policy allowing such pricing.*

***Comment**: Usage pricing may include, but is not limited to, the operating system and database management software.*

The component pricing based on usage is shown below:

- 3 Microsoft Windows NT Server 4.0 Licenses
- 1 Microsoft Windows NT Server 4.0 Enterprise Edition License
- 1 Microsoft SQL Server 7.0, Enterprise Edition, License.
- 1 Microsoft Visual C++ 32 bit Edition
- 5 Year Support for Hardware Components.

System Pricing

System pricing should include subtotals for the following components: Server Hardware, Server Software, Client Hardware, Client Software, and Network Components used for terminal connection (see Clause 7.2.2.3). Clause 6.1 describes the Server and Client components. An example of the standard pricing sheet is shown in Appendix B. (8.1.8.7)
System pricing must include line item indication where non-sponsoring companies' brands are used. System pricing must also include line item indication of third party pricing. See example in Appendix B. (8.1.8.8)

The details of the hardware and software are reported in the front of this report as part of the executive summary. All third party quotations are included at the end of this report as Appendix E.

Clause 9 -- Audit Related Items

Auditor

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. (8.1.9.1)
A review of the pricing model is required to ensure that all components required are priced (see Clause 9.2.8). The auditor is not required to review the final Full Disclosure Report or the final pricing prior to issuing the attestations letter. (8.1.9.2)

This TPC-C benchmark has been audited by Tom Sawyer of Performance Metrics.

Availability of the Full Disclosure Report

The Full Disclosure Report must be readily available to the public at a reasonable charge, similar to the charges for similar documents by the test sponsor. The report must be made available when results are made public. In order to use the phrase "TPC Benchmark™ C", the Full Disclosure Report must have been submitted to the TPC Administrator as well as written permission obtained to distribute same.

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

Transaction Processing Performance Council
c/o Shanley Public Relations
777 North First Street, Suite 6000
San Jose, CA 95112-6311
www.tpc.org

or:

Dell Computer Corporation
1 Dell Drive
Round Rock, TX 78682
Attention: Dave Jaffe

Auditor's Letter of Attestation

March 26, 1999

Tony Yaptangco
Manager, System Performance
Dell Computer Corporation
One Dell Way
Round Rock, TX 78682

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

Platform:	Dell PowerEdge 6300 Server
Database Manager:	Microsoft SQL Server Enterprise Edition 7.0
Operating System:	Microsoft Windows NT Server Enterprise Edition 4.0 (SP4)
Transaction Manager:	BEA TUXEDO CFS 6.3 for NT

Server: Dell PowerEdge 6300 Server				
CPU's	Memory	Disks	90% Response	tpmC
4 Pentium III Xeon @ 500 MHz	Main: 4 GB Cache: 2MB	184 @ 9 GB 8 @ 18 GB	**0.60 sec**	**23,460.57**
3 Clients: Dell PowerEdge 2300 (each)				
2 Pentium II @ 400 MHz	Main: 512 MB Cache: 512 KB	1 @ 4 GB	Na	na

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

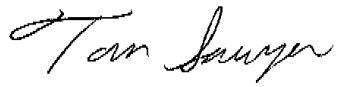
- The transactions were correctly implemented.
- The database files were properly sized and populated.

- The database was properly scaled with 1,900 warehouses. The measurement used 1,875 warehouses; I verified that d_next_o_it and w_ytd did not change for the unused warehouses
- The ACID properties were met.
- The Durability tests for loss-of-log and data-loss were performed on a 10-warehouse system.
- Input data was generated according to the specified percentages.
- Eight hours of mirrored log space was present on the tested system.
- Eight hours of growth space for the dynamic tables was present on the tested system.
- The data for the 180 day space calculation was verified.
- The steady state portion of the test was 30 minutes.
- One checkpoint was taken before the measured interval.
- One checkpoint was taken during the measured interval.
- The checkpoints were verified to be clear of the guard zone.
- The system pricing was checked for major components and maintenance.

Auditor Notes:

None.

Sincerely,



Tom Sawyer
Auditor

Appendix A – Application Source Code

tpcc.dll ISAPI DLL Source Code

tpcc.def

```
LIBRARY      tpcc

DESCRIPTION  'Dell TPC-C Client / Web Server'

EXPORTS
    GetExtensionVersion
    HttpExtensionProc
```

Appendix A – Application Source Code

tpcc.h

```
//----------------------tpcc.h: Dell TPC-C Client / Web Server-------------------------
//
// Copyright (c) 1998 Dell Computer Corporation, All Rights Reserved
//
// Author: Dave Jaffe               Last modified: 11/4/97
//
// Audited: Richard Gimarc  Performance Metrics Inc.  12/31/97
//
// header file for tpcc.dll MS ISAPI DLL for TPC-C Benchmark
//
#ifndef _TPCC_H_
#define _TPCC_H_

#include <windows.h>
#include <stdio.h>
#include <process.h>
#include <httpext.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <math.h>
#include <time.h>
//  tuxedo header
#include <tmenv.h>
#include <xa.h>
#include <atmi.h>

#include "html.h"
#include "trans.h"      // Transaction data structures

// Commands
enum _command
    {
    Login,
    NewOrderFormRequest,
    NewOrder,
    PaymentFormRequest,
    Payment,
    OrderStatusFormRequest,
    OrderStatus,
    DeliveryFormRequest,
    Delivery,
    StockLevelFormRequest,
    StockLevel,
    Logout
    };

char* command_name[] =
    {
    "Login",
    "NewOrderFormRequest",
    "NewOrder",
    "PaymentFormRequest",
    "Payment",
    "OrderStatusFormRequest",
    "OrderStatus",
    "DeliveryFormRequest",
    "Delivery",
    "StockLevelFormRequest",
    "StockLevel",
    "Logout"
    };


// Functions
BOOL ParseInput(char* pCMD, enum _command* Command, int* User, char** pInputs);
void ProcessLogin(char* cmd_input_str, char* html, BOOL* KeepConn);
int  ParseLogin(char* cmd_input_str, short* w_id, short* d_id);
void ProcessNewOrder(char* cmd_input_str, int User, char* html);
int  ParseNewOrder(char* cmd_input_str, NEW_ORDER_DATA* pNewOrderData);
void ProcessPayment(char* cmd_input_str, int User, char* html);
int  ParsePayment(char* cmd_input_str, PAYMENT_DATA* pPaymentData);
void ProcessOrderStatus(char* cmd_input_str, int User, char* html);
int  ParseOrderStatus(char* cmd_input_str, ORDER_STATUS_DATA* pOrderStatusData);
void ProcessDelivery(char* cmd_input_str, int User, char* html);
int  ParseDelivery(char* cmd_input_str, DELIVERY_DATA* pDeliveryData);
void ProcessStockLevel(char* cmd_input_str, int User, char* html);
int  ParseStockLevel(char* cmd_input_str, STOCK_LEVEL_DATA* pStockLevelData);
void ProcessLogout(int User);
void ReadTPCCRegParams();
BOOL OpenDeliveryLog();
void WriteErrorLog(char* message);
BOOL OpenErrorLog();
BOOL IsValidNonNegShort(char* str, short* number);
BOOL IsValidNonNegLong(char* str, long* number);
BOOL IsValidNonNegDouble(char* str, double* number);

// Transaction Monitor client functions
BOOL TMClientInit();
void TMClientExit();
int  TMNewOrder(NEW_ORDER_DATA* pNewOrder);
int  TMPayment(PAYMENT_DATA* pPayment);
int  TMOrderStatus(ORDER_STATUS_DATA* pOrderStatus);
int  TMDelivery(DELIVERY_DATA *pDel);
int  TMStockLevel(STOCK_LEVEL_DATA* pStockLevel);

//  tuxedo functions
static DWORD tls_idx;               // thread local storage index
static int IsTpInit();              // use TLS to determine if tpinit needs to be run
static void TPPrintError(char *msg);    // writes tperrno to logfile

// Critical sections
CRITICAL_SECTION login_crit_sec;
CRITICAL_SECTION TLS_crit_sec;

// MAX_USERS_ALL_CLIENTS is maximum number of users between all clients
#define MAX_USERS_ALL_CLIENTS 5120

// Variables read from registry key Software\TPCC on LocalMachine:

//    Client specific:
int max_users_this_client;  // Maximum number of users client can handle
                            //  (REG_DWORD: MaxUsersThisClient)
//    Same on all clients:
int n_warehouses_total;     // Number of warehouses in database
                            //   (REG_DWORD: NumberOfWarehouses)
char dll_path[250];         // HTTP path of tpcc.dll
                            //   (REG_SZ: DLLPath)
char log_path[250];         // Path for delivery and other logs
                            //   (REG_EXPAND: LogPath)


// Global variables
long NO_len_ot = sizeof(NEW_ORDER_DATA);
long PY_len_ot = sizeof(PAYMENT_DATA);
long OS_len_ot = sizeof(ORDER_STATUS_DATA);
long DL_len_ot = sizeof(DELIVERY_DATA);
long SL_len_ot = sizeof(STOCK_LEVEL_DATA);

// Return codes from TM stored procedure calls
#define SQL_ERROR       -1  // Usually incorrect C_ID or C_LAST
#define SUCCESS         0   // Success
#define DEADLOCK        1   // Still deadlocked after specified DEADLOCK_RETRYs
#define INVALID_ITEM    2   // Invalid item in NewOrder

#define DEADLOCK_WAIT   10
#define DEADLOCK_RETRY  5
```

```
typedef struct
    {
    short w_id;
    short d_id;
    BYTE TransData[2048];
    char ButtonBar[2048];
    } UserData;

UserData* pUserData[MAX_USERS_ALL_CLIENTS];

FILE *fp_delivlog, *fp_errorlog;

LARGE_INTEGER freq, tick_count0;
double freqd;

// Error codes and associated error text from browser input parse routines
#define ERR_GENERIC             1
#define ERR_BLANK_WID           2
#define ERR_NONNUM_WID          3
#define ERR_BLANK_DID           4
#define ERR_NONNUM_DID          5
#define ERR_BLANK_CID           6
#define ERR_NONNUM_CID          7
#define ERR_BLANK_CWID          8
#define ERR_NONNUM_CWID         9
#define ERR_BLANK_CDID          10
#define ERR_NONNUM_CDID         11
#define ERR_BLANK_AMOUNT        12
#define ERR_NONNUM_AMOUNT       13
#define ERR_BLANK_CARRIER       14
#define ERR_NONNUM_CARRIER      15
#define ERR_BLANK_THRESHOLD     16
#define ERR_NONNUM_THRESHOLD    17
#define ERR_NONNUM_OL_S_WID     18
#define ERR_NONNUM_OL_I_ID      19
#define ERR_NONNUM_OL_QUAN      20
#define ERR_INCOMPLETE_OL       21
#define ERR_CUST_NAME_ID        22
#define ERR_AMT_TOO_LARGE       23

char* err_text[] =
    {
    "Success",
    "Generic error",
    "Blank warehouse id field",
    "Non-numeric or negative input in warehouse id field",
    "Blank district id field",
    "Non-numeric or negative input in district id field",
    "Blank customer id field",
    "Non-numeric or negative input in customer id field",
    "Blank customer warehouse id field",
    "Non-numeric or negative input in customer warehouse id field",
    "Blank customer district id field",
    "Non-numeric or negative input in customer district id field",
    "Blank amount field",
    "Non-numeric or negative input in amount field",
    "Blank carrier field",
    "Non-numeric or negative input in carrier field",
    "Blank threshold field",
    "Non-numeric or negative input in threshold field",
    "Non-numeric or negative input in supplying warehouse id field",
    "Non-numeric or negative input in item id field",
    "Non-numeric or negative input in quantity field",
    "Incomplete order line",
    "Either customer name or id must be entered",
    "Amount must be 9999.99 or less"
    };


#include "tmclient.c"           // transaction monitor client code

#endif // _TPCC_H_ not defined
//
//--------------------------------End of tpcc.h---------------------------------------
```

Appendix A – Application Source Code

trans.h

```
//--------------------trans.h: Dell TPC-C Client / Web Server-------------------------
//
// Copyright (c) 1997 Dell Computer Corporation, All Rights Reserved
//
// Author: Dave Jaffe               Last modified: 2/2/99
//
// Audited: Richard Gimarc/Tom Sawyer  Performance Metrics Inc.
//
// transaction header file for tpcc.dll MS ISAPI DLL for TPC-C Benchmark

#ifndef _TRANS_H_
#define _TRANS_H_

// String length constants
#define ITEM_NAME_LEN               24
#define ADDRESS_LEN                 20
#define STATE_LEN                   2
#define ZIP_LEN                     9
#define FIRST_NAME_LEN              16
#define MIDDLE_NAME_LEN             2
#define LAST_NAME_LEN               16
#define PHONE_LEN                   16
#define CREDIT_LEN                  2
#define CUST_DATA_LEN               200
#define MAX_OL_NEW_ORDER_ITEMS      15
#define MAX_OL_ORDER_STATUS_ITEMS   15
#define DATETIME_LEN                19
#define DATE_LEN                    10

// transaction structures

typedef struct
    {
    short               ol_supply_w_id;
    short               ol_quantity;
    short               ol_stock;
    long                ol_i_id;
    double              ol_i_price;
    double              ol_amount;
    char                ol_i_name[ITEM_NAME_LEN+1];
    char                ol_brand_generic;
    } OL_NEW_ORDER_DATA;

typedef struct
    {
    short               ol_supply_w_id;
    short               ol_quantity;
    long                ol_i_id;
    double              ol_amount;
    char                ol_delivery_d[DATE_LEN+1];
    } OL_ORDER_STATUS_DATA;

typedef struct
    {
    short               w_id;
    short               d_id;
    short               o_ol_cnt;
    short               o_commit_flag;
    short               o_all_local;
    short               num_deadlocks;
    long                c_id;
    long                o_id;
    double              c_discount;
    double              w_tax;
    double              d_tax;
    double              total_amount;
    char                c_last[LAST_NAME_LEN+1];
    char                c_credit[CREDIT_LEN+1];
    char                o_entry_d[DATETIME_LEN+1];
    OL_NEW_ORDER_DATA   Ol[MAX_OL_NEW_ORDER_ITEMS];
    } NEW_ORDER_DATA;

typedef struct
    {
    short               w_id;
    short               d_id;
    short               c_d_id;
    short               c_w_id;
    short               num_deadlocks;
    long                c_id;
    double              h_amount;
    double              c_credit_lim;
    double              c_discount;
    double              c_balance;
    char                w_street_1[ADDRESS_LEN+1];
    char                w_street_2[ADDRESS_LEN+1];
    char                w_city[ADDRESS_LEN+1];
    char                w_state[STATE_LEN+1];
    char                w_zip[ZIP_LEN+1];
    char                d_street_1[ADDRESS_LEN+1];
    char                d_street_2[ADDRESS_LEN+1];
    char                d_city[ADDRESS_LEN+1];
    char                d_state[STATE_LEN+1];
    char                d_zip[ZIP_LEN+1];
    char                c_first[FIRST_NAME_LEN+1];
    char                c_middle[MIDDLE_NAME_LEN + 1];
    char                c_last[LAST_NAME_LEN+1];
    char                c_street_1[ADDRESS_LEN+1];
    char                c_street_2[ADDRESS_LEN+1];
    char                c_city[ADDRESS_LEN+1];
    char                c_state[STATE_LEN+1];
    char                c_zip[ZIP_LEN+1];
    char                c_phone[PHONE_LEN+1];
    char                c_credit[CREDIT_LEN+1];
    char                c_data[CUST_DATA_LEN+1];
    char                h_date[DATETIME_LEN+1];
    char                c_since[DATE_LEN+1];
    } PAYMENT_DATA;

typedef struct
    {
    short               w_id;
    short               d_id;
    short               o_carrier_id;
    short               o_ol_cnt;
    short               num_deadlocks;
    long                c_id;
    long                o_id;
    double              c_balance;
    char                c_first[FIRST_NAME_LEN+1];
    char                c_middle[MIDDLE_NAME_LEN+1];
    char                c_last[LAST_NAME_LEN+1];
    char                o_entry_d[DATETIME_LEN+1];
    OL_ORDER_STATUS_DATA    OlOrderStatusData[MAX_OL_ORDER_STATUS_ITEMS];
    } ORDER_STATUS_DATA;

typedef struct
    {
    short               w_id;
    short               o_carrier_id;
    LONGLONG            queued_time;
    long                o_id[10];
    } DELIVERY_DATA;

typedef struct
    {
```

```
    short               w_id;
    short               d_id;
    short               threshold;
    short               num_deadlocks;
    long                low_stock;
    } STOCK_LEVEL_DATA;

#endif // _TRANS_H_ not defined
//
//-------------------------------End of trans.h-------------------------------------
```

Appendix A – Application Source Code

html.h

```
//--------------------html.h: Dell TPC-C Client / Web Server-------------------------
//
// Copyright (c) 1998 Dell Computer Corporation, All Rights Reserved
//
// Author: Dave Jaffe               Last modified: 9/4/97
//
// Audited: Richard Gimarc  Performance Metrics Inc. 9/24/97
//
// html header file for tpcc.dll MS ISAPI DLL for TPC-C Benchmark

char* ButtonBar =
    "<CENTER><TABLE><TR>\n"
    "<TD>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=NewOrderFormRequest>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<INPUT TYPE=SUBMIT VALUE=\"(1) New Order\">\n"
    "</FORM>\n"
    "<TD>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=PaymentFormRequest>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<INPUT TYPE=SUBMIT VALUE=\"(2) Payment\">\n"
    "</FORM>\n"
    "<TD>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=OrderStatusFormRequest>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<INPUT TYPE=SUBMIT VALUE=\"(3) Order Status\">\n"
    "</FORM>\n"
    "<TD>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=DeliveryFormRequest>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<INPUT TYPE=SUBMIT VALUE=\"(4) Delivery\">\n"
    "</FORM>\n"
    "<TD>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=StockLevelFormRequest>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<INPUT TYPE=SUBMIT VALUE=\"(5) Stock\">\n"
    "</FORM>\n"
    "<TD>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=Logout>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<INPUT TYPE=SUBMIT VALUE=\"(9) Exit\">\n"
    "</FORM>\n"
    "</TR></TABLE></CENTER>\r\n";

char* ResponseHTML =
    "<HTML>\n"
    "<HEAD><TITLE>%s</TITLE></HEAD>\n"
    "<BODY>\n"
    "%s"
    "<PRE>\n"
    "%s"
    "</PRE>\n"
    "</BODY>\n"
    "</HTML>\n";

char* NewOrderForm =
    "<HTML>\n"
    "<HEAD><TITLE>New Order Form</TITLE></HEAD>\n"
    "<BODY>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=NewOrder>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<PRE>\n"
    "                                   New Order\n"
    "Warehouse: %04d"
    "   District:  <INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=D_ID>"
    "                         Date:\n"
    "Customer:  <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=C_ID>"
    "   Name:                     Credit:      %%Disc:\n"
    "Order Number:           Number of Lines:           W_tax:         D_tax:\n\n"
    " Supp_W  Item_Id  Item Name                 Qty  Stock  B/G  Price    Amount\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W00>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I00>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q00>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W01>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I01>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q01>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W02>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I02>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q02>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W03>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I03>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q03>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W04>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I04>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q04>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W05>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I05>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q05>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W06>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I06>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q06>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W07>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I07>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q07>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W08>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I08>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q08>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W09>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I09>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q09>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W10>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I10>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q10>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W11>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I11>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q11>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W12>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I12>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q12>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W13>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I13>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q13>\n"
    " <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W14>   "
    "<INPUT TYPE=TEXT SIZE=6 MAXLENGTH=6 NAME=I14>                                "
    "<INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=Q14>\n"
    "</PRE>\n"
    "<INPUT TYPE=SUBMIT VALUE=Submit>\n"
    "<INPUT TYPE=RESET VALUE=Reset>\n"
    "</FORM>\n"
    "</BODY>\n"
    "</HTML>\n";

char* PaymentForm =
    "<HTML>\n"
    "<HEAD><TITLE>Payment Form</TITLE></HEAD>\n"
    "<BODY>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=Payment>\n"
```

```
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<PRE>\n"
    "                                    Payment\n"
    "Warehouse: %04d                    "
    "District: <INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=D_ID>\n\n\n\n\n"
    "Customer: <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=C_ID>"
    "  Cust-Warehouse: <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=C_W_ID>"
    "  Cust-District: <INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=C_D_ID>\n"
    "Name:   <INPUT TYPE=TEXT SIZE=17 MAXLENGTH=16 NAME=C_LAST>"
    "                           Since:\n"
    "                                                 Credit:\n"
    "                                                 %%Disc:\n"
    "                                                 Phone:\n"
    "Amount Paid:           $<INPUT TYPE=TEXT SIZE=7 MAXLENGTH=7 NAME=AMOUNT>"
    "      New Cust-Balance:\n"
    "Credit Limit:\n"
    "Cust-Data:\n"
    "</PRE>\n"
    "<INPUT TYPE=SUBMIT VALUE=Submit>\n"
    "<INPUT TYPE=RESET VALUE=Reset>\n"
    "</FORM>\n"
    "</BODY>\n"
    "</HTML>\n";

char* OrderStatusForm =
    "<HTML>\n"
    "<HEAD><TITLE>Order Status Form</TITLE></HEAD>\n"
    "<BODY>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=OrderStatus>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<PRE>\n"
    "                                  Order-Status\n"
    "Warehouse: %04d"
    "   District:  <INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=D_ID>\n"
    "Customer:  <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=C_ID>   "
    "Name:  <INPUT TYPE=TEXT SIZE=17 MAXLENGTH=16 NAME=C_LAST>\n"
    "Cust-Balance:\n\n"
    "Order-Number:            Entry-Date:                    Carrier-Number:\n"
    "Supply-W     Item-Id    Qty     Amount      Delivery-Date\n"
    "</PRE>"
    "<INPUT TYPE=SUBMIT VALUE=Submit>\n"
    "<INPUT TYPE=RESET VALUE=Reset>\n"
    "</FORM>\n"
    "</BODY>\n"
    "</HTML>\n";

char* DeliveryForm =
    "<HTML>\n"
    "<HEAD><TITLE>Delivery Form</TITLE></HEAD>\n"
    "<BODY>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=Delivery>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<PRE>\n"
    "                                  Delivery\n"
    "Warehouse: %04d\n\n"
    "Carrier Number:  <INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=CARRIER>\n\n"
    "Execution Status:\n"
    "</PRE>"
    "<INPUT TYPE=SUBMIT VALUE=Submit>\n"
    "<INPUT TYPE=RESET VALUE=Reset>\n"
    "</FORM>\n"
    "</BODY>\n"
    "</HTML>\n";

char* StockLevelForm =
    "<HTML>\n"
    "<HEAD><TITLE>Stock Level Form</TITLE></HEAD>\n"
    "<BODY>\n"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=StockLevel>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=%d>\n"
    "<PRE>\n"
    "                                  Stock-Level\n"
    "Warehouse: %04d   District:  %02d\n\n"
    "Stock Level Threshold:  <INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=THRESHOLD>\n\n"
    "low stock:\n"
    "</PRE>\n"
    "<INPUT TYPE=SUBMIT VALUE=Submit>\n"
    "<INPUT TYPE=RESET VALUE=Reset>\n"
    "</FORM>\n"
    "</BODY>\n"
    "</HTML>\n";

char* LoginForm =
    "<HTML>\n"
    "<HEAD><TITLE>Dell TPC-C Client Login</TITLE></HEAD>\n"
    "<BODY>\n"
    "<H1>Dell TPC-C Client Login</H1>"
    "<FORM METHOD=GET ACTION=\"%s\">\n"
    "<INPUT TYPE=HIDDEN NAME=CMD VALUE=Login>\n"
    "<INPUT TYPE=HIDDEN NAME=USER VALUE=-1>\n"
    "Warehouse: <INPUT TYPE=TEXT SIZE=4 MAXLENGTH=4 NAME=W_ID><BR>\n"
    "District:  <INPUT TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME=D_ID><BR>\n"
    "<INPUT TYPE=SUBMIT VALUE=Submit>\n"
    "<INPUT TYPE=RESET VALUE=Reset>\n"
    "</FORM>\n"
    "</BODY>\n"
    "</HTML>\n";
//
//--------------------------------End of html.h---------------------------------------
```

Appendix A – Application Source Code

tpcc.c

```
//-------------------tpcc.c: Dell TPC-C Client / Web Server--------------------------
//
// Copyright (c) 1998 Dell Computer Corporation, All Rights Reserved
//
// Author: Dave Jaffe/James Jordan          Last modified: 3/26/99
//
// Audited: Richard Gimarc/Tom Sawyer  Performance Metrics Inc.
//
// source code for tpcc.dll MS ISAPI DLL for TPC-C Benchmark
//
#include "tpcc.h"
//
//
//--------------------------------DllMain-----------------------------------------
// called when DLL loads
//
BOOL WINAPI DllMain(HINSTANCE hInst, ULONG ulReason, LPVOID lpReserved)
    {
    switch(ulReason)
        {
        case DLL_PROCESS_ATTACH:
            {
            if(!OpenErrorLog()) return FALSE;
            WriteErrorLog("Opened error log");
            if(!ReadTPCCRegParams())
                {
                WriteErrorLog("TPCC registry key or value missing...exiting");
                return FALSE;
                }

            InitializeCriticalSection(&login_crit_sec);
            InitializeCriticalSection(&TLS_crit_sec);

            // Obtain frequency and initial count of 64-bit counter
            QueryPerformanceFrequency(&freq);
            freqd = (double) freq.QuadPart;
            QueryPerformanceCounter(&tick_count0);

            if(!TMClientInit()) return FALSE;

            break;
            }
        case DLL_PROCESS_DETACH:
            {
            TMClientExit();
            WriteErrorLog("Exiting");
            fclose(fp_errorlog);
            break;
            }
        default:
            break;
        }
    return TRUE;
    } // DllMain

//
//
//-------------------------------GetExtensionVersion--------------------------------
// called by IIS when DLL loads
//
BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO *pVer)
    {
    pVer->dwExtensionVersion = MAKELONG(HSE_VERSION_MINOR, HSE_VERSION_MAJOR);
    lstrcpyn(pVer->lpszExtensionDesc, "Dell TPC-C Client / Web Server",
        HSE_MAX_EXT_DLL_NAME_LEN);
    return TRUE;
    }  // GetExtensionVersion
//
//
//-------------------------------HttpExtensionProc----------------------------------
// called by IIS for each request
//
DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
    {
    enum _command Command;
    int User;
    char* cmd_input_str;
    char html[4096];
    char response[4096];
    BOOL KeepConn = TRUE;
    DWORD len;

    // Get the input string  (assumes GET)
    if (ParseInput(pECB->lpszQueryString, &Command, &User, &cmd_input_str))
        {
        switch(Command)
            {
            case Login:
                ProcessLogin(cmd_input_str, html, &KeepConn);
                break;
            case NewOrderFormRequest:
                sprintf(html, NewOrderForm, dll_path, User, pUserData[User]->w_id);
                break;
            case NewOrder:
                ProcessNewOrder(cmd_input_str, User, html);
                break;
            case PaymentFormRequest:
                sprintf(html, PaymentForm, dll_path, User, pUserData[User]->w_id);
                break;
            case Payment:
                ProcessPayment(cmd_input_str, User, html);
                break;
            case OrderStatusFormRequest:
                sprintf(html, OrderStatusForm, dll_path, User, pUserData[User]->w_id);
                break;
            case OrderStatus:
                ProcessOrderStatus(cmd_input_str, User, html);
                break;
            case DeliveryFormRequest:
                sprintf(html, DeliveryForm, dll_path, User, pUserData[User]->w_id);
                break;
            case Delivery:
                ProcessDelivery(cmd_input_str, User, html);
                break;
            case StockLevelFormRequest:
                sprintf(html, StockLevelForm, dll_path, User, pUserData[User]->w_id,
                    pUserData[User]->d_id);
                break;
            case StockLevel:
                ProcessStockLevel(cmd_input_str, User, html);
                break;
            case Logout:
                ProcessLogout(User);
                sprintf(html, LoginForm, dll_path);
                KeepConn = FALSE;
                break;
            default:
                strcpy(html, "<HTML>ERROR: Command not recognized</HTML>");
                KeepConn = TRUE;
                break;
            }
        }
    else
        {
        strcpy(html, "<HTML> ERROR: Can't parse input</HTML>");
```

Appendix A – Application Source Code

```
        }


    sprintf(response,
        "Content-Type: text/html\r\n"
        "Content-Length: %d\r\n"
        "%s\r\n",
        strlen(html),
        (KeepConn ? "Connection: Keep-Alive\r\n" : ""));

    strcat(response, html);

    len = strlen(response);

    pECB->ServerSupportFunction( pECB->ConnID, HSE_REQ_SEND_RESPONSE_HEADER,
        "200 OK", &len, (LPDWORD) response);

    pECB->dwHttpStatusCode=200; // 200 OK

    return (KeepConn ? HSE_STATUS_SUCCESS_AND_KEEP_CONN : HSE_STATUS_SUCCESS);

    } // HttpExtensionProc
//
//
//--------------------------------ParseInput-----------------------------------------
// parse input string (URL) from browser
// determine command, user, and command input string (remainder of string)
// returns TRUE if successful, FALSE if parsing error
// forms of input string:  CMD=NewOrderFormRequest&USER=1234
//                         |                      |
//                         pCMD                   pUSER
//
//                         CMD=NewOrder&USER=1234&INPUT1=...
//                         |            |         |
//                         pCMD         pUSER     pINPUTS
//
BOOL ParseInput(char* pCMD, enum _command* pCommand, int* User, char** pInputs)
    {
    int i, len;
    char* pUSER;
    char cmd_str[32];
    char user_str[6];
    char* pTemp;


    // Check input string starts with CMD=
    if (strncmp(pCMD, "CMD=", 4)) return FALSE;

    // Copy command string to cmd_str, look up in array of command names
    pUSER = 1 + strchr(pCMD, '&');
    len = pUSER - pCMD - 5;
    strncpy(cmd_str, pCMD + 4, len);
    cmd_str[len] = '\0';
    for(i=0; i<=Logout; i++) if (!strcmp(command_name[i], cmd_str)) break;
    *pCommand = (enum _command) i;

    // Check next PARAM=value pair is USER
    if (strncmp(pUSER, "USER=", 5)) return FALSE;  // string does not contain USER=

    // Copy user string to user_str, convert to integer
    if ((pTemp = strchr(pUSER, '&')) == NULL) //  USER is end of input
        {
        strcpy(user_str, pUSER + 5);
        *pInputs = NULL;
        }
    else
        {
        len = pTemp - pUSER - 5;
        strncpy(user_str, pUSER + 5, len);
        user_str[len] = '\0';
        *pInputs = pTemp + 1;
        }

    *User = atoi(user_str);

    return TRUE;

    } // ParseInput
//
//
//--------------------------------ProcessLogin-----------------------------------------
// process login request; assign user number; create user's command button bar
//
void ProcessLogin(char* cmd_input_str, char* html, BOOL* KeepConn)
    {
    char text[256];
    short w_id, d_id;
    int User = -1;
    int rc, i;
    BOOL success = FALSE;

    *KeepConn = FALSE;

    // If input cannot be parsed or warehouse or district is out of range tell user
    // to re-submit
    if ((rc = ParseLogin(cmd_input_str, &w_id, &d_id)) != SUCCESS)
        sprintf(text, "ERROR: %s", err_text[rc]);
    else if (!(0 < w_id && w_id <= n_warehouses_total))
        strcpy(text, "ERROR: Warehouse out of range - use Back button and resubmit");
    else if (!(0 < d_id && d_id <= 10))
        strcpy(text, "ERROR: District out of range - use Back button and resubmit");
    else // Everything checks out - assign the user a User number if client not full
        {
        // Find first open slot in UserData array and mark it taken
        EnterCriticalSection(&login_crit_sec);
        for (i=0; i < max_users_this_client; i++)
            {
            if (pUserData[i] == 0)
                {
                User = i;
                pUserData[i] = (UserData*) 1;
                break;
                }
            }
        LeaveCriticalSection(&login_crit_sec);

        if (User == -1) // No open slots - tell user to get lost
            strcpy(text,
                "ERROR: Max number of users has been reached - please retry later");

        else            // Success! - initialize User's data
            {
            success = TRUE;
            // Assign user space for his data
            if(!(pUserData[User] = (UserData*) malloc(sizeof(UserData))))
                strcpy(text,
                    "ERROR: no space to allocate for this user - see sys admin");
            else
                {
                // Initialize User's data area with w_id, d_id and button bar
                pUserData[User]->w_id = w_id;
                pUserData[User]->d_id = d_id;
                sprintf(pUserData[User]->ButtonBar, ButtonBar,
                    dll_path, User, dll_path, User, dll_path, User, dll_path, User,
                    dll_path, User, dll_path, User);

                sprintf(text, "Congratulations! You have successfully logged into "
                    "the Dell TPC-C Client!\n"
                    "User: %d  Thread: 0x%X  w_id: %d  d_id: %d\n",
                    User, GetCurrentThreadId(), w_id, d_id);

                *KeepConn = TRUE;
```

Appendix A – Application Source Code

```
                }
            }
        }

    sprintf(html, ResponseHTML, "TPC-C Main Menu",
        success ? pUserData[User]->ButtonBar : "", text);

    } // ProcessLogin
//
//
//--------------------------------ParseLogin----------------------------------------
// parse login input string
// determine w_id, d_id
// returns SUCCESS if successful, error code if parsing error
// form of input string: W_ID=1234&D_ID=12
//                       |         |
//                       pW        pD
//
int ParseLogin(char* pW, short* w_id, short* d_id)
    {
    int len;
    char* pD;
    char wid_str[6];

    pD = 1 + strchr(pW, '&');

    // Check input string
    if (strncmp(pW, "W_ID=", 5)) return ERR_GENERIC;    // first param not W_ID
    if ((len = pD - pW - 6) < 1) return ERR_BLANK_WID;  // blank W_ID field
    if (strncmp(pD, "D_ID=", 5)) return ERR_GENERIC;    // second param not D_ID
    if (strchr(pD, '&') != NULL) return ERR_GENERIC;    // D_ID not last PARAM
    if (*(pD+5) == '\0') return ERR_BLANK_DID;          // blank D_ID field

    // Copy W_ID string to wid_str, convert to integer, checking for non-numerics
    strncpy(wid_str, pW + 5, len);
    wid_str[len] = '\0';
    if (!IsValidNonNegShort(wid_str, w_id)) return ERR_NONNUM_WID;

    //convert D_ID string to integer, checking for non-numerics
    if (!IsValidNonNegShort(pD+5, d_id)) return ERR_NONNUM_DID;

    return SUCCESS;

    } // ParseLogin
//
//
//--------------------------------ProcessNewOrder-----------------------------------
// process NewOrder form
//
void ProcessNewOrder(char* cmd_input_str, int User, char* html)
    {
    int i,rc;
    char text1[2048];
    char text[2048];
    NEW_ORDER_DATA* pNO = (NEW_ORDER_DATA*) pUserData[User]->TransData;

    memset(pUserData[User]->TransData,'\0',sizeof(pUserData[User]->TransData));
    pNO->w_id = pUserData[User]->w_id;

    if ((rc = ParseNewOrder(cmd_input_str, pNO)) != SUCCESS)
        sprintf(text, "ERROR: %s", err_text[rc]);

    else
        {
        // call NewOrder database lookup
        rc = TMNewOrder(pNO);

        switch(rc)
            {
            case SUCCESS:
                pNO->total_amount *=
                    (1 + pNO->w_tax + pNO->d_tax) * (1 - pNO->c_discount);

                sprintf(text, "                                       New Order\n"
                    "Warehouse: %04d"
                    "   District: %02d                        Date: %s\n"
                    "Customer:  %04d"
                    "   Name: %-16s   Credit: %-2s   %%Disc: %05.2f\n"
                    "Order Number: %08d  Number of Lines: %02d        "
                    "W_tax: %05.2f   D_tax: %05.2f\n\n"
                    " Supp_W  Item_Id  Item Name                 Qty  Stock  B/G  "
                    "Price    Amount\n",
                    pNO->w_id, pNO->d_id, pNO->o_entry_d, pNO->c_id, pNO->c_last,
                    pNO->c_credit, 100.0*pNO->c_discount, pNO->o_id, pNO->o_ol_cnt,
                    100.0*pNO->w_tax, 100.0*pNO->d_tax);

                for(i=0; i < pNO->o_ol_cnt; i++)
                    {
                    sprintf(text1, "  %04d   %06d   %-24s  %02d    %03d    "
                        "%1c   $%06.2f  $%07.2f\n",
                        pNO->Ol[i].ol_supply_w_id,
                        pNO->Ol[i].ol_i_id,
                        pNO->Ol[i].ol_i_name,
                        pNO->Ol[i].ol_quantity,
                        pNO->Ol[i].ol_stock,
                        pNO->Ol[i].ol_brand_generic,
                        pNO->Ol[i].ol_i_price,
                        pNO->Ol[i].ol_amount
                        );
                    strcat(text, text1);
                    }

                for (i=pNO->o_ol_cnt; i<15; i++) strcat(text, "\n");
                sprintf(text1, "Execution Status: Transaction committed"
                    "                     Total:  $%08.2f\n", pNO->total_amount);
                strcat(text, text1);
                break;
            case DEADLOCK:
                strcat(text, "ERROR: DEADLOCK");
                break;
            case INVALID_ITEM:
                sprintf(text, "                                       New Order\n"
                    "Warehouse: %04d"
                    "   District:  %02d                       Date:\n"
                    "Customer:  %04d"
                    "   Name: %-16s   Credit: %-2s   %%Disc:\n"
                    "Order Number: %08d  Number of Lines:           "
                    "W_tax:        D_tax:\n\n"
                    " Supp_W  Item_Id  Item Name                 Qty  Stock  B/G  "
                    "Price    Amount\n",
                    pNO->w_id, pNO->d_id, pNO->c_id, pNO->c_last, pNO->c_credit,
                    pNO->o_id);
                strcat(text, "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"
                    "Execution Status: Item number is not valid");
                break;
            case SQL_ERROR:
                strcpy(text, "ERROR:  SQL Error");
                break;
            } // End switch
        } // End else

    sprintf(html, ResponseHTML, "New Order Response",pUserData[User]->ButtonBar,text);

    } // ProcessNewOrder
//
//
//--------------------------------ParseNewOrder-------------------------------------
// parse NewOrder input string
// fill in New Order data structure
// returns SUCCESS if successful, error code if parsing error
// form of input string:
//    D_ID=12&C_ID=1234&W00=1234&I00=123456&Q00=12&W02.... Q14=xx
```

Appendix A – Application Source Code

```
//    |       |         |        |          |
//    pD      pC        pW       pI         pQ
//
int ParseNewOrder(char* pD, NEW_ORDER_DATA* pNO)
    {
    int len, i, n_blanks;
    char str[7];
    char *pC, *pW, *pI, *pQ, *pTemp;
    BOOL last_line;

    pNO->o_all_local = TRUE;

    // Check input string starts with D_ID=
    if (strncmp(pD, "D_ID=", 5)) return ERR_GENERIC;

    // Copy D_ID string to str, convert to integer, checking for non-numerics
    pC = 1 + strchr(pD, '&');
    if ((len = pC - pD - 6) < 1) return ERR_BLANK_DID;          // blank D_ID field
    strncpy(str, pD + 5, len);
    str[len] = '\0';

    if (!IsValidNonNegShort(str, &pNO->d_id)) return ERR_NONNUM_DID; // non-num D_ID

    // Copy C_ID string to str, convert to integer, checking for non-numerics
    pW = 1 + strchr(pC, '&');
    if ((len = pW - pC - 6) < 1) return ERR_BLANK_CID;          // blank C_ID field
    strncpy(str, pC + 5, len);
    str[len] = '\0';
    if (!IsValidNonNegLong(str, &pNO->c_id)) return ERR_NONNUM_CID; // non-num C_ID

    // Parse triplets of Wnn, Inn, Qnn
    //   - item valid only if all 3 fields are valid
    //   - scan all lines; ignore all-blank lines
    //   - check all fields for non-numeric or negative input

    i=0;
    do
        {
        n_blanks = 0;
        last_line = FALSE;

        // Supplying w_id
        pI = 1 + strchr(pW, '&');
        if ((len = pI - pW - 5) < 1) ++n_blanks;            // blank Wxx field
        else
            {
            strncpy(str, pW + 4, len);
            str[len] = '\0';
            if (!IsValidNonNegShort(str, &pNO->Ol[i].ol_supply_w_id))
                return ERR_NONNUM_OL_S_WID;
            pNO->o_all_local &= (pNO->w_id == pNO->Ol[i].ol_supply_w_id);
            }

        // Item id
        pQ = 1 + strchr(pI, '&');
        if ((len = pQ - pI - 5) < 1) ++n_blanks;                // blank Ixx field
        else
            {
            strncpy(str, pI + 4, len);
            str[len] = '\0';
            if (!IsValidNonNegLong(str, &pNO->Ol[i].ol_i_id))
                return ERR_NONNUM_OL_I_ID;
            }

        // Quantity
        if ((pTemp = strchr(pQ, '&')) == NULL) // End of input
            {
            last_line = TRUE;
            pW = strchr(pQ, '\0');
            }
        else
            {
            pW = pTemp + 1;
            }
        if ((len = pW - pQ - (5 - last_line)) < 1 ) ++n_blanks;
        else
            {
            strncpy(str, pQ + 4, len);
            str[len] = '\0';
            if (!IsValidNonNegShort(str, &pNO->Ol[i].ol_quantity))
                return ERR_NONNUM_OL_QUAN;
            }

        if (n_blanks == 1 || n_blanks == 2) return ERR_INCOMPLETE_OL;

        if (n_blanks != 3) ++i;

        }  while (!last_line); // End do

    pNO->o_ol_cnt = i;


    return SUCCESS;

    } // ParseNewOrder
//
//
//--------------------------------ProcessPayment---------------------------------------
// process Payment form
//
void ProcessPayment(char* cmd_input_str, int User, char* html)
    {
    int rc;
    char text[2048];
    PAYMENT_DATA* pPmt = (PAYMENT_DATA*) pUserData[User]->TransData;

    memset(pUserData[User]->TransData,'\0',sizeof(pUserData[User]->TransData));
    pPmt->w_id = pUserData[User]->w_id;

    if ((rc = ParsePayment(cmd_input_str, pPmt)) != SUCCESS)
        sprintf(text, "ERROR: %s", err_text[rc]);

    else
        {
        // call Payment database lookup
        rc = TMPayment(pPmt);

        switch(rc)
            {
            case SUCCESS:
                sprintf(text, "                                                Payment\n"
                    "Date: %s\n\n"
                    "Warehouse: %04d                          "
                    "District: %02d\n"
                    "%-20s                        %-20s\n"
                    "%-20s                        %-20s\n"
                    "%-20s %-2s %5.5s-%4.4s       "
                    "%-20s %-2s %5.5s-%4.4s\n\n"
                    "Customer: %04d  Cust-Warehouse: %04d  Cust-District: %02d\n"
                    "Name:   %-16s %-2s %-16s     Since:  %s\n"
                    "        %-20s                      Credit: %-2s\n"
                    "        %-20s                      %%Disc:  %05.2f\n"
                    "        %-20s %-2s %5.5s-%4.4s       "
                    "Phone:  %-6.6s-%-3.3s-%-3.3s-%-4.4s\n\n"
                    "Amount Paid:          $%07.2f      New Cust-Balance: $%014.2f\n"
                    "Credit Limit:   $%013.2f\n\n"
                    "Cust-Data: %-50.50s\n"
                    "           %-50.50s\n"
                    "           %-50.50s\n"
                    "           %-50.50s\n",
                    pPmt->h_date, pPmt->w_id, pPmt->d_id,
                    pPmt->w_street_1, pPmt->d_street_1,
```

```
                    pPmt->w_street_2, pPmt->d_street_2,
                    pPmt->w_city, pPmt->w_state, pPmt->w_zip, pPmt->w_zip+5,
                    pPmt->d_city, pPmt->d_state, pPmt->d_zip, pPmt->d_zip+5,
                    pPmt->c_id, pPmt->c_w_id, pPmt->c_d_id,
                    pPmt->c_first, pPmt->c_middle, pPmt->c_last, pPmt->c_since,
                    pPmt->c_street_1, pPmt->c_credit,
                    pPmt->c_street_2, 100.0*pPmt->c_discount,
                    pPmt->c_city, pPmt->c_state, pPmt->c_zip, pPmt->c_zip+5,
                    pPmt->c_phone, pPmt->c_phone+6 ,pPmt->c_phone+9, pPmt->c_phone+12,
                    pPmt->h_amount, pPmt->c_balance, pPmt->c_credit_lim,
                    pPmt->c_data, pPmt->c_data+50, pPmt->c_data+100,pPmt->c_data+150);
                break;
            case DEADLOCK:
                strcpy(text, "ERROR: DEADLOCK");
                break;
            case SQL_ERROR:
                strcpy(text, "ERROR:  SQL Error");
                break;
            } // End switch
        } // End else

    sprintf(html, ResponseHTML, "Payment Response",pUserData[User]->ButtonBar,text);

    } // ProcessPayment
//
//
//--------------------------------ParsePayment---------------------------------------
// parse Payment input string
// fill in Payment data structure
// returns SUCCESS if successful, error code if parsing error
// form of input string:
//   D_ID=12&C_ID=1234&C_W_ID=1234&C_D_ID=12&C_LAST=123..16&AMOUNT=1234.56
//   |       |         |           |         |              |
//   pD      pC        pCL         pCD       pCL            pA
//
//   (either C_ID or C_LAST must be blank)
//
int ParsePayment(char* pD, PAYMENT_DATA* pPmt)
    {
    int len;
    char str[7];
    char *pC, *pCW, *pCD, *pCL, *pA;
    BOOL c_id_blank = FALSE, c_last_blank = FALSE;

    // Check input string starts with D_ID=
    if (strncmp(pD, "D_ID=", 5)) return ERR_GENERIC;

    // Copy D_ID string to str, convert to integer, checking for non-numerics
    pC = 1 + strchr(pD, '&');
    if ((len = pC - pD - 6) < 1) return ERR_BLANK_DID;          // blank D_ID field
    strncpy(str, pD + 5, len);
    str[len] = '\0';
    if (!IsValidNonNegShort(str, &pPmt->d_id)) return ERR_NONNUM_DID;// non-num D_ID

    // Copy C_ID string to str, convert to integer, checking for non-numerics
    pPmt->c_id = 0;
    pCW = 1 + strchr(pC, '&');
    if ((len = pCW - pC - 6) < 1) c_id_blank = TRUE;        // blank C_ID field
    else
        {
        strncpy(str, pC + 5, len);
        str[len] = '\0';
        if (!IsValidNonNegLong(str, &pPmt->c_id)) return ERR_NONNUM_CID;
        }

    // Copy C_W_ID string to str, convert to integer, checking for non-numerics
    pCD = 1 + strchr(pCW, '&');
    if ((len = pCD - pCW - 8) < 1) return ERR_BLANK_CWID;       // blank C_W_ID field
    strncpy(str, pCW + 7, len);
    str[len] = '\0';
    if(!IsValidNonNegShort(str, &pPmt->c_w_id))return ERR_NONNUM_CWID;

    // Copy C_D_ID string to str, convert to integer, checking for non-numerics
    pCL = 1 + strchr(pCD, '&');
    if ((len = pCL - pCD - 8) < 1) return ERR_BLANK_CDID;       // blank C_D_ID field
    strncpy(str, pCD + 7, len);
    str[len] = '\0';
    if(!IsValidNonNegShort(str, &pPmt->c_d_id))return ERR_NONNUM_CDID;

    // Copy C_LAST string to c_last
    pA = 1 + strchr(pCL, '&');
    if ((len = pA - pCL - 8) < 1)  c_last_blank = TRUE; // blank C_LAST field
    // both C_ID and C_LAST are blank or both are filled in (error either way)
    if (c_id_blank == c_last_blank) return ERR_CUST_NAME_ID;
    strncpy(pPmt->c_last, pCL + 7, len);
    pPmt->c_last[len] = '\0';

    // Copy AMOUNT string to str, convert to double, checking for non-numerics
    strcpy(str, pA + 7);
    if (!strlen(str)) return ERR_BLANK_AMOUNT;              // blank AMOUNT field
    if (!IsValidNonNegDouble(str, &pPmt->h_amount)) return ERR_NONNUM_AMOUNT;
    if (pPmt->h_amount > 9999.99) return ERR_AMT_TOO_LARGE; // amount is too large

    return SUCCESS;

    } // ParsePayment
//
//
//--------------------------------ProcessOrderStatus---------------------------------
// process OrderStatus form
//
void ProcessOrderStatus(char* cmd_input_str, int User, char* html)
    {
    int rc, i;
    char text[2048], ol_text[100];
    ORDER_STATUS_DATA* pOS = (ORDER_STATUS_DATA*) pUserData[User]->TransData;

    memset(pUserData[User]->TransData,'\0',sizeof(pUserData[User]->TransData));
    pOS->w_id = pUserData[User]->w_id;
    if ((rc = ParseOrderStatus(cmd_input_str, pOS)) != SUCCESS)
        sprintf(text, "ERROR: %s", err_text[rc]);

    else
        {
        // call OrderStatus database lookup
        rc = TMOrderStatus(pOS);

        switch(rc)
            {
            case SUCCESS:
                sprintf(text, "                                        Order-Status\n"
                    "Warehouse: %04d   District: %02d\n"
                    "Customer: %04d   Name: %-16s %-2s %-16s\n"
                    "Cust-Balance: $%09.2f\n\n"
                    "Order-Number: %08d   Entry-Date: %-19s   Carrier-Number: %02d\n"
                    "Supply-W     Item-Id    Qty     Amount      Delivery-Date\n",
                    pOS->w_id, pOS->d_id,
                    pOS->c_id, pOS->c_first, pOS->c_middle, pOS->c_last,
                    pOS->c_balance,
                    pOS->o_id, pOS->o_entry_d, pOS->o_carrier_id);

                for (i=0; i<pOS->o_ol_cnt; i++)
                    {
                    sprintf(ol_text,
                        "  %04d       %06d     %02d     $%08.2lf      %-10s\n",
                        pOS->OlOrderStatusData[i].ol_supply_w_id,
                        pOS->OlOrderStatusData[i].ol_i_id,
                        pOS->OlOrderStatusData[i].ol_quantity,
                        pOS->OlOrderStatusData[i].ol_amount,
                        pOS->OlOrderStatusData[i].ol_delivery_d);
                    strcat(text, ol_text);
                    }
```

```
                break;
            case DEADLOCK:
                strcat(text, "ERROR: DEADLOCK");
                break;
            case SQL_ERROR:
                strcpy(text, "ERROR:  SQL Error");
                break;
            } // End switch
        } // End else

    sprintf(html,ResponseHTML,"OrderStatus Response",pUserData[User]->ButtonBar,text);

    } // ProcessOrderStatus
//
//
//--------------------------------ParseOrderStatus---------------------------------
// parse OrderStatus input string
// fill in OrderStatus data structure
// returns SUCCESS if successful, error code if parsing error
// form of input string:
//   D_ID=12&C_ID=1234&C_LAST=1234567890123456
//    |       |         |
//    pD      pC        pCL
//
//    (either C_ID or C_LAST must be blank)
//
int ParseOrderStatus(char* pD, ORDER_STATUS_DATA* pOS)
    {

    int len;
    char str[7];
    char *pC, *pCL;
    BOOL c_id_blank = FALSE, c_last_blank = FALSE;

    // Check input string starts with D_ID=
    if (strncmp(pD, "D_ID=", 5)) return ERR_GENERIC;

    // Copy D_ID string to str, convert to integer, checking for non-numerics
    pC = 1 + strchr(pD, '&');
    if ((len = pC - pD - 6) < 1) return ERR_BLANK_DID;          // blank D_ID field
    strncpy(str, pD + 5, len);
    str[len] = '\0';
    if (!IsValidNonNegShort(str, &pOS->d_id)) return ERR_NONNUM_DID; // non-num D_ID

    // Copy C_ID string to str, convert to integer, checking for non-numerics
    pOS->c_id = 0;
    pCL = 1 + strchr(pC, '&');
    if ((len = pCL - pC - 6) < 1) c_id_blank = TRUE;        // blank C_ID field
    else
        {
        strncpy(str, pC + 5, len);
        str[len] = '\0';
        if (!IsValidNonNegLong(str, &pOS->c_id)) return ERR_NONNUM_CID;//non-num C_ID
        }

    // Copy C_LAST string to c_last
    strcpy(pOS->c_last, pCL + 7);
    if (!strlen(pOS->c_last)) c_last_blank = TRUE;  // blank C_LAST field
    // both are blank or both are filled in (error either way)
    if (c_id_blank == c_last_blank) return ERR_CUST_NAME_ID;
    return SUCCESS;
    } // ParseOrderStatus
//
//
//--------------------------------ProcessDelivery----------------------------------
// process Delivery form
//
void ProcessDelivery(char* cmd_input_str, int User, char* html)
    {
    int rc;
    char text[256];
    struct _timeb queued;
    DELIVERY_DATA* pDel = (DELIVERY_DATA*) pUserData[User]->TransData;

    memset(pUserData[User]->TransData,'\0',sizeof(pUserData[User]->TransData));
    pDel->w_id = pUserData[User]->w_id;

    // milliseconds since 1/1/70
    _ftime(&queued);
    pDel->queued_time = (LONGLONG) 1000*queued.time + queued.millitm;

    if ((rc = ParseDelivery(cmd_input_str, pDel)) != SUCCESS)
        sprintf(text, "ERROR: %s", err_text[rc]);

    else
        {
        // call Delivery database lookup
        rc = TMDelivery(pDel);
        switch(rc)
            {
            case SUCCESS:
                sprintf(text, "                                          Delivery\n"
                "Warehouse: %04d\n\n"
                "Carrier Number: %02d\n\n"
                "Execution Status: Delivery has been queued\n",
                    pDel->w_id, pDel->o_carrier_id);
                break;
            case DEADLOCK:
                strcat(text, "ERROR: DEADLOCK");
                break;
            case SQL_ERROR:
                strcpy(text, "ERROR:  SQL Error");
                break;
            } // End switch

        } // End else

    sprintf(html,ResponseHTML,"Delivery Response",pUserData[User]->ButtonBar,text);

    } // ProcessDelivery
//
//
//--------------------------------ParseDelivery-----------------------------------
// parse Delivery input string
// fill in Delivery data structure
// returns SUCCESS if successful, error code if parsing error
// form of input string:
//   CARRIER=12
//    |
//    pCAR
//
int ParseDelivery(char* pCAR, DELIVERY_DATA* pDel)
    {

    char str[7];

    // Check input string starts with CARRIER=
    if (strncmp(pCAR, "CARRIER=", 8)) return ERR_GENERIC;

    // Copy CARRIER string to str, convert to integer, checking for non-numerics
    strcpy(str, pCAR + 8);
    if (!strlen(str)) return ERR_BLANK_CARRIER;     // blank CARRIER field
    if (!IsValidNonNegShort(str, &pDel->o_carrier_id))
        return ERR_NONNUM_CARRIER; // CARRIER field is negative or invalid
    return SUCCESS;
    } // ParseDelivery
//
//--------------------------------ProcessStockLevel--------------------------------
// process StockLevel form
//
void ProcessStockLevel(char* cmd_input_str, int User, char* html)
```

Appendix A – Application Source Code

```
    {
    int rc;
    char text[256];
    STOCK_LEVEL_DATA* pSL = (STOCK_LEVEL_DATA*) pUserData[User]->TransData;

    memset(pUserData[User]->TransData,'\0',sizeof(pUserData[User]->TransData));
    pSL->w_id = pUserData[User]->w_id;
    pSL->d_id = pUserData[User]->d_id;
    if ((rc = ParseStockLevel(cmd_input_str, pSL)) != SUCCESS)
        sprintf(text, "ERROR: %s", err_text[rc]);

    else
        {
        // call StockLevel database lookup
        rc = TMStockLevel(pSL);

        switch(rc)
            {
            case SUCCESS:
                sprintf(text, "                                          Stock-Level\n"
                    "Warehouse: %04d   District: %02d\n\n"
                    "Stock Level Threshold: %02d\n\n"
                    "low stock: %03d\n",
                    pSL->w_id, pSL->d_id, pSL->threshold, pSL->low_stock);
                break;
            case DEADLOCK:
                strcat(text, "ERROR: DEADLOCK");
                break;
            case SQL_ERROR:
                strcpy(text, "ERROR:  SQL Error");
                break;
            } // End switch
        } // End else

    sprintf(html,ResponseHTML,"StockLevel Response",pUserData[User]->ButtonBar,text);

    } // ProcessStockLevel
//
//
//--------------------------------ParseStockLevel-----------------------------------
// parse StockLevel input string
// fill in StockLevel data structure
// returns SUCCESS if successful, error code if parsing error
// form of input string:
//   THRESHOLD=12
//    |
//    pT
//
int ParseStockLevel(char* pT, STOCK_LEVEL_DATA* pSL)
    {

    char str[7];

    // Check input string starts with THRESHOLD=
    if (strncmp(pT, "THRESHOLD=", 10)) return ERR_GENERIC;

    // Copy THRESHOLD string to str, convert to integer, checking for non-numerics
    strcpy(str, pT + 10);
    if (!strlen(str)) return ERR_BLANK_THRESHOLD;           // blank THRESHOLD field
    if (!IsValidNonNegShort(str, &pSL->threshold)) return ERR_NONNUM_THRESHOLD;
    return SUCCESS;
    } // ParseStockLevel
//
//-------------------------------ProcessLogout-----------------------------------
// process logout request; reset entry in UserData array, free memory
//
void ProcessLogout(int User)
    {
    EnterCriticalSection(&login_crit_sec);
    pUserData[User] = 0;
    free(pUserData[User]);
    LeaveCriticalSection(&login_crit_sec);

    } // ProcessLogout
//
//
//-------------------------------ReadTPCCRegParams-----------------------------------
// read client-specific and database params from registry
//
BOOL ReadTPCCRegParams()
    {
    char SubKey[100];
    char Value[100];
    HKEY Key;
    DWORD Type;
    DWORD lenData = 4;
    unsigned char Text[200];
    union
        {
        unsigned char data_char[4];
        unsigned int data_int;
        }
        RegDWORD;
    char message[100];

    strcpy(SubKey, "Software\\TPCC");
    if (ERROR_SUCCESS !=
        RegOpenKeyEx(HKEY_LOCAL_MACHINE, SubKey, 0, KEY_QUERY_VALUE, &Key))
        {
        sprintf(message, "No HKEY_LOCAL_MACHINE\\Software\\TPCC registry key");
        WriteErrorLog(message);
        return FALSE;
        }

    strcpy(Value, "MaxUsersThisClient");
    RegQueryValueEx(Key, Value, 0, &Type, RegDWORD.data_char, &lenData);
    max_users_this_client = (int) RegDWORD.data_int;
    sprintf(message, "max_users_this_client= %d", max_users_this_client);
    WriteErrorLog(message);

    strcpy(Value, "NumberOfWarehousesTotal");
    RegQueryValueEx(Key, Value, 0, &Type, RegDWORD.data_char, &lenData);
    n_warehouses_total = (int) RegDWORD.data_int;
    sprintf(message, "n_warehouses_total= %d", n_warehouses_total);
    WriteErrorLog(message);

    lenData = 200;
    strcpy(Value, "DLLPath");
    RegQueryValueEx(Key, Value, 0, &Type, Text, &lenData);
    strncpy(dll_path, (const char*)Text, lenData);
    sprintf(message, "dll_path= %s", dll_path);
    WriteErrorLog(message);

    return TRUE;

    } // ReadTPCCRegParams
//
//
//-------------------------------OpenErrorLog-----------------------------------------
// create and open error log w/ filename error.log in directory log_path
//
BOOL OpenErrorLog()
    {
    char errorlog_fn[250];
    char SubKey[100];
    char Value[] = "LogPath";
    HKEY Key;
    DWORD Type;
    DWORD lenData = 200;
    unsigned char Text[200];

    // read log_path from registry key \Software\TPCC\LogPath
```

```
    strcpy(SubKey, "Software\\TPCC");
    if (ERROR_SUCCESS !=
        RegOpenKeyEx(HKEY_LOCAL_MACHINE, SubKey, 0, KEY_QUERY_VALUE, &Key))
        return FALSE;
    RegQueryValueEx(Key, Value, 0, &Type, Text, &lenData);
    ExpandEnvironmentStrings((const char*) Text, log_path, 250);

    // Create file pathname, open file, return FALSE if error
    sprintf(errorlog_fn, "%s\\error_client.log", log_path);
    if((fp_errorlog = fopen(errorlog_fn, "w")) == NULL) return FALSE;
    else return TRUE;
    } // OpenErrorLog
//
//
//--------------------------------WriteErrorLog---------------------------------------
// Prepend error messages with date/timestamp and thread info; write to file & flush
//
void WriteErrorLog(char* message)
    {
    char d[10];
    char t[10];
    struct _timeb tb;

    _strdate(d);
    _strtime(t);
    _ftime(&tb);
    fprintf(fp_errorlog, "%s %s.%03u Thread: 0x%03X %s\n",
        d, t, tb.millitm, GetCurrentThreadId(), message);
    fflush(fp_errorlog);
    } // WriteErrorLog
//
//
//
//--------------------------------IsValidNonNegShort-----------------------------
// check if text string represents a valid, non-negative SHORT and convert
//
BOOL IsValidNonNegShort(char* str, short* number)
    {
    char *non_numeric_chars;
    BOOL all_trailing_chars_are_blanks;
    int i, l;

    *number = (short) strtol(str, &non_numeric_chars, 10);

    // If strtol says there are non numeric trailing chars we need to check
    // in case they are all blank
    if (l=strlen(non_numeric_chars))
        {
        if (l == (int) strlen(str)) return FALSE;  // entire str is non-numeric
        all_trailing_chars_are_blanks = TRUE;
        for (i=0; i<l; i++)
            {
            if (*(non_numeric_chars + i) != ' ' &&
                *(non_numeric_chars + i) != '+')        // blanks are +'s
                {
                all_trailing_chars_are_blanks = FALSE;
                break;
                }
            }
        if (!all_trailing_chars_are_blanks) return FALSE;
        }

    // Check number not negative
    if (*number < 0) return FALSE;


    return TRUE;
    }

//
//
```

```
//--------------------------------IsValidNonNegLong-----------------------------
// check if text string represents a valid, non-negative LONG and convert
//
BOOL IsValidNonNegLong(char* str, long* number)
    {
    char *non_numeric_chars;
    BOOL all_trailing_chars_are_blanks;
    int i, l;

    *number = (long) strtol(str, &non_numeric_chars, 10);

    // If strtol says there are non numeric trailing chars we need to check
    // in case they are all blank
    if (l=strlen(non_numeric_chars))
        {
        if (l == (int) strlen(str)) return FALSE;  // entire str is non-numeric
        all_trailing_chars_are_blanks = TRUE;
        for (i=0; i<l; i++)
            {
            if (*(non_numeric_chars + i) != ' ' &&
                *(non_numeric_chars + i) != '+')        // blanks are +'s
                {
                all_trailing_chars_are_blanks = FALSE;
                break;
                }
            }
        if (!all_trailing_chars_are_blanks) return FALSE;
        }

    // Check number not negative
    if (*number < 0) return FALSE;


    return TRUE;
    }

//
//
//--------------------------------IsValidNonNegDouble-----------------------------
// check if text string represents a valid, non-negative DOUBLE and convert
//
BOOL IsValidNonNegDouble(char* str, double* number)
    {
    char *non_numeric_chars;
    BOOL all_trailing_chars_are_blanks;
    int i, l;

    *number = (double) strtod(str, &non_numeric_chars);

    // If strtol says there are non numeric trailing chars we need to check
    // in case they are all blank
    if (l=strlen(non_numeric_chars))
        {
        if (l == (int) strlen(str)) return FALSE;  // entire str is non-numeric
        all_trailing_chars_are_blanks = TRUE;
        for (i=0; i<l; i++)
            {
            if (*(non_numeric_chars + i) != ' ' &&
                *(non_numeric_chars + i) != '+')        // blanks are +'s
                {
                all_trailing_chars_are_blanks = FALSE;
                break;
                }
            }
        if (!all_trailing_chars_are_blanks) return FALSE;
        }

    // Check number not negative
    if (*number < 0) return FALSE;
```

```
    return TRUE;
    }

//

//--------------------------------End of tpcc.c-------------------------------------
```

Appendix A – Application Source Code

tmclient.c

```
//----------------------tmclient.c: Dell TPC-C Client / Web Server--------------------
//
// Copyright (c) 1997 Dell Computer Corporation, All Rights Reserved
//
// Author: James Jordan/Dave Jaffe              Last modified: 2/2/99
//
// Audited: Richard Gimarc/Tom Sawyer  Performance Metrics Inc.
//
//
// Transaction Monitor client code; compiled into tpcc.dll ISAPI; included by tpcc.h
//
//
//--------------------------------TMClientInit-----------------------------------------
// Initializes transaction monitor client
//
BOOL TMClientInit()
    {
    int  spid = 0;

    tls_idx = TlsAlloc();
    if (tls_idx < 0)
        {
        WriteErrorLog("ERROR: TMInit -> TLS index failed to allocate!");
        return SQL_ERROR;
        }

    return TRUE;
    } // TMInit
//
//
//--------------------------------TMClientExit-----------------------------------------
// Exits transaction monitor client
//
void TMClientExit()
        {
        if ((TlsFree(tls_idx)) == 0) WriteErrorLog("Failed to free TLS");
        }
//
//
//
//--------------------------------TMNewOrder-------------------------------------------
// Transaction Monitor NewOrder client function: calls Tuxedo server TMNO
//
int TMNewOrder(NEW_ORDER_DATA* p1)
    {
        int rc;
        NEW_ORDER_DATA *p2;

        if (IsTpInit() == -1)
            {
            TPPrintError("IsTPInit_NO");
            return SQL_ERROR;
            }

        if((p2 = (NEW_ORDER_DATA *) tpalloc("CARRAY", NULL, NO_len_ot)) == NULL)
            {
            TPPrintError("TPALLOC_NO");
            return SQL_ERROR;
            }
        memcpy(p2,p1,NO_len_ot);
        rc = tpcall("TMNO",(char *) p2, NO_len_ot,(char**)&p2,&NO_len_ot, TPSIGRSTRT);
        if ( rc == -1)
            {
            TPPrintError("TPCALL_NO");
            return SQL_ERROR;
            };
        memcpy(p1,p2,NO_len_ot);
        tpfree((char*)p2);
        return tpurcode;
    }
//
//
//--------------------------------TMPayment--------------------------------------------
// Transaction Monitor Payment client function: calls Tuxedo server TMPY
//
int TMPayment(PAYMENT_DATA* p1)
    {
        int rc;
        PAYMENT_DATA *p2;

        if (IsTpInit() == -1)
            {
            TPPrintError("IsTPInit_PY");
            return SQL_ERROR;
            }

        if((p2 = (PAYMENT_DATA *) tpalloc("CARRAY", NULL, PY_len_ot)) == NULL)
            {
            TPPrintError("TPALLOC_PY");
            return SQL_ERROR;
            }
        memcpy(p2,p1,PY_len_ot);
        rc = tpcall("TMPY",(char *) p2, PY_len_ot,(char**)&p2,&PY_len_ot, TPSIGRSTRT);
        if (rc  == -1)
            {
            TPPrintError("TPCALL_PY");
            return SQL_ERROR;
            };
        memcpy(p1,p2,PY_len_ot);
        tpfree((char*)p2);
        return tpurcode;
    }
//
//
//--------------------------------TMOrderStatus----------------------------------------
// Transaction Monitor OrderStatus client function: calls Tuxedo server TMOS
//
int TMOrderStatus(ORDER_STATUS_DATA* p1)
    {
        int rc;
        ORDER_STATUS_DATA *p2;

        if (IsTpInit() == -1)
            {
            TPPrintError("IsTPInit_OS");
            return SQL_ERROR;
            }

        if((p2 = (ORDER_STATUS_DATA *) tpalloc("CARRAY", NULL, OS_len_ot)) == NULL)
            {
            TPPrintError("TPALLOC_OS");
            return SQL_ERROR;
            }
        memcpy(p2,p1,OS_len_ot);
        rc = tpcall("TMOS",(char *) p2, OS_len_ot,(char**)&p2,&OS_len_ot, TPSIGRSTRT);
        if ( rc == -1)
            {
            TPPrintError("TPCALL_OS");
            return SQL_ERROR;
            };
        memcpy(p1,p2,OS_len_ot);
        tpfree((char*)p2);
        return tpurcode;
    }
//
//
```

Appendix A – Application Source Code

```
//-------------------------------TMDelivery-----------------------------------------
// Transaction Monitor Delivery client function: calls Tux server TMDL asynchronously
//
int TMDelivery(DELIVERY_DATA *p1)
    {
        int rc;
        DELIVERY_DATA *p2;

        if (IsTpInit() == -1)
            {
            TPPrintError("IsTPInit_DL");
            return SQL_ERROR;
            }


        if((p2 = (DELIVERY_DATA *) tpalloc("CARRAY", NULL, DL_len_ot)) == NULL)
            {
            TPPrintError("TPALLOC_DL");
            return SQL_ERROR;
            }
        memcpy(p2,p1,DL_len_ot);
        rc = tpacall("TMDL",(char *) p2, DL_len_ot,TPNOREPLY);
        if ( rc  == -1)
            {
            TPPrintError("TPCALL_DL");
            return SQL_ERROR;
            };
        tpfree((char*)p2);
        return SUCCESS;
    }
//
//
//-------------------------------TMStockLevel---------------------------------------
// Transaction Monitor StockLevel client function: calls Tuxedo server TMSL
//
int TMStockLevel(STOCK_LEVEL_DATA* p1)
    {
        int rc;
        STOCK_LEVEL_DATA *p2;

        if (IsTpInit() == -1)
            {
            TPPrintError("IsTPInit_SL");
            return SQL_ERROR;
            }

        if((p2 = (STOCK_LEVEL_DATA *) tpalloc("CARRAY", NULL, SL_len_ot)) == NULL)
            {
            TPPrintError("TPALLOC_SL");
            return SQL_ERROR;
            }
        memcpy(p2,p1,SL_len_ot);
        rc = tpcall("TMSL",(char *) p2, SL_len_ot,(char**)&p2,&SL_len_ot, TPSIGRSTRT);
        if (rc  == -1)
            {
            TPPrintError("TPCALL_SL");
            return SQL_ERROR;
            };
        memcpy(p1,p2,SL_len_ot);
        tpfree((char*)p2);
        return tpurcode;
    }
//
//
//-------------------------------IsTPInit()-----------------------------------------
//
int IsTpInit()
    {
    TPINIT *tpinf;
    int x=1, rc=-1, cnt=0;
    static int num_tpinits=0;

    if(!(TlsGetValue(tls_idx)))
        {
        EnterCriticalSection(&TLS_crit_sec);
        while(rc == -1)
            {
            tpinf = (TPINIT *) tpalloc("TPINIT","",sizeof(TPINIT));
            tpinf->flags |= TPMULTICONTEXTS;
            itoa(++num_tpinits, tpinf->cltname, 10);
            rc=tpinit(tpinf);
            if (rc != -1) TlsSetValue(tls_idx,&x);
            else TPPrintError("TPINIT");
            tpfree((char*)tpinf);

            if(cnt > 20)
                {
                rc=SQL_ERROR;
                break;
                }
            Sleep(cnt++);
            }
        LeaveCriticalSection(&TLS_crit_sec);
        return (rc);
        }
    return 0;
    } //    End IsTPInit()
//
//
//-------------------------------TPPrintError()-------------------------------------
// print tuxedo error message
//
//
static void TPPrintError(char *msg)
    {
     char errbuf[512];

    switch(tperrno)
        {
        case TPEINVAL :
            sprintf(errbuf, "TPERROR: %s TPEINVAL error\n", msg);
            break;
        case TPENOENT :
            sprintf(errbuf, "TPERROR:%s TPENOENT error\n", msg);
            break;
        case TPEPERM :
            sprintf(errbuf, "TPERROR: %s TPEPERM error\n", msg);
            break;
        case TPEPROTO :
            sprintf(errbuf, "TPERROR: %s TPEPROTO error\n", msg);
            break;
        case TPESYSTEM :
            sprintf(errbuf, "TPERROR: %s TPESYSTEM error see tuxedo log file\n", msg);
            break;
        case TPEOS :
            sprintf(errbuf, "TPERROR: %s TPEOS error\n", msg);
            break;
        default:
            sprintf(errbuf, "TPERROR: %s TP default Tuxedo error tperrno %d\n",
                msg,tperrno);
        } // end of switch

    WriteErrorLog(errbuf);
    return;
    } // end of TPPrintError
//
//
//
//-------------------------------End of tmclient.c----------------------------------
```

Appendix A – Application Source Code

Commands for compiling and linking tpcc.dll

```
rem   tpcctuxdllmak.cmd: run this from NT command prompt to compile tpcc.dll
rem   then copy tpcc.dll to its ISAPI script location
cl.exe /nologo /MT /W3 /O2 /I c:\tuxedo\include /D WIN32 /D _WINDOWS  /D _TMSTHREADS /c
    tpcc.c
link.exe kernel32.lib  advapi32.lib libtux.lib libtux2.lib libbuft.lib libgp.lib
    libwsc.lib /nologo /subsystem:windows /dll /incremental:no /machine:I386 /def:tpcc.def
    /out:tpcc.dll /implib:tpcc.lib /libpath:c:\tuxedo\lib tpcc.obj
```

Appendix A – Application Source Code

Tuxedo Server Source Code

tmserver.h

```
//----------------------tmserver.h: Dell TPC-C Transaction Monitor Server------------
//
// Copyright (c) 1998 Dell Computer Corporation, All Rights Reserved
//
// Author: James Jordan             Last modified: 10/8/97
//
// Audited: Richard Gimarc, Performance Metrics Inc. 10/9/97
//
// header file for Transaction Monitor tpcc server code
//

#ifndef _TM_SERVER_
#define _TM_SERVER_

#include <windows.h>
#include <stdio.h>
#include <process.h>
#include <httpext.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <math.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <io.h>
#include <ctype.h>

//  tuxedo header
#include <tmenv.h>
#include <xa.h>
#include <atmi.h>
#include <userlog.h>

// DBLib Headers for MS SQL Server
#define DBNTWIN32
#include <sqlfront.h>
#include <sqldb.h>

// tpcc include files
#include "trans.h"      //  transaction data structure definitions

//Functions
void ReadTPCCRegParams();
BOOL OpenDeliveryLog();
int  OpenErrorLog();
void WriteErrorLog(char* message);


// Transaction Monitor functions
void TMNO (TPSVCINFO *rqst);
void TMPY (TPSVCINFO *rqst);
void TMOS (TPSVCINFO *rqst);
void TMDL (TPSVCINFO *rqst);
void TMSL (TPSVCINFO *rqst);

// Database-specific functions
void DBInit();
void DBExit();
BOOL DBOpenConnection(DBPROCESS** ppDbproc, char* server, char* database, char* user,
                      char* password, char* app, int* spid, long* pack_size);
void DBCloseConnection(DBPROCESS* pDbproc);
int  DBNewOrder(DBPROCESS* pDbproc, NEW_ORDER_DATA* pNewOrder);
int  DBPayment(DBPROCESS* pDbproc, PAYMENT_DATA* pPayment);
int  DBOrderStatus(DBPROCESS* pDbproc, ORDER_STATUS_DATA* pOrderStatus);
int  DBDelivery(DBPROCESS* pDbproc, DELIVERY_DATA* pDelivery);
int  DBStockLevel(DBPROCESS* pDbproc, STOCK_LEVEL_DATA* pStockLevel);

// Variables read from registry key Software\TPCC on LocalMachine
//    Same on all clients:
char server[32];            // Name of database server machine
                            //   (REG_SZ: DatabaseServer)
char database_name[32];     // Name of database (REG_SZ: DatabaseName)
char database_user[32];     // Database user login name (REG_SZ: DatabaseUser)
char database_passwd[32];   // Database user login password
                            //   (REG_SZ: DatabasePassword)
char log_path[250];         // Path for delivery and other logs
                            //   (REG_EXPAND: LogPath)

// Return codes from TM stored procedure calls
#define SQL_ERROR       -1  // Usually incorrect C_ID or C_LAST
#define SUCCESS         0   // Success
#define DEADLOCK        1   // Still deadlocked after specified DEADLOCK_RETRYs
#define INVALID_ITEM    2   // Invalid item in NewOrder

#define DEADLOCK_WAIT   10
#define DEADLOCK_RETRY  5

//Global variables
FILE *fp_delivlog, *fp_errorlog;

LARGE_INTEGER freq;
double freqd;

CRITICAL_SECTION deliv_write_crit_sec;

// dbproc handle for connection to database.
DBPROCESS* hDB;

#include "sqlfuncs.c"       //      MSSQL tpc-c transaction function wrappers


#endif// _TM_SERVER_
```

Appendix A – Application Source Code

tmserver_stub.c

```
//----------------------tmserver_stub.c: Dell TPC-C Transaction Monitor Server-------
//
// Copyright (c) 1997 Dell Computer Corporation, All Rights Reserved
//
// Author: James Jordan             Last modified: 2/2/99
//
// Audited: Richard Gimarc/Tom Sawyer  Performance Metrics Inc.
//
// Transaction Monitor server code for tpcc benchmark

// Stub used to compile any of the following:
//              _3_TRANS = NO, PY, OS
//              _4_TRANS = NO, PY, OS, SL
//              _ALL_TRANS = NO, PY, OS, DL, SL
//              each transaction in an individual tuxedo server.
//
//

#include <stdio.h>
#include <xa.h>
#include <atmi.h>

#if defined(__cplusplus)
extern "C" {
#endif
extern int _tmrunserver _((int));
#ifdef _TUX_D
extern void TMDL _((TPSVCINFO *));
#endif
#ifdef _TUX_NO
extern void TMNO _((TPSVCINFO *));
#endif
#ifdef _TUX_OS
extern void TMOS _((TPSVCINFO *));
#endif
#ifdef _TUX_PY
extern void TMPY _((TPSVCINFO *));
#endif
#ifdef _TUX_SL
extern void TMSL _((TPSVCINFO *));
#endif
#if defined(__cplusplus)
}
#endif

#ifdef _3_TRANS
static struct tmdsptchtbl_t _tmdsptchtbl[] = {
#ifdef _TUX_NO
    { "TMNO", "TMNO", (void (*) _((TPSVCINFO *))) TMNO, 0, 0 },
#endif
#ifdef _TUX_OS
    { "TMOS", "TMOS", (void (*) _((TPSVCINFO *))) TMOS, 1, 0 },
#endif
#ifdef _TUX_PY
    { "TMPY", "TMPY", (void (*) _((TPSVCINFO *))) TMPY, 2, 0 },
#endif
#ifdef _TUX_SL
    { "TMSL", "TMSL", (void (*) _((TPSVCINFO *))) TMSL, 3, 0 },
#endif
    { NULL, NULL, NULL, 0, 0 }
};

#elif _4_TRANS
static struct tmdsptchtbl_t _tmdsptchtbl[] = {
#ifdef _TUX_NO
    { "TMNO", "TMNO", (void (*) _((TPSVCINFO *))) TMNO, 0, 0 },
#endif
#ifdef _TUX_OS
    { "TMOS", "TMOS", (void (*) _((TPSVCINFO *))) TMOS, 1, 0 },
#endif
#ifdef _TUX_PY
    { "TMPY", "TMPY", (void (*) _((TPSVCINFO *))) TMPY, 2, 0 },
#endif
#ifdef _TUX_SL
    { "TMSL", "TMSL", (void (*) _((TPSVCINFO *))) TMSL, 3, 0 },
#endif
    { NULL, NULL, NULL, 0, 0 }
};

#elif _ALL_TRANS
static struct tmdsptchtbl_t _tmdsptchtbl[] = {
#ifdef _TUX_D
    { "TMDL", "TMDL", (void (*) _((TPSVCINFO *))) TMDL, 0, 0 },
#endif
#ifdef _TUX_NO
    { "TMNO", "TMNO", (void (*) _((TPSVCINFO *))) TMNO, 1, 0 },
#endif
#ifdef _TUX_OS
    { "TMOS", "TMOS", (void (*) _((TPSVCINFO *))) TMOS, 2, 0 },
#endif
#ifdef _TUX_PY
    { "TMPY", "TMPY", (void (*) _((TPSVCINFO *))) TMPY, 3, 0 },
#endif
#ifdef _TUX_SL
    { "TMSL", "TMSL", (void (*) _((TPSVCINFO *))) TMSL, 4, 0 },
#endif
    { NULL, NULL, NULL, 0, 0 }
};

#else
static struct tmdsptchtbl_t _tmdsptchtbl[] = {
#ifdef _TUX_D
    { "TMDL", "TMDL", (void (*) _((TPSVCINFO *))) TMDL, 0, 0 },
#endif
#ifdef _TUX_NO
    { "TMNO", "TMNO", (void (*) _((TPSVCINFO *))) TMNO, 0, 0 },
#endif
#ifdef _TUX_OS
    { "TMOS", "TMOS", (void (*) _((TPSVCINFO *))) TMOS, 0, 0 },
#endif
#ifdef _TUX_PY
    { "TMPY", "TMPY", (void (*) _((TPSVCINFO *))) TMPY, 0, 0 },
#endif
#ifdef _TUX_SL
    { "TMSL", "TMSL", (void (*) _((TPSVCINFO *))) TMSL, 0, 0 },
#endif
    { NULL, NULL, NULL, 0, 0 }
};
#endif

#ifndef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

_TMDLLIMPORT extern struct xa_switch_t tmnull_switch;

struct tmsvrargs_t tmsvrargs = {
    NULL,
    &_tmdsptchtbl[0],
    0,
    tpsvrinit,
    tpsvrdone,
    _tmrunserver,   /* PRIVATE  */
    NULL,           /* RESERVED */
    NULL,           /* RESERVED */
```

```
    NULL,           /* RESERVED */
    NULL            /* RESERVED */
};

struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
_tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
    tmsvrargs.xa_switch = &tmnull_switch;
    return(&tmsvrargs);
}

int
#ifdef _TMPROTOTYPES
main(int argc, char **argv)
#else
main(argc,argv)
int argc;
char **argv;
#endif
{
#ifdef TMMAINEXIT
#include "mainexit.h"
#endif

    return( _tmstartserver( argc, argv, _tmgetsvrargs()));
}
```

Appendix A – Application Source Code

tmserver.c

```
//----------------------tmserver.c: Dell TPC-C Transaction Monitor Server-------------
//
// Copyright (c) 1997 Dell Computer Corporation, All Rights Reserved
//
// Author: James Jordan, Dave Jaffe         Last modified: 2/2/99
//
// Audited: Richard Gimarc/Tom Sawyer  Performance Metrics Inc.
//
// Transaction Monitor server code for tpcc benchmark
//
//

#include "tmserver.h"

//-----------------------------------tpsvrinit-----------------------------------------
//  main entry point for application. called by tmboot at process startup time.
//  returns zero if able to init database and open database connection. otherwise,
//  returns negative one(-1) which will cause tmboot to shutdown process.

int tpsvrinit(int argc, char *argv[])
    {
    int spid = 0; // database connection process id

    // so compiler does not complain about not using variables
    argc = argc;
    argv = argv;

    // Open Error Log
    if(!OpenErrorLog()) return FALSE;
    WriteErrorLog("Error Log Opened...");

    // Read database parameters from registry
    if(!ReadTPCCRegParams())
        {
        WriteErrorLog("TPCC registry key or value missing...exiting");
        return FALSE;
        }

#ifdef _TUX_D
    if(!OpenDeliveryLog()) return FALSE;
#endif

    DBInit();
    DBOpenConnection(&hDB, server, database_name, database_user, database_passwd,
        "Client", &spid, (long*) 4096);

#ifdef _TUX_NO
    userlog("Starting NewOrder service.....");
#endif
#ifdef _TUX_PY
    userlog("Starting Payment service.....");
#endif
#ifdef _TUX_OS
    userlog("Starting OrderStatus service.....");
#endif
#ifdef _TUX_D
    userlog("Starting Delivery service.....");
#endif
#ifdef _TUX_SL
    userlog("Starting StockLevel service.....");
#endif

    userlog("TPC-C Tuxedo Server started");
    return(0);
    } // End tpsvrinit()

//
//
//-----------------------------------tpsvrdone-----------------------------------------
// Exits transaction monitor server function - called by TMShutdown

void tpsvrdone( void )
    {
    DBCloseConnection(hDB);
    DBExit();
    fclose(fp_errorlog);
#ifdef _TUX_D
    fclose(fp_delivlog);
#endif
    }
//
#ifdef _TUX_NO
//-----------------------------------TMNO----------------------------------------------
// Transaction Monitor NewOrder server function: calls DBNewOrder
//
void TMNO (TPSVCINFO *rqst)
    {

    int rc;

    tpreturn(TPSUCCESS,rc = (DBNewOrder(hDB, (NEW_ORDER_DATA *) rqst->data)) ,
             rqst->data, rqst->len, 0);
    }
//
#endif

#ifdef _TUX_PY
//
//-----------------------------------TMPY----------------------------------------------
// Transaction Monitor Payment server function: calls DBPayment
//
void TMPY (TPSVCINFO *rqst)
    {
    int rc;

    tpreturn(TPSUCCESS,rc = (DBPayment(hDB, (PAYMENT_DATA *) rqst->data)) ,
             rqst->data, rqst->len , 0);
    }
//
#endif

#ifdef _TUX_OS
//
//-----------------------------------TMOS----------------------------------------------
// Transaction Monitor OrderStatus server function: calls DBOrderStatus
//
void TMOS (TPSVCINFO *rqst)
    {
    int rc;

    tpreturn(TPSUCCESS,
        rc = (DBOrderStatus(hDB, (ORDER_STATUS_DATA *) rqst->data)),
             rqst->data, rqst->len, 0);
    }
//
#endif

#ifdef _TUX_D
//
//-----------------------------------TMDL----------------------------------------------
// Transaction Monitor Delivery server function: calls DBDelivery
//
void TMDL (TPSVCINFO *rqst)
    {
    int rc;
    rc = (DBDelivery(hDB, (DELIVERY_DATA *) rqst->data));
    tpreturn(TPSUCCESS, 0 ,
```

```
            NULL, 0, 0);
    }
//
#endif

//
//-----------------------------------TMSL-------------------------------------------
// Transaction Monitor StockLevel server function: calls DBStockLevel
//
#ifdef _TUX_SL
void TMSL (TPSVCINFO *rqst)
    {
    int rc;

    tpreturn(TPSUCCESS,
        rc = (DBStockLevel(hDB, (STOCK_LEVEL_DATA *) rqst->data)) ,
        rqst->data, rqst->len, 0);
    }
#endif
//
//
//-------------------------------OpenErrorLog---------------------------------------
// create and open error log w/ filename error_server_XX.log in directory log_path
//
BOOL OpenErrorLog()
    {
    char Value[] = "LogPath";
    char SubKey[100];
    HKEY Key;
    DWORD Type;
    DWORD lenData = 200;
    unsigned char Text[200];
    char errorlog_fn[250];

    strcpy(SubKey, "Software\\TPCC");

    if (ERROR_SUCCESS !=
        RegOpenKeyEx(HKEY_LOCAL_MACHINE, SubKey, 0, KEY_QUERY_VALUE, &Key))
        {
        userlog("Can't read registry entry TPCC");
        return FALSE;
        }
    RegQueryValueEx(Key, Value, 0, &Type, Text, &lenData);
    ExpandEnvironmentStrings((const char*) Text, log_path, 250);

    // Create file pathname, open file, return FALSE if error
    // ALL_TRANS: one server program for all 5 txn types
#ifdef _ALL_TRANS
    sprintf(errorlog_fn, "%s\\error_server.log", log_path);
    //  _3_TRANS: NO, PY, OS
#elif _3_TRANS
    sprintf(errorlog_fn, "%s\\error_server.log", log_path);
    //  _4_TRANS: NO, PY, OS, SL
#elif _4_TRANS
    sprintf(errorlog_fn, "%s\\error_server.log", log_path);
#elif _TUX_NO
    sprintf(errorlog_fn, "%s\\error_server_no.log", log_path);
#elif _TUX_PY
    sprintf(errorlog_fn, "%s\\error_server_py.log", log_path);
#elif _TUX_OS
    sprintf(errorlog_fn, "%s\\error_server_os.log", log_path);
#elif _TUX_D
    sprintf(errorlog_fn, "%s\\error_server_dl.log", log_path);
#elif _TUX_SL
    sprintf(errorlog_fn, "%s\\error_server_sl.log", log_path);
#endif

    if((fp_errorlog = fopen(errorlog_fn, "w")) == NULL)
        {
        userlog("Can't open error log %s\n", errorlog_fn);
        return FALSE;
        }
    else return TRUE;

    } // OpenErrorLog
//
//
///------------------------------ReadTPCCRegParams---------------------------------
// read database params from registry
//
BOOL ReadTPCCRegParams()
    {
    char SubKey[100];
    char Value[100];
    HKEY Key;
    DWORD Type;
    DWORD lenData = 200;
    unsigned char Text[200];
    char message[100];

    strcpy(SubKey, "Software\\TPCC");
    if (ERROR_SUCCESS !=
        RegOpenKeyEx(HKEY_LOCAL_MACHINE, SubKey, 0, KEY_QUERY_VALUE, &Key))
        {
        sprintf(message, "No HKEY_LOCAL_MACHINE\\Software\\TPCC registry key");
        WriteErrorLog(message);
        return FALSE;
        }

    lenData = 200;
    strcpy(Value, "DatabaseServer");
    RegQueryValueEx(Key, Value, 0, &Type, Text, &lenData);
    strncpy(server, (const char*)Text, lenData);
    sprintf(message, "server= %s", server);
    WriteErrorLog(message);

    lenData = 200;
    strcpy(Value, "DatabaseName");
    RegQueryValueEx(Key, Value, 0, &Type, Text, &lenData);
    strncpy(database_name, (const char*)Text, lenData);
    sprintf(message, "database_name= %s", database_name);
    WriteErrorLog(message);

    lenData = 200;
    strcpy(Value, "DatabaseUser");
    RegQueryValueEx(Key, Value, 0, &Type, Text, &lenData);
    strncpy(database_user, (const char*)Text, lenData);
    sprintf(message, "database_user= %s", database_user);
    WriteErrorLog(message);

    lenData = 200;
    strcpy(Value, "DatabasePassword");
    RegQueryValueEx(Key, Value, 0, &Type, Text, &lenData);
    strncpy(database_passwd, (const char*)Text, lenData);
    sprintf(message, "database_passwd= %s", database_passwd);
    WriteErrorLog(message);

    return TRUE;

    } // ReadTPCCRegParams
//
//
//-------------------------------OpenDeliveryLog------------------------------------
// create and open delivery log w/ filename log_path\dyymmddhhmm_processid.log
//
#ifdef _TUX_D
BOOL OpenDeliveryLog()
    {
    char d[10];
    char t[10];
    char delivlog_fn[64];
```

```
    InitializeCriticalSection(&deliv_write_crit_sec);

    _strdate(d);
    _strtime(t);
    sprintf(delivlog_fn, "%s\\d%c%c%c%c%c%c%c%c%c_%d.log",
        log_path, d[6], d[7], d[0], d[1], d[3], d[4], t[0], t[1], t[3], t[4],
        GetCurrentProcessId());
    if((fp_delivlog = fopen(delivlog_fn, "w")) == NULL)
        {
        return FALSE;
        }
    else return TRUE;
    } // OpenDeliveryLog
#endif
//
//
//-------------------------------WriteErrorLog-------------------------------------
// Prepend error messages with date/timestamp and thread info; write to file & flush
//
void WriteErrorLog(char* message)
    {
    char d[10];
    char t[10];
    struct _timeb tb;

    _strdate(d);
    _strtime(t);
    _ftime(&tb);
    fprintf(fp_errorlog, "%s %s.%03u Thread: 0x%03X %s\n",
        d, t, tb.millitm, GetCurrentThreadId(), message);
    fflush(fp_errorlog);
    } // WriteErrorLog
//
//
//
//-------------------------------End of tmserver.c---------------------------------
```

sqlfuncs.c

```
//--------------------sqlfuncs.c: Dell TPC-C Client / Web Server----------------------
//
// Based on SQLFUNCS.C from Microsoft (Damien Lindauer)
//
// Modfied by: Dave Jaffe/James Jordan              Last modified: 2/2/99
//
// Audited: Richard Gimarc/Tom Sawyer  Performance Metrics Inc.
//
// MS SQL Server interface for TPC-C
// Included by tmserver.h
//
//
// Local functions
int SQLErrHandler(DBPROCESS* pDbproc, int severity, int err, int oserr,
                  char* dberrstr, char* oserrstr);
int SQLMsgHandler(DBPROCESS* pDbproc, DBINT msgno, int msgstate, int severity,
                  char *msgtext);
//
//
// Local defines
#define BUF_LEN 100
//
//
//------------------------------DBInit-----------------------------------------------
// initializes database
//
void DBInit()
    {
    dbinit();
    dberrhandle((DBERRHANDLE_PROC) SQLErrHandler);
    dbmsghandle((DBMSGHANDLE_PROC) SQLMsgHandler);
    } // DBInit
//
//
//------------------------------DBExit-----------------------------------------------
// exits database
//
void DBExit()
    {
    dbexit();
    } // DBExit
//
//
//------------------------------SQLErrHandler----------------------------------------
// SQL Server Error handler code
//
int SQLErrHandler(DBPROCESS* pDbproc, int severity, int err, int oserr,
                  char* dberrstr, char* oserrstr)
    {
    char msg[512];

    sprintf(msg, "DB-Library: (%ld) : %s\n", err, dberrstr);
    WriteErrorLog(msg);

    if (oserr != DBNOERR)
        {
        sprintf(msg, "OS Error: (%ld) : %s\n", oserr, oserrstr);
        WriteErrorLog(msg);
        }

    // Place error in thread's data space for retrieval in calling function
    // If there already is an error for this call leave it
    if (*((int *) dbgetuserdata(pDbproc)) == 0)
        *((int *) dbgetuserdata(pDbproc)) = err;

    return (INT_CANCEL); // Sets ret code of calling dbfunc to FAIL

    } // SQLErrHandler
//
//
//------------------------------SQLMsgHandler----------------------------------------
// SQL Server message handler code
//
int SQLMsgHandler(DBPROCESS* pDbproc, DBINT msgno, int msgstate, int severity,
                  char *msgtext)
    {
    char msg[256];

    // Ignore informational messages like 5701 (changed database context)
    if ( (msgno == 5701) || (msgno == 2528) || (msgno == 5703) || (msgno == 6006) )
        {
        return(0);
        }

    // Write error to error log
    sprintf(msg, "SQL Server Message: (%ld) : %s\n", msgno, msgtext);
    WriteErrorLog(msg);

    // Place error in thread's data space for retrieval in calling function
    // If there already is an error for this call leave it
    if (*((int *) dbgetuserdata(pDbproc)) == 0)
        *((int *) dbgetuserdata(pDbproc)) = msgno;

    // Return to calling function
    return(0);
    } // SQLMsgHandler
//
//
//------------------------------DBOpenConnection-------------------------------------
// opens database connection
//
void DBOpenConnection(DBPROCESS** ppDbproc, char* server, char* database, char* user,
                      char* password, char* app, int* spid, long* pack_size)
    {
    LOGINREC    *login;

    login = dblogin();

    DBSETLUSER(login, user);
    DBSETLPWD(login,  password);
    DBSETLHOST(login, app);
    DBSETLPACKET(login, (unsigned short) pack_size);

    if ((*ppDbproc = dbopen(login, server )) == NULL)
        {
        WriteErrorLog("DBOpenConnection: Could not open connection, exiting");
        exit(-1);
        }

    // Use the the right database
    dbuse(*ppDbproc, database);
    // Set up thread storage area to pass error codes from SQLMsgHandler
    dbsetuserdata(*ppDbproc, malloc(sizeof(int)));
    *((int *) dbgetuserdata(*ppDbproc)) = 0;

    // Return process id
    dbcmd(*ppDbproc, "select @@spid");
    dbsqlexec(*ppDbproc);
    while (dbresults(*ppDbproc) != NO_MORE_RESULTS)
        {
        dbbind(*ppDbproc, 1, SMALLBIND, (DBINT) 0, (BYTE *) spid);
        while (dbnextrow(*ppDbproc) != NO_MORE_ROWS) ;
        }
    // Set SQL options
    // don't return number of rows affected by each SQL statement
    dbcmd(*ppDbproc, "set nocount on");
```

```
    dbsqlexec(*ppDbproc);
    while (dbresults(*ppDbproc) != NO_MORE_RESULTS)
        {
        while (dbnextrow(*ppDbproc) != NO_MORE_ROWS) ;
        }

    //rollback transaction on abort
    dbcmd(*ppDbproc, "set XACT_ABORT ON");
    dbsqlexec(*ppDbproc);
    while (dbresults(*ppDbproc) != NO_MORE_RESULTS)
        {
        while (dbnextrow(*ppDbproc) != NO_MORE_ROWS) ;
        }

    } // DBOpenConnection
//
//
//--------------------------------DBCloseConnection-----------------------------------
// closes database connection
//
void DBCloseConnection(DBPROCESS* pDbproc)
    {
    dbclose(pDbproc);
    } // DBCloseConnection
//
//
//--------------------------------DBNewOrder------------------------------------------
// Database calls for NewOrder transaction
//
int DBNewOrder(DBPROCESS* pDbproc, NEW_ORDER_DATA* pNewOrder)
    {
    RETCODE             rc;
    int                 tryit, i, len, sql_error_code;
    DBINT               commit_flag;
    char                printbuf[BUF_LEN];
    DBDATETIME          datetime;
    DBDATEREC           d;
    BYTE                *pData;

    pNewOrder->num_deadlocks = 0;

    for (tryit=0; tryit < DEADLOCK_RETRY; tryit++)
        {
        if (dbrpcinit(pDbproc, "tpcc_neworder", 0) == SUCCEED)
            {
            dbrpcparam(pDbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pNewOrder->w_id);
            dbrpcparam(pDbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pNewOrder->d_id);
            dbrpcparam(pDbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pNewOrder->c_id);
            dbrpcparam(pDbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *)
                &pNewOrder->o_ol_cnt);
            dbrpcparam(pDbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *)
                &pNewOrder->o_all_local);

            for (i = 0; i < pNewOrder->o_ol_cnt; i++)
                {
                dbrpcparam(pDbproc, NULL, 0, SQLINT4, -1, -1,
                    (BYTE *) &pNewOrder->Ol[i].ol_i_id);
                dbrpcparam(pDbproc, NULL, 0, SQLINT2, -1, -1,
                    (BYTE *) &pNewOrder->Ol[i].ol_supply_w_id);
                dbrpcparam(pDbproc, NULL, 0, SQLINT2, -1, -1,
                    (BYTE *) &pNewOrder->Ol[i].ol_quantity);
                }

            if (dbrpcexec(pDbproc) == SUCCEED)
                {
                pNewOrder->total_amount = 0.0;

                // Get results from order line

                for (i = 0; i<pNewOrder->o_ol_cnt; i++)
                    {
                    if (((rc = dbresults(pDbproc)) != NO_MORE_RESULTS) &&(rc != FAIL))
                        {
                        if (DBROWS(pDbproc) && (dbnumcols(pDbproc) == 5))
                            {
                            while (dbnextrow(pDbproc) != NO_MORE_ROWS)
                                {
                                if(pData=dbdata(pDbproc, 1))
                                    {
                                    len = dbdatlen(pDbproc, 1);
                                    strncpy(pNewOrder->Ol[i].ol_i_name,
                                        (const char*) pData, len);
                                    pNewOrder->Ol[i].ol_i_name[len] = '\0';
                                    }
                                if(pData=dbdata(pDbproc, 2))
                                    pNewOrder->Ol[i].ol_stock =(*(DBSMALLINT *)pData);
                                if(pData=dbdata(pDbproc, 3))
                                    pNewOrder->Ol[i].ol_brand_generic = *pData;
                                if(pData=dbdata(pDbproc, 4))
                                    pNewOrder->Ol[i].ol_i_price = (*(DBFLT8 *) pData);
                                if(pData=dbdata(pDbproc, 5))
                                    pNewOrder->Ol[i].ol_amount = (*(DBFLT8 *) pData);

                                pNewOrder->total_amount += pNewOrder->Ol[i].ol_amount;
                                } // end while
                            } // end if (DBROWS
                        } // end if (((rc
                    } // end for (i=0

                while (((rc = dbresults(pDbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
                    {
                    if (DBROWS(pDbproc) && (dbnumcols(pDbproc) == 8))
                        {
                        while (((rc = dbnextrow(pDbproc)) != NO_MORE_ROWS) &&
                            (rc != FAIL))
                            {
                            if(pData=dbdata(pDbproc, 1))
                                pNewOrder->w_tax = (*(DBFLT8 *) pData);
                            if(pData=dbdata(pDbproc, 2))
                                pNewOrder->d_tax = (*(DBFLT8 *) pData);
                            if(pData=dbdata(pDbproc, 3))
                                pNewOrder->o_id = (*(DBINT *) pData);
                            if(pData=dbdata(pDbproc, 4))
                                {
                                len = dbdatlen(pDbproc, 4);
                                strncpy(pNewOrder->c_last, (const char*) pData, len);
                                pNewOrder->c_last[len] = '\0';
                                }
                            if(pData=dbdata(pDbproc, 5))
                                pNewOrder->c_discount = (*(DBFLT8 *) pData);
                            if(pData=dbdata(pDbproc, 6))
                                {
                                len = dbdatlen(pDbproc, 6);
                                strncpy(pNewOrder->c_credit, (const char*) pData,len);
                                pNewOrder->c_credit[len] = '\0';
                                }
                            if(pData=dbdata(pDbproc, 7))
                                {
                                datetime = *((DBDATETIME *) pData);
                                dbdatecrack(pDbproc, &d, &datetime);
                                sprintf(pNewOrder->o_entry_d,
                                    "%02d-%02d-%4d %02d:%02d:%02d",
                                    d.day, d.month, d.year, d.hour,d.minute,d.second);
                                }
                            if(pData=dbdata(pDbproc, 8))
                                commit_flag = (*(DBTINYINT *) pData);
                            } // end while
                        } // end if (DBROWS
                    } // end if (((rc = dbresults
                } // end if (dbrpcexec
            } // end if (dbrpcinit
```

```
        sql_error_code =  0;
        sql_error_code =  *((int *) dbgetuserdata(pDbproc));
        *((int *) dbgetuserdata(pDbproc)) = 0; //reset thread's error code

        if (sql_error_code == 0) return (commit_flag == 1) ? SUCCESS : INVALID_ITEM;
        if (sql_error_code != 1205) return SQL_ERROR;

        // Note deadlock (1205 code) and retry
        sprintf(printbuf,"DBNewOrder:deadlock: retry: %d",++pNewOrder->num_deadlocks);
        WriteErrorLog(printbuf);
        Sleep(DEADLOCK_WAIT*tryit);
        } // end for (tryit

    // If we reached here, it means we quit after MAX_RETRY deadlocks
    WriteErrorLog("DBNewOrder: deadlock max retry reached!");

    return DEADLOCK;
    } // DBNewOrder
//
//
//-------------------------------DBPayment-----------------------------------------
// Database calls for Payment transaction
//
int DBPayment(DBPROCESS* pDbproc, PAYMENT_DATA* pPayment)
    {
    RETCODE             rc;
    int                 tryit, len, sql_error_code;
    char                printbuf[BUF_LEN];
    BOOL                by_name = FALSE;
    DBDATETIME          datetime;
    DBDATEREC           d;
    BYTE                *pData;

    pPayment->num_deadlocks = 0;

    if (pPayment->c_id == 0) by_name = TRUE;

    for (tryit=0; tryit < DEADLOCK_RETRY; tryit++)
        {
        if (dbrpcinit(pDbproc, "tpcc_payment", 0) == SUCCEED)
            {
            dbrpcparam(pDbproc, NULL, 0, SQLINT2, -1, -1,(BYTE*) &pPayment->w_id);
            dbrpcparam(pDbproc, NULL, 0, SQLINT2, -1, -1,(BYTE*) &pPayment->c_w_id);
            dbrpcparam(pDbproc, NULL, 0, SQLFLT8, -1, -1,(BYTE*) &pPayment->h_amount);
            dbrpcparam(pDbproc, NULL, 0, SQLINT1, -1, -1,(BYTE*) &pPayment->d_id);
            dbrpcparam(pDbproc, NULL, 0, SQLINT1, -1, -1,(BYTE*) &pPayment->c_d_id);
            dbrpcparam(pDbproc, NULL, 0, SQLINT4, -1, -1,(BYTE*) &pPayment->c_id);

            if (by_name)
                dbrpcparam(pDbproc, NULL, 0, SQLCHAR, -1, strlen(pPayment->c_last),
                    (unsigned char *const) pPayment->c_last);

            if (dbrpcexec(pDbproc) == SUCCEED)
                {
                while (((rc = dbresults(pDbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
                    {
                    if (DBROWS(pDbproc) && (dbnumcols(pDbproc) == 27))
                        {
                        while (((rc = dbnextrow(pDbproc)) != NO_MORE_ROWS) &&
                            (rc != FAIL))
                            {
                            if(pData=dbdata(pDbproc, 1))
                                pPayment->c_id = *((DBINT *) pData);

                            if(pData=dbdata(pDbproc, 2))
                                {
                                len = dbdatlen(pDbproc, 2);
                                strncpy(pPayment->c_last, (const char*)pData, len);
                                pPayment->c_last[len] = '\0';
                                }
```

```
if(pData=dbdata(pDbproc, 3))
    {
    datetime = *((DBDATETIME *) pData);
    dbdatecrack(pDbproc, &d, &datetime);
    sprintf(pPayment->h_date,
        "%02d-%02d-%4d %02d:%02d:%02d",
        d.day, d.month, d.year, d.hour,d.minute,d.second);
    }

if(pData=dbdata(pDbproc, 4))
    {
    len = dbdatlen(pDbproc, 4);
    strncpy(pPayment->w_street_1, (const char*)pData,len);
    pPayment->w_street_1[len] = '\0';
    }

if(pData=dbdata(pDbproc, 5))
    {
    len = dbdatlen(pDbproc, 5);
    strncpy(pPayment->w_street_2, (const char*)pData,len);
    pPayment->w_street_2[len] = '\0';
    }

if(pData=dbdata(pDbproc, 6))
    {
    len = dbdatlen(pDbproc, 6);
    strncpy(pPayment->w_city, (const char*)pData, len);
    pPayment->w_city[len] = '\0';
    }

if(pData=dbdata(pDbproc, 7))
    {
    len = dbdatlen(pDbproc, 7);
    strncpy(pPayment->w_state, (const char*)pData, len);
    pPayment->w_state[len] = '\0';
    }

if(pData=dbdata(pDbproc, 8))
    {
    len = dbdatlen(pDbproc, 8);
    strncpy(pPayment->w_zip, (const char*)pData, len);
    pPayment->w_zip[len] = '\0';
    }

if(pData=dbdata(pDbproc, 9))
    {
    len = dbdatlen(pDbproc, 9);
    strncpy(pPayment->d_street_1, (const char*)pData,len);
    pPayment->d_street_1[len] = '\0';
    }

if(pData=dbdata(pDbproc, 10))
    {
    len = dbdatlen(pDbproc, 10);
    strncpy(pPayment->d_street_2, (const char*)pData,len);
    pPayment->d_street_2[len] = '\0';
    }

if(pData=dbdata(pDbproc, 11))
    {
    len = dbdatlen(pDbproc, 11);
    strncpy(pPayment->d_city, (const char*)pData, len);
    pPayment->d_city[len] = '\0';
    }

if(pData=dbdata(pDbproc, 12))
    {
    len = dbdatlen(pDbproc, 12);
    strncpy(pPayment->d_state, (const char*)pData, len);
    pPayment->d_state[len] = '\0';
    }
```

```
if(pData=dbdata(pDbproc, 13))
    {
    len = dbdatlen(pDbproc, 13);
    strncpy(pPayment->d_zip, (const char*)pData, len);
    pPayment->d_zip[len] = '\0';
    }

if(pData=dbdata(pDbproc, 14))
    {
    len = dbdatlen(pDbproc, 14);
    strncpy(pPayment->c_first, (const char*)pData, len);
    pPayment->c_first[len] = '\0';
    }

if(pData=dbdata(pDbproc, 15))
    {
    len = dbdatlen(pDbproc, 15);
    strncpy(pPayment->c_middle, (const char*)pData, len);
    pPayment->c_middle[len] = '\0';
    }

if(pData=dbdata(pDbproc, 16))
    {
    len = dbdatlen(pDbproc, 16);
    strncpy(pPayment->c_street_1, (const char*)pData,len);
    pPayment->c_street_1[len] = '\0';
    }

if(pData=dbdata(pDbproc, 17))
    {
    len = dbdatlen(pDbproc, 17);
    strncpy(pPayment->c_street_2, (const char*)pData,len);
    pPayment->c_street_2[len] = '\0';
    }

if(pData=dbdata(pDbproc, 18))
    {
    len = dbdatlen(pDbproc, 18);
    strncpy(pPayment->c_city, (const char*)pData, len);
    pPayment->c_city[len] = '\0';
    }

if(pData=dbdata(pDbproc, 19))
    {
    len = dbdatlen(pDbproc, 19);
    strncpy(pPayment->c_state, (const char*)pData, len);
    pPayment->c_state[len] = '\0';
    }

if(pData=dbdata(pDbproc, 20))
    {
    len = dbdatlen(pDbproc, 20);
    strncpy(pPayment->c_zip, (const char*)pData, len);
    pPayment->c_zip[len] = '\0';
    }

if(pData=dbdata(pDbproc, 21))
    {
    len = dbdatlen(pDbproc, 21);
    strncpy(pPayment->c_phone, (const char*)pData, len);
    pPayment->c_phone[len] = '\0';
    }

if(pData=dbdata(pDbproc, 22))
    {
    datetime = *((DBDATETIME *) pData);
    dbdatecrack(pDbproc, &d, &datetime);
    sprintf(pPayment->c_since, "%02d-%02d-%4d",
        d.day, d.month, d.year);
    }

if(pData=dbdata(pDbproc, 23))
    {
    len = dbdatlen(pDbproc, 23);
    strncpy(pPayment->c_credit, (const char*)pData, len);
    pPayment->c_credit[len] = '\0';
    }

if(pData=dbdata(pDbproc, 24))
    pPayment->c_credit_lim = (*(DBFLT8 *) pData);

if(pData=dbdata(pDbproc, 25))
    pPayment->c_discount = (*(DBFLT8 *) pData);

if(pData=dbdata(pDbproc, 26))
    pPayment->c_balance = (*(DBFLT8 *) pData);

if(pData=dbdata(pDbproc, 27))
    {
    len = dbdatlen(pDbproc, 27);
    strncpy(pPayment->c_data, (const char*)pData, len);
    pPayment->c_data[len] = '\0';
    }
} // end while (((rc = dbnextrow
} // end if (DBROWS
} // end while (((rc = dbresults
} // end if (dbrpcexec
} // end if (dbrpcinit

sql_error_code =  0;
sql_error_code =  *((int *) dbgetuserdata(pDbproc));
*((int *) dbgetuserdata(pDbproc)) = 0; //reset thread's error code

if (sql_error_code == 0) return  SUCCESS;
if (sql_error_code != 1205) return SQL_ERROR;

// Note deadlock (1205 code) and retry
sprintf(printbuf,"DBPayment: deadlock: retry: %d", ++pPayment->num_deadlocks);
WriteErrorLog(printbuf);
Sleep(DEADLOCK_WAIT*tryit);
} // end for (tryit

// If we reached here, it means we quit after MAX_RETRY deadlocks
WriteErrorLog("DBPayment: deadlock max retry reached!");

return DEADLOCK;
} // DBPayment
//
//
//--------------------------------DBOrderStatus---------------------------------------
// Database calls for OrderStatus transaction
//
int DBOrderStatus(DBPROCESS* pDbproc, ORDER_STATUS_DATA* pOrderStatus)
    {
    RETCODE             rc;
    int                 tryit, i, len, sql_error_code;
    char                printbuf[BUF_LEN];
    BOOL                by_name = FALSE;
    DBDATETIME          datetime;
    DBDATEREC           d;
    BYTE                *pData;

    pOrderStatus->num_deadlocks = 0;

    if (pOrderStatus->c_id == 0) by_name = TRUE;

    for (tryit=0; tryit < DEADLOCK_RETRY; tryit++)
        {
        if (dbrpcinit(pDbproc, "tpcc_orderstatus", 0) == SUCCEED)
            {
            dbrpcparam(pDbproc, NULL, 0, SQLINT2, -1, -1,(BYTE*) &pOrderStatus->w_id);
```

```
dbrpcparam(pDbproc, NULL, 0, SQLINT1, -1, -1,(BYTE*) &pOrderStatus->d_id);
dbrpcparam(pDbproc, NULL, 0, SQLINT4, -1, -1,(BYTE*) &pOrderStatus->c_id);

if (by_name)
    dbrpcparam(pDbproc, NULL, 0, SQLCHAR, -1,strlen(pOrderStatus->c_last),
        (unsigned char *const)pOrderStatus->c_last);
if (dbrpcexec(pDbproc) == SUCCEED)
    {
    while (((rc = dbresults(pDbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
        {
        if (DBROWS(pDbproc) && (dbnumcols(pDbproc) == 5))
            {
            i=0;

            while (((rc = dbnextrow(pDbproc)) != NO_MORE_ROWS) &&
                (rc !=FAIL))
                {
                if(pData=dbdata(pDbproc, 1))
                    pOrderStatus->OlOrderStatusData[i].ol_supply_w_id =
                        (*(DBSMALLINT *) pData);

                if(pData=dbdata(pDbproc, 2))
                    pOrderStatus->OlOrderStatusData[i].ol_i_id =
                        (*(DBINT *) pData);

                if(pData=dbdata(pDbproc, 3))
                    pOrderStatus->OlOrderStatusData[i].ol_quantity =
                        (*(DBSMALLINT *) pData);

                if(pData=dbdata(pDbproc, 4))
                    pOrderStatus->OlOrderStatusData[i].ol_amount =
                        (*(DBFLT8 *) pData);

                if(pData=dbdata(pDbproc, 5))
                    {
                    datetime = *((DBDATETIME *) pData);
                    dbdatecrack(pDbproc, &d, &datetime);
                    sprintf(pOrderStatus->
                        OlOrderStatusData[i].ol_delivery_d,
                        "%02d-%02d-%4d", d.day, d.month, d.year);
                    }

                i++;
                } // end while (((rc = dbnextrow

            pOrderStatus->o_ol_cnt = i;
            } // end if (DBROWS

        else if (DBROWS(pDbproc) && (dbnumcols(pDbproc) == 8))
            {
            while (((rc = dbnextrow(pDbproc)) != NO_MORE_ROWS) &&
                (rc !=FAIL))
                {
                if(pData=dbdata(pDbproc, 1))
                    pOrderStatus->c_id = (*(DBINT *) pData);

                if(pData=dbdata(pDbproc, 2))
                    {
                    len = dbdatlen(pDbproc, 2);
                    strncpy(pOrderStatus->c_last, (const char*)pData,len);
                    pOrderStatus->c_last[len] = '\0';
                    }

                if(pData=dbdata(pDbproc, 3))
                    {
                    len = dbdatlen(pDbproc, 3);
                    strncpy(pOrderStatus->c_first,(const char*)pData,len);
                    pOrderStatus->c_first[len] = '\0';
                    }

                if(pData=dbdata(pDbproc, 4))
                    {
                    len = dbdatlen(pDbproc, 4);
                    strncpy(pOrderStatus->c_middle, (const char*)pData,
                        len);
                    pOrderStatus->c_middle[len] = '\0';
                    }

                if(pData=dbdata(pDbproc, 5))
                    {
                    datetime = *((DBDATETIME *) pData);
                    dbdatecrack(pDbproc, &d, &datetime);
                    sprintf(pOrderStatus->o_entry_d,
                        "%02d-%02d-%4d %02d:%02d:%02d",
                        d.day, d.month, d.year, d.hour,d.minute,d.second);
                    }

                if(pData=dbdata(pDbproc, 6))
                    pOrderStatus->o_carrier_id = (*(DBSMALLINT *) pData);

                if(pData=dbdata(pDbproc, 7))
                    pOrderStatus->c_balance = (*(DBFLT8 *) pData);

                if(pData=dbdata(pDbproc, 8))
                    pOrderStatus->o_id = (*(DBINT *) pData);

                } // end while (((rc = dbnextrow
            } // end else if (DBROWS
        } // end while (((rc = dbresults
    } // end if (dbrpcexec
} // end if(dbrpcinit

sql_error_code =  0;
sql_error_code =  *((int *) dbgetuserdata(pDbproc));
*((int *) dbgetuserdata(pDbproc)) = 0; //reset thread's error code

if (sql_error_code == 0) return  SUCCESS;
if (sql_error_code != 1205) return SQL_ERROR;

// Note deadlock (1205 code) and retry
sprintf(printbuf,"DBOrderStatus: deadlock: retry: %d",
    ++pOrderStatus->num_deadlocks);
WriteErrorLog(printbuf);
Sleep(DEADLOCK_WAIT*tryit);
} // end for (tryit

// If we reached here, it means we quit after MAX_RETRY deadlocks
WriteErrorLog("DBOrderStatus: deadlock max retry reached!");

return DEADLOCK;
} // DBOrderStatus
//
//
//---------------------------------DBDelivery-----------------------------------------
// Database calls for Delivery transaction
//
int DBDelivery(DBPROCESS* pDbproc, DELIVERY_DATA* pDel)
    {
    int                 i, rc, sql_error_code, iteration = 0;
    BOOL                deadlocked = FALSE;
    char                printbuf[255];
    BYTE                *pData;
    char                d[10];
    char                t[10];
    struct _timeb       tb;
    int                 elapsed_time;
    LONGLONG            current_time;  // milliseconds since 1/1/70

    do
        {
        if (dbrpcinit(pDbproc, "tpcc_delivery", 0) == SUCCEED)
            {
```

```
            dbrpcparam(pDbproc, NULL, 0, SQLINT2, -1, -1, (BYTE*) &pDel->w_id);
            dbrpcparam(pDbproc, NULL, 0, SQLINT1, -1, -1, (BYTE*)&pDel->o_carrier_id);

            if (dbrpcexec(pDbproc) == SUCCEED)
                {
                while (((rc = dbresults(pDbproc)) != NO_MORE_RESULTS)
                    && (rc != FAIL))
                    {
                    while (((rc = dbnextrow(pDbproc)) != NO_MORE_ROWS)
                           && (rc != FAIL))
                        {
                        for (i=0;i<10;i++)
                            {
                            if(pData=dbdata(pDbproc, i+1))
                                pDel->o_id[i] = *((DBINT *) pData);
                            }
                        } // end while (((rc = dbnextrow
                    } // end while (((rc = dbresults
                } // end if (dbrpcexec
            } // end if (dbrpcinit

        sql_error_code =  0;
        sql_error_code =  *((int *) dbgetuserdata(pDbproc));
        *((int *) dbgetuserdata(pDbproc)) = 0; //reset thread's error code

        if (sql_error_code == 0) break;

        deadlocked = TRUE;

        sprintf(printbuf,"DBDelivery: error %d\n", sql_error_code);
        WriteErrorLog(printbuf);
        Sleep(DEADLOCK_WAIT*++iteration);
        } while (deadlocked);


    // Read current time in a few formats and write record to delivery log
    // Each record:
    // today's date (mm/dd/yy), time now (hh:mm:ss.sss), queued_time (msec since
    //    1/1/70), time now (msec since 1/1/70), elapsed time (msec),
    //    home warehouse, carrier, last o_id for district 1, ... last o_id for
    //    district 10
    _strdate(d);
    _strtime(t);
    _ftime(&tb);

    current_time = (LONGLONG) 1000*tb.time + tb.millitm; // milliseconds since 1/1/70
    elapsed_time = (int) (current_time - pDel->queued_time);

    sprintf(printbuf,
        "%s %s.%03u %I64d %I64d %d %d %d %d %d %d %d %d %d %d %d %d\n",
        d, t, tb.millitm, pDel->queued_time, current_time, elapsed_time,
        pDel->w_id, pDel->o_carrier_id,pDel->o_id[0],pDel->o_id[1],pDel->o_id[2],
        pDel->o_id[3],pDel->o_id[4],pDel->o_id[5],pDel->o_id[6],pDel->o_id[7],
        pDel->o_id[8],pDel->o_id[9]);

    EnterCriticalSection(&deliv_write_crit_sec);
    fprintf(fp_delivlog,printbuf);
    fflush(fp_delivlog);
    LeaveCriticalSection(&deliv_write_crit_sec);

    return SUCCESS;
    } // DBDelivery
//
//
//--------------------------------DBStockLevel-----------------------------------------
// Database calls for StockLevel transaction
//
int DBStockLevel(DBPROCESS* pDbproc, STOCK_LEVEL_DATA* pSL)
    {
    RETCODE             rc;
    int                 tryit, sql_error_code;
    BOOL                not_done = TRUE;
    char                printbuf[255];
    BYTE                *pData;

    pSL->num_deadlocks = 0;

    for (tryit=0; tryit < DEADLOCK_RETRY; tryit++)
        {
        if (dbrpcinit(pDbproc, "tpcc_stocklevel", 0) == SUCCEED)
            {
            dbrpcparam(pDbproc, NULL, 0, SQLINT2, -1, -1, (BYTE*) &pSL->w_id);
            dbrpcparam(pDbproc, NULL, 0, SQLINT1, -1, -1, (BYTE*) &pSL->d_id);
            dbrpcparam(pDbproc, NULL, 0, SQLINT2, -1, -1, (BYTE*) &pSL->threshold);

            if (dbrpcexec(pDbproc) == SUCCEED)
                {
                while (((rc = dbresults(pDbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
                    {
                    if (DBROWS(pDbproc))
                        {
                        while (((rc = dbnextrow(pDbproc)) != NO_MORE_ROWS) &&
                            (rc != FAIL))
                            {
                            if(pData=dbdata(pDbproc, 1))
                                pSL->low_stock = *((long *) pData);
                            } // end while (((rc = dbnextrow
                        } // end if (DBROWS
                    } // end while (((rc = dbresults
                } // end if (dbrpcexec
            } // end if (dbrpcinit

        sql_error_code =  0;
        sql_error_code =  *((int *) dbgetuserdata(pDbproc));
        *((int *) dbgetuserdata(pDbproc)) = 0; //reset thread's error code

        if (sql_error_code == 0) return  SUCCESS;
        if (sql_error_code != 1205) return SQL_ERROR;

        // Note deadlock (1205 code) and retry
        sprintf(printbuf,"DBStockLevel: deadlock: retry: %d", ++pSL->num_deadlocks);
        WriteErrorLog(printbuf);
        Sleep(DEADLOCK_WAIT*tryit);
        } // end for (tryit

    // If we reached here, it means we quit after MAX_RETRY deadlocks
    WriteErrorLog("DBStockLevel: deadlock max retry reached!");

    return DEADLOCK;
    } // DBStockLevel
//
//
//
//--------------------------------End of sqlfuncs.c-----------------------------------
```

Appendix A – Application Source Code

Commands for compiling and linking Tuxedo servers

```
rem   tpcctuxservmak.cmd: run this from NT command prompt to compile Tuxedo Transaction
Monitor servers of type TMDL (Delivery)

cl.exe /nologo /ML /W3 /O2 /I c:\tuxedo\include /I c:\mssql\dblib\include /D WIN32 /D
    _CONSOLE /D _MBCS /D _TUX /D _TUX_D /D _TMSTHREADS /c tmserver_stub.c tmserver.c
link.exe  kernel32.lib user32.lib advapi32.lib ntwdblib.lib libtux.lib libbuft.lib
    libtux2.lib libgp.lib /nologo /subsystem:console /incremental:no /machine:I386
    /out:TMDL.exe /libpath:c:\tuxedo\lib /libpath:c:\mssql\dblib\lib tmserver.obj
    tmserver_stub.obj
erase *.obj


rem   tpcctuxt4mak.cmd: run this from NT command prompt to compile Tuxedo Transaction
Monitor servers of type TMT4 (NO,PY,OS,SL)

cl.exe /nologo /ML /W3 /O2 /I c:\tuxedo\include /I c:\mssql\dblib\include /D WIN32 /D
    _CONSOLE /D _MBCS /D _TUX /D _4_TRANS /D _TUX_NO /D _TUX_PY /D _TUX_OS /D _TUX_SL /D
    _TMSTHREADS /c tmserver_stub.c tmserver.c
link.exe  kernel32.lib user32.lib advapi32.lib ntwdblib.lib libtux.lib libbuft.lib
    libtux2.lib libgp.lib /nologo /subsystem:console /incremental:no /machine:I386
    /out:TMT4.exe /libpath:\tuxedo\lib /libpath:\mssql\dblib\lib tmserver.obj
    tmserver_stub.obj
erase *.obj
```

Appendix B - Database Design

Build Scripts

setup.cmd

```
ECHO OFF

@ECHO ******************************************************************************
@ECHO *                                                                            *
@ECHO *  Microsoft TPC-C Benchmark Kit Ver. 4.01                                    *
@ECHO *                                                                            *
@ECHO ******************************************************************************

if '%1'==''  goto usage
if '%2'==''  goto usage
if '%3'==''  goto usage
if '%4'==''  goto usage
if not '%5'==''  if not '%5' == 'scaled' goto usage

::Cleanup any old .err files
@if exist logs\*.err del logs\*.err
          >nul

if  '%3'=='full' goto start
if  '%3'=='builddb' goto builddb
if  '%3'=='objects' goto objects
if  '%3'=='bulkload' goto bulkload
if  '%3'=='objectsfull' goto objects
if  '%3'=='bulkloadfull' goto bulkload
if  '%3'=='backup' goto backup
goto usage

:start
:: Cleanup the logs directory...
@if exist logs\version.log del logs\version.log                              >nul
@if exist logs\db.log del logs\db.log                                        >nul
@if exist logs\objects.log del logs\objects.log                              >nul
@if exist logs\objects.log del logs\objects.log                              >nul
@if exist logs\bulkload.log del logs\bulkload.log                    >nul
@if exist logs\backup.log del logs\backup.log                                >nul

isql -Usa -P -S%1 -Q"select @@version"                                       >
logs\version.log
isql -Usa -P -S%1 -Q"select getdate()"                                       >>
logs\version.log

:builddb
@if exist logs\db.log del logs\db.log                                        >nul
@ECHO Building database files and database...
isql -Usa -P -S%1 -e < scripts\%2.war\%4\createdb.sql                        >
logs\db.log
@ECHO Database build complete.
if  '%3'=='full' goto objects
goto end

:objects
@if exist logs\objects.log del logs\objects.log                              >nul
@ECHO Creating database objects...
isql -Usa -P -S%1 -e < scripts\ddl\%4\tables.sql                     >  logs\objects.log
isql -Usa -P -S%1 -e < scripts\dml\%4\neword.sql                     >> logs\objects.log
isql -Usa -P -S%1 -e < scripts\dml\%4\payment.sql                    >> logs\objects.log
isql -Usa -P -S%1 -e < scripts\dml\%4\ordstat.sql                    >> logs\objects.log
isql -Usa -P -S%1 -e < scripts\dml\%4\delivery.sql                           >>
logs\objects.log
isql -Usa -P -S%1 -e < scripts\dml\%4\stocklev.sql                           >>
logs\objects.log
@ECHO Database object creation complete.
if  '%3'=='full' goto bulkload
if  '%3'=='objectsfull' goto bulkload
goto end

:bulkload
@if exist logs\bulkload.log del logs\bulkload.log                    >nul
@ECHO Beginning data load and index creation...
isql -Usa -P -S%1 -e < scripts\utility\%4\dbopt1.sql                         >>
logs\objects.log
if '%4'=='mssql70' goto odbc
if '%4'=='mssql65' goto dblib
goto usage
:dblib
if '%5'==''      loader\%4\bin\tpccldr -S%1 -W%2 -flogs\bulkload.log -dscripts\ddl\%4 -c0
if '%5'=='normal' loader\%4\bin\tpccldr -S%1 -W%2 -flogs\bulkload.log -dscripts\ddl\%4 -c0
if '%5'=='scaled' loader\%4\bin\tpccldr -S%1 -W%2 -flogs\bulkload.log -dscripts\ddl\%4 -c1
goto bulkloaddone
:odbc
if '%5'==''      loader\%4\bin\tpccldr -S%1 -W%2 -flogs\bulkload.log -dscripts\ddl\%4 -c0
if '%5'=='normal' loader\%4\bin\tpccldr -S%1 -W%2 -flogs\bulkload.log -dscripts\ddl\%4 -c0
if '%5'=='scaled' loader\%4\bin\tpccldr -S%1 -W%2 -flogs\bulkload.log -dscripts\ddl\%4 -c1
goto bulkloaddone
:bulkloaddone
isql -Usa -P -S%1 -e < scripts\utility\%4\dbopt2.sql                         >>
logs\bulkload.log
@ECHO Data load and index creation complete.
if  '%3'=='full' goto backup
if  '%3'=='objectsfull' goto backup
if  '%3'=='bulkloadfull' goto backup
goto end

:backup
@if exist logs\backup.log del logs\backup.log                                >nul
@ECHO Backing up database...
isql -Usa -P -S%1 -e < scripts\%2.war\%4\backup.sql                          >
logs\backup.log
@ECHO Database backup complete.
if  '%3'=='full' goto verifyload
if  '%3'=='objectsfull' goto verifyload
if  '%3'=='bulkloadfull' goto verifyload
goto complete

:verifyload
@if exist logs\verifyload.log del logs\verifyload.log                        >nul
@Echo Verifying TPC-C database load...
isql -Usa -P -S%1 < scripts\utility\%4\verifytpccload.sql                    >
logs\verifyload.log
@ECHO Check logs\verifyload.log to verify database load.

:complete
@ECHO ******************************************************************************
@ECHO *                                                                            *
@ECHO *  Full TPC-C build complete.  Check logs directory for setup errors.         *
@ECHO *                                                                            *
@ECHO ******************************************************************************

goto end

:usage
@ECHO ******************************************************************************
@ECHO *                                                                            *
@ECHO *  The TPC-C setup command file requires the following parameters:            *
@ECHO *                                                                            *
@ECHO *  setup SERVER NUMWAR BLDOPT VERSION DBTYPE                                  *
```

```
@ECHO *                                                                      *
@ECHO *      SERVER  = machine name of server (use "" for local server)      *
@ECHO *      NUMWAR  = number of warehouses                                  *
@ECHO *      BLDOPT  = full, builddb, objects, objectsfull, bulkload,        *
@ECHO *                bulkloadfull, or backup                               *
@ECHO *      VERSION = mssql65 or mssql70                                    *
@ECHO *      DBTYPE  = normal or scaled                                      *
@ECHO *                                                                      *
@ECHO *  Note #1: the BLDOPT and VERSION parameters are case sensitive.      *
@ECHO *                                                                      *
@ECHO *  Note #2: the DBTYPE is optional.  If no DBTYPE is specified, SETUP  *
@ECHO *           will default to NORMAL.                                    *
@ECHO *                                                                      *
@ECHO *  Example:                                                            *
@ECHO *                                                                      *
@ECHO *  The following command would be used to build a complete 200         *
@ECHO *  warehouse database on SQL Server 7.0 runnning on server \\myserver. *
@ECHO *                                                                      *
@ECHO *      SETUP myserver 200 full mssql70                                 *
@ECHO *                                                                      *
@ECHO *  Note, this command file does a backup of the database by default    *
@ECHO *  after the database build process is complete.  If you do not wish   *
@ECHO *  to make a backup (strongly discouraged), you must edit this file    *
@ECHO *  and comment that section out.  Also, if you need to run the dbcheck *
@ECHO *  and the dbtables scripts on the fresh database load for an audit,   *
@ECHO *  you must either run them manually or edit this file to include them.*
@ECHO *                                                                      *
@ECHO ************************************************************************

:end

echo on
```

createdb.sql

```
--  File:     CREATEDB.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates tpcc database and backup files
--  Script to build 1900 warehouse with 6 file groups (CS, Misc, and Backup) - dated
3/14/99
--        1900 warehouse - Config 1
--        1 Hendrix/1 Channel for Logs, all others for data
--                  AFA5 - Data, Logs, and Backups
--                  AFA0-4 - Data, blank, and Backups.
--        jpj. 3.14.99
--        Remove misc from AFA5 due to log drives there.
--        jpj.      3.25.99   striped log files across 4-channels, RAID10


use master
go

--  remove any existing database and backup files

exec sp_dbremove tpcc, dropdev
exec sp_dropdevice 'tpccback1'
exec sp_dropdevice 'tpccback2'
exec sp_dropdevice 'tpccback3'
exec sp_dropdevice 'tpccback4'
exec sp_dropdevice 'tpccback5'
exec sp_dropdevice 'tpccback6'
go

declare @startdate datetime
declare @enddate datetime

select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

--  create main database files

create database tpcc on
          (name="MSSQL70_tpcc_root",filename="D:\MSSQL70_tpcc_root.mdf",size=8MB,
FILEGROWTH=0)
log on
          (name="MSSQL70_tpcc_log1",filename="\\.\Q:",size=40000MB, FILEGROWTH=0)

--  create filegroups

alter database tpcc add filegroup MSSQL70_cs_fg
alter database tpcc add filegroup MSSQL70_misc_fg

--  add files to filegroups


alter database tpcc add file
          (name="MSSQL70_cs1",filename="\\.\E:",size=19200MB, FILEGROWTH=0),
          (name="MSSQL70_cs2",filename="\\.\G:",size=19200MB, FILEGROWTH=0),
          (name="MSSQL70_cs3",filename="\\.\I:",size=19200MB, FILEGROWTH=0),
          (name="MSSQL70_cs4",filename="\\.\K:",size=19200MB, FILEGROWTH=0),
          (name="MSSQL70_cs5",filename="\\.\M:",size=19200MB, FILEGROWTH=0),
          (name="MSSQL70_cs6",filename="\\.\O:",size=19200MB, FILEGROWTH=0)

to filegroup MSSQL70_cs_fg

alter database tpcc add file
          (name="MSSQL70_misc1",filename="\\.\H:",size=15360MB, FILEGROWTH=0),
          (name="MSSQL70_misc2",filename="\\.\J:",size=15360MB, FILEGROWTH=0),
          (name="MSSQL70_misc3",filename="\\.\L:",size=15360MB, FILEGROWTH=0),
          (name="MSSQL70_misc4",filename="\\.\N:",size=15360MB, FILEGROWTH=0),
          (name="MSSQL70_misc5",filename="\\.\P:",size=15360MB, FILEGROWTH=0)

to filegroup MSSQL70_misc_fg




select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)
go

--  create backup devices

exec sp_addumpdevice 'disk','tpccback1','T:\tpccback1.dmp'
exec sp_addumpdevice 'disk','tpccback2','U:\tpccback2.dmp'
exec sp_addumpdevice 'disk','tpccback3','V:\tpccback3.dmp'
exec sp_addumpdevice 'disk','tpccback4','W:\tpccback4.dmp'
exec sp_addumpdevice 'disk','tpccback5','X:\tpccback5.dmp'
exec sp_addumpdevice 'disk','tpccback6','Y:\tpccback6.dmp'
go
```

tables.sql

```
--  File:     TABLES.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates TPC-C tables


use tpcc
```

Appendix B – Database Design

```
go

if exists ( select name from sysobjects where name = 'warehouse' )
        drop table warehouse
go
create table warehouse
(
        w_id                                    smallint,
        w_name                                  char(10),
        w_street_1                              char(20),
        w_street_2                              char(20),
        w_city                                  char(20),
        w_state                                 char(2),
        w_zip                                   char(9),
        w_tax                                   numeric(4,4),
        w_ytd                                   numeric(12,2)
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'district' )
        drop table district
go
create table district
(
        d_id                                    tinyint,
        d_w_id                                  smallint,
        d_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_tax                                   numeric(4,4),
        d_ytd                                   numeric(12,2),
        d_next_o_id                             int
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'customer' )
        drop table customer
go
create table customer
(
        c_id                                    int,
        c_d_id                                  tinyint,
        c_w_id                                  smallint,
        c_first                                 char(16),
        c_middle                        char(2),
        c_last                                  char(16),
        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_discount                              numeric(4,4),
        c_balance                       numeric(12,2),
        c_ytd_payment                   numeric(12,2),
        c_payment_cnt                   smallint,
        c_delivery_cnt                  smallint,
        c_data                                  char(500)
) on MSSQL70_cs_fg
go

if exists ( select name from sysobjects where name = 'history' )
        drop table history
go
create table history
(
        h_c_id                                  int,
        h_c_d_id                        tinyint,
        h_c_w_id                        smallint,
        h_d_id                                  tinyint,
        h_w_id                                  smallint,
        h_date                                  datetime,
        h_amount                        numeric(6,2),
        h_data                                  char(24)
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'new_order' )
        drop table new_order
go
create table new_order
(
        no_o_id                                 int,
        no_d_id                                 tinyint,
        no_w_id                                 smallint
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'orders' )
        drop table orders
go
create table orders
(
        o_id                                    int,
        o_d_id                                  tinyint,
        o_w_id                                  smallint,
        o_c_id                                  int,
        o_entry_d                       datetime,
        o_carrier_id                    tinyint,
        o_ol_cnt                        tinyint,
        o_all_local                             tinyint
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'order_line' )
        drop table order_line
go
create table order_line
(
        ol_o_id                                 int,
        ol_d_id                                 tinyint,
        ol_w_id                                 smallint,
        ol_number                       tinyint,
        ol_i_id                                 int,
        ol_supply_w_id                  smallint,
        ol_delivery_d                   datetime,
        ol_quantity                             smallint,
        ol_amount                       numeric(6,2),
        ol_dist_info                    char(24)
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'item' )
        drop table item
go
create table item
(
        i_id                                    int,
        i_im_id                                 int,
        i_name                                  char(24),
        i_price                                 numeric(5,2),
        i_data                                  char(50)
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'stock' )
```

```
          drop table stock
go
create table stock
(
          s_i_id                                int,
          s_w_id                                smallint,
          s_quantity                            smallint,
          s_dist_01                     char(24),
          s_dist_02                     char(24),
          s_dist_03                     char(24),
          s_dist_04                     char(24),
          s_dist_05                     char(24),
          s_dist_06                     char(24),
          s_dist_07                     char(24),
          s_dist_08                     char(24),
          s_dist_09                     char(24),
          s_dist_10                     char(24),
          s_ytd                                 int,
          s_order_cnt                           smallint,
          s_remote_cnt                  smallint,
          s_data                                char(50)
) on MSSQL70_cs_fg
go
```

idxcuscl.sql

```
--  File:     IDXCUSCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on customer table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where name = 'customer_c1' )
          drop index customer.customer_c1

create unique clustered index customer_c1 on customer(c_w_id, c_d_id, c_id)
          on MSSQL70_cs_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

idxcusnc.sql

```
--  File:     IDXCUSNC.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates non-clustered index on customer table

use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where  name = 'customer_nc1' )
          drop index customer.customer_nc1

create unique nonclustered index customer_nc1 on customer(c_w_id, c_d_id, c_last, c_first,
c_id)
          on MSSQL70_cs_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

idxdiscl.sql

```
--  File:     IDXDISCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on district table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where name = 'district_c1' )
          drop index district.district_c1

create unique clustered index  district_c1 on district(d_w_id, d_id)
          with fillfactor=100 on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

idxitmcl.sql

```
--  File:     IDXITMCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on item table

use tpcc
go
```

```
declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where  name = 'item_c1' )
          drop index item.item_c1

create unique clustered index item_c1 on item(i_id)
          on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

idxnodcl.sql

```
--  File:     IDXNODCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on new_order table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where  name = 'new_order_c1' )
          drop index new_order.new_order_c1

create unique clustered index new_order_c1 on new_order(no_w_id, no_d_id, no_o_id)
          on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

idxodlcl.sql

```
--  File:     IDXNODCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on new_order table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where  name = 'new_order_c1' )
          drop index new_order.new_order_c1

create unique clustered index new_order_c1 on new_order(no_w_id, no_d_id, no_o_id)
          on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

idxordcl.sql

```
--  File:     IDXORDCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on orders table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where  name = 'orders_c1' )
          drop index orders.orders_c1

create unique clustered index orders_c1 on orders(o_w_id, o_d_id, o_id)
          on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

idxstkcl.sql

```
--  File:     IDXSTKCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on stock table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where  name = 'stock_c1' )
          drop index stock.stock_c1

create unique clustered index stock_c1 on stock(s_i_id, s_w_id)
```

```
         on MSSQL70_cs_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

idxwarcl.sql

```
--  File:     IDXWARCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on warehouse table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where name = 'warehouse_c1' )
         drop index warehouse.warehouse_c1

create unique clustered index warehouse_c1 on warehouse(w_id)
         with fillfactor=100 on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

dbopt1.sql

```
--  File:     DBOPT1.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Sets database options for data load


use master
go

exec sp_dboption tpcc,'select into/bulkcopy',true
exec sp_dboption tpcc,'trunc. log on chkpt.',true
go

use tpcc
go

checkpoint
go
```

dbopt2.sql

```
--  File:     DBOPT2.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Resets database options after data load


use master
go

sp_dboption tpcc,'select ',false
go

sp_dboption tpcc,'trunc. ',false
go

use tpcc
go

checkpoint
go

sp_configure allow,1
go

reconfigure with override
go

/*                                                */
/* Set option values for user-defined indexes */
/*                                                */

sp_indexoption 'customer','AllowPageLocks',FALSE
go
sp_indexoption 'district','AllowPageLocks',FALSE
go
sp_indexoption 'warehouse','AllowPageLocks',FALSE
go
sp_indexoption 'stock','AllowPageLocks',FALSE
go
sp_indexoption 'order_line','AllowPageLocks',FALSE
go
sp_indexoption 'orders','AllowPageLocks',FALSE
go
sp_indexoption 'new_order','AllowRowLocks',FALSE
go
sp_indexoption 'item','AllowRowLocks',FALSE
go
sp_indexoption 'item','AllowPageLocks',FALSE
go

Print ' '
Print '******************'
Print 'Pre-specified Locking Hierarchy:'
Print '   Lockflag = 0 ==> No pre-pecified hierarchy'
Print '   Lockflag = 1 ==> Lock at Page-level then Table-level'
Print '   Lockflag = 2 ==> Lock at Row-level then Table-level'
Print '   Lockflag = 3 ==> Lock at Table-level'
Print ' '

select name,lockflags
from sysindexes
where object_id("warehouse")=id or
           object_id("district")=id or
           object_id("customer")=id or
           object_id("stock")=id or
           object_id("orders")=id or
           object_id("order_line")=id or
```

```
            object_id("history")=id or
            object_id("new_order")=id or
            object_id("item")=id
order by lockflags asc
go

sp_configure allow,0
go

reconfigure with override
go

exec sp_dboption tpcc, 'auto update statistics', FALSE
exec sp_dboption tpcc, 'auto create statistics', FALSE
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
```

dbopt3.sql

```
use tpcc
go
sp_indexoption 'orders','AllowPagelocks',TRUE
go
sp_indexoption 'orders','AllowRowlocks',FALSE
go
sp_indexoption 'order_line','AllowPagelocks',TRUE
go
sp_indexoption 'order_line','AllowRowlocks',FALSE
go
```

backup.sql

```
-- File:    BACKUP.SQL
--          Microsoft TPC-C Benchmark Kit Ver. 4.00
--          Copyright Microsoft, 1996
-- Purpose:  Creates backup of tpcc database


declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

dump database tpcc to tpccback1, tpccback2, tpccback3, tpccback4, tpccback5, tpccback6 with init, stats = 5

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

restore.sql

```
-- File:    RESTORE.SQL
--          Microsoft TPC-C Benchmark Kit Ver. 4.00
--          Copyright Microsoft, 1996
-- Purpose:  Loads database backup from backup files


declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

load database tpcc from tpccback1, tpccback2, tpccback3, tpccback4, tpccback5, tpccback6 with stats = 10, replace

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

Stored Procedures

neword.sql

```
--  File:     NEWORD.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.01
--            Copyright Microsoft, 1996
--  Purpose:  Creates new order transaction stored procedure
--
--  Modified 9/21/98 - Jamie Reding - Microsoft Corporation
--                  Reordered @rowcount check so that invalid supply warehouse id,
--      as well as invalid item id, is detected and causes explicit
--                  transaction rollback.
--

use tpcc
go

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_id                       int,
                    @commit_flag    tinyint,
        @li_id          int,
        @li_s_w_id      smallint,
        @li_qty         smallint,
                    @ol_number                  int,
                    @c_id_local                 int

begin

          begin transaction n

--  get district tax and next availible order id and update
--  plus initialize local variables

          update    district
          set                 @d_tax       = d_tax,
                              @o_id        = d_next_o_id,
                              d_next_o_id  = d_next_o_id + 1,
                              @o_entry_d   = getdate(),
                              @li_no       = 0,
                              @commit_flag = 1
          where     d_w_id       = @w_id and
                              d_id         = @d_id

--  process orderlines

    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,

                                 @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,

                                @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 repeatableread)
                    where     i_id = @li_id

--  update stock values

                    update    stock
                    set                 s_ytd        = s_ytd + @li_qty,
                                        @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

--  if there actually is a stock (and item) with these ids, go to work

                if (@@rowcount > 0)
                                    begin

--  insert order_line data (using data from item and stock)

                                                insert into order_line values(@o_id,

                                @d_id,

                                @w_id,

                                @li_no,

                                @li_id,

                                @li_s_w_id,

                                "dec 31, 1899",

                                @li_qty,

                                @i_price * @li_qty,

                                @s_dist)


--  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 (or stock) 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 (repeatableread)
                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 (repeatableread)
          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 Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates payment transaction stored procedure


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),
         @data           char(500),
         @c_data         char(500),
         @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 (repeatableread)
                    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 (repeatableread)
                    where c_last = @c_last and
                          c_w_id = @c_w_id and
                          c_d_id = @c_d_id
                    order by 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,
                @data          = c_data,
                @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 = 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(@data, 1, 458)

--  update customer info

                update customer set
                       c_data = @c_data
                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,1,200)
        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

```
--  File:     ORDSTAT.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates order status transaction stored procedure


use tpcc
go

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

```
create proc tpcc_orderstatus @w_id               smallint,
                                                                @d_id
         tinyint,
                                                                @c_id
         int,
                                                                @c_last  char(16) =
""

as

declare @c_balance            numeric(12,2),
        @c_first              char(16),
        @c_middle             char(2),
        @o_id                         int,
        @o_entry_d            datetime,
        @o_carrier_id         smallint,
        @cnt                          smallint

begin tran o

         if (@c_id = 0)
                  begin

--  get customer id and info using last name

                  select @cnt = (count(*)+1)/2
                  from customer (repeatableread)
                  where c_last = @c_last and
                    c_w_id = @w_id and
                    c_d_id = @d_id

                  set rowcount @cnt

                  select @c_id = c_id,
                               @c_balance = c_balance,
                               @c_first   = c_first,
                               @c_last    = c_last,
                               @c_middle  = c_middle
                  from customer (repeatableread)
                  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 (repeatableread)
                  where c_id   = @c_id and
                        c_d_id = @d_id and
                        c_w_id = @w_id

                  select @cnt = @@rowcount

                  end

--  if no such customer

         if (@cnt = 0)
         begin
                  raiserror("Customer not found",18,1)
                  goto custnotfound
         end

--  get order info

         select @o_id = o_id,
                        @o_entry_d    = o_entry_d,
            @o_carrier_id = o_carrier_id
         from orders (serializable)
         where o_c_id = @c_id and
         o_d_id = @d_id and
         o_w_id = @w_id
         order by o_id asc

--  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 (repeatableread)
         where ol_o_id = @o_id and
               ol_d_id = @d_id and
               ol_w_id = @w_id

custnotfound:

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 Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates delivery transaction stored procedure


use tpcc
go

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 top 1 @o_id = no_o_id
                    from new_order (serializable updlock)
        where no_w_id = @w_id and
                         no_d_id = @d_id
                    order by no_o_id asc

        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

-- return delivery data to client

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

go
```

stocklev.sql

```
--  File:     STOCKLEV.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates stock level transaction stored procedure


use tpcc
go

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
```

Loader Source Code

tpcc.h

```
//        File:               TPCC.H
//                                      Microsoft TPC-C Kit Ver. 4.00
//                                      Copyright Microsoft, 1996, 1997, 1998

//        Purpose:  Header file for TPC-C database loader


// Build number of TPC Benchmark Kit
#define TPCKIT_VER  "4.00"

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

// ODBC headers
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>

// General constants
#define MILLI                           1000
#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                          ""

// Default loader arguments
#define BATCH                           10000
#define DEFLDPACKSIZE                           32768
#define ORDERS_PER_DIST                 3000
#define LOADER_RES_FILE                   "logs\\load.out"
#define LOADER_NURAND_C                           123
#define DEF_STARTING_WAREHOUSE                  1
#define BUILD_INDEX                                     1         // build both data
and indexes
#define INDEX_ORDER                                     1         // build indexes
before load
#define SCALE_DOWN                                      0   // build a normal scale
database
#define INDEX_SCRIPT_PATH     "scripts"

typedef struct
{
    char                                        *server;
    char                                        *database;
    char                                        *user;
    char                                        *password;
          BOOL                                  tables_all;
          // set if loading all tables
          BOOL                                  table_item;
          // set if loading ITEM table specifically
          BOOL                                  table_warehouse;    // set if loading
WAREHOUSE, DISTRICT, and STOCK
          BOOL                                  table_customer;               // set if
loading CUSTOMER and HISTORY
          BOOL                                  table_orders;                 // set if
loading NEW-ORDER, ORDERS, ORDER-LINE
    long                                        num_warehouses;
    long                                        batch;
    long                                        verbose;
          long                                  pack_size;
          char                                  *loader_res_file;
          char                                  *synch_servername;
          long                                  case_sensitivity;
          long                                  starting_warehouse;
          long                                  build_index;
          long                                  index_order;
          long                                  scale_down;
          char                                  *index_script_path;
} TPCCLDR_ARGS;

// String length constants
          #define SERVER_NAME_LEN           20
          #define DATABASE_NAME_LEN         20
          #define USER_NAME_LEN             20
          #define PASSWORD_LEN              20
          #define TABLE_NAME_LEN            20
          #define I_DATA_LEN                50
          #define I_NAME_LEN                24
          #define BRAND_LEN                  1
          #define LAST_NAME_LEN             16
          #define W_NAME_LEN                10
          #define ADDRESS_LEN               20
          #define STATE_LEN                  2
          #define ZIP_LEN                    9
          #define S_DIST_LEN                24
          #define S_DATA_LEN                50
          #define D_NAME_LEN                10
          #define FIRST_NAME_LEN            16
          #define MIDDLE_NAME_LEN            2
          #define PHONE_LEN                 16
          #define CREDIT_LEN                 2
          #define C_DATA_LEN               500
          #define H_DATA_LEN                24
          #define DIST_INFO_LEN             24
          #define MAX_OL_NEW_ORDER_ITEMS    15
          #define MAX_OL_ORDER_STATUS_ITEMS 15
          #define STATUS_LEN                25
          #define OL_DIST_INFO_LEN                  24
          #define C_SINCE_LEN                                 23
          #define H_DATE_LEN                                  23
          #define OL_DELIVERY_D_LEN                 23
          #define O_ENTRY_D_LEN             23


// Functions in random.c
void      seed();
long      irand();
double    drand();
void      WUCreate();
```

```
short     WURand();
long      RandomNumber(long lower, long upper);

// Functions in getargs.c;
void      GetArgsLoader();
void      GetArgsLoaderUsage();

// Functions in time.c
long      TimeNow();

// Functions in strings.c
void      MakeAddress();
void      LastName();
int       MakeAlphaString();
int       MakeOriginalAlphaString();
int       MakeNumberString();
int       MakeZipNumberString();
void      InitString();
void      InitAddress();
void      PaddString();
```

tpccldr.c

```
//        File:               TPCCLDR.C
//                                    Microsoft TPC-C Kit Ver. 4.00
//                                    Copyright Microsoft, 1996, 1997, 1998
//        Purpose:  Source file for TPC-C database loader


// Includes
#include "tpcc.h"
#include "search.h"

// Defines
#define MAXITEMS                100000
#define MAXITEMS_SCALE_DOWN             100
#define CUSTOMERS_PER_DISTRICT  3000
#define CUSTOMERS_SCALE_DOWN  30
#define DISTRICT_PER_WAREHOUSE  10
#define ORDERS_PER_DISTRICT     3000
#define ORDERS_SCALE_DOWN       30
#define MAX_CUSTOMER_THREADS    2
#define MAX_ORDER_THREADS       3
#define MAX_MAIN_THREADS        4

// Functions declarations

void HandleErrorDBC (SQLHDBC  hdbc1);

long NURand();
void LoadItem();
void LoadWarehouse();

void Stock();
void District();

void LoadCustomer();
void CustomerBufInit();
void CustomerBufLoad();
void LoadCustomerTable();
void LoadHistoryTable();
void LoadOrders();
void OrdersBufInit();
void OrdersBufLoad();
void LoadOrdersTable();
void LoadNewOrderTable();
void LoadOrderLineTable();
void GetPermutation();
void CheckForCommit();
void OpenConnections();
void BuildIndex();
void FormatDate ();

// Shared memory structures

typedef struct
{
    long                ol;
    long                ol_i_id;
    short               ol_supply_w_id;
    short               ol_quantity;
    double              ol_amount;
    char                ol_dist_info[DIST_INFO_LEN+1];
          char                                    ol_delivery_d[OL_DELIVERY_D_LEN+1];
} ORDER_LINE_STRUCT;

typedef struct
{
    long                o_id;
    short               o_d_id;
    short               o_w_id;
    long                o_c_id;
    short               o_carrier_id;
    short               o_ol_cnt;
    short               o_all_local;
          ORDER_LINE_STRUCT   o_ol[15];
} ORDERS_STRUCT;

typedef struct
{
    long                                c_id;
    short                               c_d_id;
    short                               c_w_id;
          char                                      c_first[FIRST_NAME_LEN+1];
    char                                c_middle[MIDDLE_NAME_LEN+1];
    char                                c_last[LAST_NAME_LEN+1];
    char                                c_street_1[ADDRESS_LEN+1];
    char                                c_street_2[ADDRESS_LEN+1];
    char                                c_city[ADDRESS_LEN+1];
    char                                c_state[STATE_LEN+1];
    char                                c_zip[ZIP_LEN+1];
    char                                c_phone[PHONE_LEN+1];
    char                                c_credit[CREDIT_LEN+1];
    double                                      c_credit_lim;
    double                                      c_discount;
//  fix to avoid ODBC float to numeric conversion problem.
//  double                                      c_balance;
          char                                      c_balance[6];

    double                                      c_ytd_payment;
          short                                     c_payment_cnt;
          short                                     c_delivery_cnt;
          char                                      c_data[C_DATA_LEN+1];
          double                                    h_amount;
    char                                h_data[H_DATA_LEN+1];
} CUSTOMER_STRUCT;

typedef struct
{
          char                                      c_last[LAST_NAME_LEN+1];
          char                                      c_first[FIRST_NAME_LEN+1];
    long                                c_id;
```

```
} CUSTOMER_SORT_STRUCT;

typedef struct
{
    long                time_start;
} LOADER_TIME_STRUCT;


// Global variables

char      szLastError[300];

HENV      henv;

HDBC      i_hdbc1;                                          // for ITEM table
HDBC      w_hdbc1;                                          // for WAREHOUSE, DISTRICT,
STOCK
HDBC      c_hdbc1;                                          // for CUSTOMER
HDBC      c_hdbc2;                                          // for HISTORY
HDBC      o_hdbc1;                                          // for ORDERS
HDBC      o_hdbc2;                                          // for NEW-ORDER

HDBC      o_hdbc3;                                          // for ORDER-LINE

HSTMT     i_hstmt1;
HSTMT     w_hstmt1;
HSTMT     c_hstmt1, c_hstmt2;
HSTMT     o_hstmt1, o_hstmt2, o_hstmt3;

ORDERS_STRUCT   orders_buf[ORDERS_PER_DISTRICT];
CUSTOMER_STRUCT customer_buf[CUSTOMERS_PER_DISTRICT];
long            orders_rows_loaded;
long            new_order_rows_loaded;
long            order_line_rows_loaded;
long            history_rows_loaded;
long            customer_rows_loaded;
long            stock_rows_loaded;
long            district_rows_loaded;
long            item_rows_loaded;
long            warehouse_rows_loaded;
long            main_time_start;
long            main_time_end;
long                        max_items;
long                        customers_per_district;
long                        orders_per_district;
long                        first_new_order;
long                        last_new_order;


TPCCLDR_ARGS    *aptr, args;


//==============================================================================
//
// Function name: main
//
//==============================================================================

int main(int  argc, char **argv)
{

          DWORD           dwThreadID[MAX_MAIN_THREADS];
          HANDLE          hThread[MAX_MAIN_THREADS];
          FILE            *fLoader;
          char            buffer[255];
          int                                   i;

          for (i=0; i<MAX_MAIN_THREADS; i++)
                    hThread[i] = NULL;

    printf("\n*******************************************************");
          printf("\n*                                                     *");
          printf("\n*  Microsoft SQL Server                               *");
          printf("\n*                                                     *");
          printf("\n*  TPC-C BENCHMARK KIT:  Database loader              *");
          printf("\n*  Version %s                                        *", TPCKIT_VER);
          printf("\n*                                                     *");
          printf("\n*******************************************************\n\n");

    // process command line arguments

    aptr = &args;
    GetArgsLoader(argc, argv, aptr);

          printf("Build interface is ODBC.\n");

          if (aptr->build_index == 0)
                    printf("Data load only - no index creation.\n");
          else
                    printf("Data load and index creation.\n");

          if (aptr->index_order == 0)
                    printf("Clustered indexes will be created after bulk load.\n");
          else
                    printf("Clustered indexes will be created before bulk load.\n");

          // set database scale values
          if (aptr->scale_down == 1)
          {
                    printf("*** Scaled Down Database ***\n");
                    max_items = MAXITEMS_SCALE_DOWN;
                    customers_per_district = CUSTOMERS_SCALE_DOWN;
                    orders_per_district = ORDERS_SCALE_DOWN;
                    first_new_order = 0;
                    last_new_order  = 30;
          }
          else
          {
                    max_items = MAXITEMS;
                    customers_per_district = CUSTOMERS_PER_DISTRICT;
                    orders_per_district = ORDERS_PER_DISTRICT;
                    first_new_order = 2100;
                    last_new_order  = 3000;
          }

          // open connections to SQL Server

          OpenConnections();

          // open file for loader results
    fLoader = fopen(aptr->loader_res_file, "w");

    if (fLoader == NULL)
          {
                    printf("Error, loader result file open failed.");
                    exit(-1);
          }

    // start loading data

    sprintf(buffer,"TPC-C load started for %ld warehouses.\n",aptr->num_warehouses);

          printf("%s",buffer);
          fprintf(fLoader,"%s",buffer);

          main_time_start = (TimeNow() / MILLI);

          // start parallel load threads


          if (aptr->tables_all || aptr->table_item)
          {
```

```
                    fprintf(fLoader, "\nStarting loader threads for: item\n");

                    hThread[0] = CreateThread(NULL,
                                                                        0,
(LPTHREAD_START_ROUTINE) LoadItem,
                                                                        NULL,
                                                                        0,
&dwThreadID[0]);

                    if (hThread[0] == NULL)
                    {
                              printf("Error, failed in creating creating thread = 0.\n");
                              exit(-1);
                    }
          }

          if (aptr->tables_all || aptr->table_warehouse)
          {
                    fprintf(fLoader, "Starting loader threads for: warehouse\n");

                    hThread[1] = CreateThread(NULL,
                                                                        0,
(LPTHREAD_START_ROUTINE) LoadWarehouse,
                                                                        NULL,
                                                                        0,
&dwThreadID[1]);

                    if (hThread[1] == NULL)
                    {
                              printf("Error, failed in creating creating thread = 1.\n");
                              exit(-1);
                    }
          }

          if (aptr->tables_all || aptr->table_customer)
          {
                    fprintf(fLoader, "Starting loader threads for: customer\n");

                    hThread[2] = CreateThread(NULL,
                                                                        0,
(LPTHREAD_START_ROUTINE) LoadCustomer,
                                                                        NULL,
                                                                        0,
&dwThreadID[2]);

                    if (hThread[2] == NULL)
                    {
                              printf("Error, failed in creating creating main thread =
2.\n");
                              exit(-1);
                    }
          }

          if (aptr->tables_all || aptr->table_orders)
          {
                    fprintf(fLoader, "Starting loader threads for: orders\n");

                    hThread[3] = CreateThread(NULL,
                                                                        0,
(LPTHREAD_START_ROUTINE) LoadOrders,
                                                                        NULL,
                                                                        0,
&dwThreadID[3]);

                    if (hThread[3] == NULL)
                    {
                              printf("Error, failed in creating creating main thread =
3.\n");
                              exit(-1);
                    }
          }

          // Wait for threads to finish...
          for (i=0; i<MAX_MAIN_THREADS; i++)
          {
                    if (hThread[i] != NULL)
                    {
                              WaitForSingleObject( hThread[i], INFINITE );
                              CloseHandle(hThread[i]);
                              hThread[i] = NULL;
                    }
          }

          main_time_end = (TimeNow() / MILLI);

    sprintf(buffer,"\nTPC-C load completed successfully in %ld minutes.\n",
                              (main_time_end - main_time_start)/60);

          printf("%s",buffer);
          fprintf(fLoader, "%s", buffer);

          fclose(fLoader);

          SQLFreeEnv(henv);

    exit(0);

          return 0;
}


//===============================================================================
//
// Function name: LoadItem
//
//===============================================================================

void LoadItem()
{
    long            i_id;
          long            i_im_id;
    char            i_name[I_NAME_LEN+1];
    double                  i_price;
    char            i_data[I_DATA_LEN+1];
          char            name[20];
          long            time_start;
          RETCODE         rc;
          DBINT           rcint;
          char            bcphint[128];

          // Seed with unique number
          seed(1);

          printf("Loading item table...\n");

          // if build index before load
          if ((aptr->build_index == 1) && (aptr->index_order == 1))
                    BuildIndex("idxitmcl");

          InitString(i_name, I_NAME_LEN+1);
          InitString(i_data, I_DATA_LEN+1);

          sprintf(name, "%s..%s", aptr->database, "item");
```

```
        rc = bcp_init(i_hdbc1, name, NULL, "logs\\item.err", DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        if ((aptr->build_index == 1) && (aptr->index_order == 1))
        {
                sprintf(bcphint, "tablock, order (i_id), ROWS_PER_BATCH = 100000");
                rc = bcp_control(i_hdbc1, BCPHINTS, (void*) bcphint);
                if (rc != SUCCEED)
                        HandleErrorDBC(i_hdbc1);
        }

        rc = bcp_bind(i_hdbc1, (BYTE *) &i_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 1);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        rc = bcp_bind(i_hdbc1, (BYTE *) &i_im_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4,
2);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        rc = bcp_bind(i_hdbc1, (BYTE *) i_name, 0, I_NAME_LEN, NULL, 0, 0, 3);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        rc = bcp_bind(i_hdbc1, (BYTE *) &i_price, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8,
4);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        rc = bcp_bind(i_hdbc1, (BYTE *) i_data, 0, I_DATA_LEN, NULL, 0, 0, 5);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        time_start = (TimeNow() / MILLI);

        item_rows_loaded = 0;

        for (i_id = 1; i_id <= max_items; i_id++)
        {
                i_im_id = RandomNumber(1L, 10000L);

                MakeAlphaString(14, 24, I_NAME_LEN, i_name);

                i_price = ((float) RandomNumber(100L, 10000L))/100.0;

                MakeOriginalAlphaString(26, 50, I_DATA_LEN, i_data, 10);

                rc = bcp_sendrow(i_hdbc1);
                if (rc != SUCCEED)
                        HandleErrorDBC(i_hdbc1);

                item_rows_loaded++;
                CheckForCommit(i_hdbc1, i_hstmt1, item_rows_loaded, "item",
&time_start);
        }

        rcint = bcp_done(i_hdbc1);
        if (rcint < 0)
                HandleErrorDBC(i_hdbc1);

        printf("Finished loading item table.\n");

        SQLFreeStmt(i_hstmt1, SQL_DROP);
        SQLDisconnect(i_hdbc1);
        SQLFreeConnect(i_hdbc1);

        // if build index after load
        if ((aptr->build_index == 1) && (aptr->index_order == 0))
                BuildIndex("idxitmcl");
}


//============================================================================
//
// Function   : LoadWarehouse
//
// Loads WAREHOUSE table and loads Stock and District as Warehouses are created
//
//============================================================================

void LoadWarehouse()
{
    short w_id;
    char  w_name[W_NAME_LEN+1];
    char  w_street_1[ADDRESS_LEN+1];
    char  w_street_2[ADDRESS_LEN+1];
    char  w_city[ADDRESS_LEN+1];
    char  w_state[STATE_LEN+1];
    char  w_zip[ZIP_LEN+1];
    double          w_tax;
    double          w_ytd;
          char      name[20];
          long      time_start;
          RETCODE   rc;
          DBINT     rcint;
          char      bcphint[128];


          // Seed with unique number
          seed(2);

          printf("Loading warehouse table...\n");

          // if build index before load...
          if ((aptr->build_index == 1) && (aptr->index_order == 1))
                  BuildIndex("idxwarcl");

          InitString(w_name, W_NAME_LEN+1);
          InitAddress(w_street_1, w_street_2, w_city, w_state, w_zip);

          sprintf(name, "%s..%s", aptr->database, "warehouse");

          rc = bcp_init(w_hdbc1, name, NULL, "logs\\whouse.err", DB_IN);
          if (rc != SUCCEED)
                  HandleErrorDBC(w_hdbc1);

          if ((aptr->build_index == 1) && (aptr->index_order == 1))
          {
                  sprintf(bcphint, "tablock, order (w_id), ROWS_PER_BATCH = %d", aptr-
>num_warehouses);
                  rc = bcp_control(w_hdbc1, BCPHINTS, (void*) bcphint);
                  if (rc != SUCCEED)
                          HandleErrorDBC(w_hdbc1);
          }

          rc = bcp_bind(w_hdbc1, (BYTE *) &w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 1);
          if (rc != SUCCEED)
                  HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) w_name, 0, W_NAME_LEN, NULL, 0, 0, 2);
          if (rc != SUCCEED)
                  HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) w_street_1, 0, ADDRESS_LEN, NULL, 0, 0, 3);
          if (rc != SUCCEED)
                  HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) w_street_2, 0, ADDRESS_LEN, NULL, 0, 0, 4);
          if (rc != SUCCEED)
                  HandleErrorDBC(w_hdbc1);
```

```
          rc = bcp_bind(w_hdbc1, (BYTE *) w_city, 0, ADDRESS_LEN, NULL, 0, 0, 5);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) w_state, 0, STATE_LEN, NULL, 0, 0, 6);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) w_zip, 0, ZIP_LEN, NULL, 0, 0, 7);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) &w_tax, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8,
8);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) &w_ytd, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8,
9);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          time_start = (TimeNow() / MILLI);

          warehouse_rows_loaded = 0;

          for (w_id = (short)aptr->starting_warehouse; w_id <= aptr->num_warehouses;
w_id++)
          {
                    MakeAlphaString(6,10, W_NAME_LEN, w_name);

                    MakeAddress(w_street_1, w_street_2, w_city, w_state, w_zip);

                    w_tax = ((float) RandomNumber(0L,2000L))/10000.00;

                    w_ytd = 300000.00;

                    rc = bcp_sendrow(w_hdbc1);
                    if (rc != SUCCEED)
                              HandleErrorDBC(w_hdbc1);

                    warehouse_rows_loaded++;
                    CheckForCommit(w_hdbc1, i_hstmt1, warehouse_rows_loaded, "warehouse",
&time_start);
          }

          rcint = bcp_done(w_hdbc1);
          if (rcint < 0)
                    HandleErrorDBC(w_hdbc1);

          printf("Finished loading warehouse table.\n");

          // if build index after load...
          if ((aptr->build_index == 1) && (aptr->index_order == 0))
                    BuildIndex("idxwarcl");

          stock_rows_loaded = 0;
          district_rows_loaded = 0;

          District();
          Stock();

}


//==============================================================================
//
// Function   : District
//
//==============================================================================
```

```
void District()
{
    short d_id;
    short d_w_id;
    char  d_name[D_NAME_LEN+1];
    char  d_street_1[ADDRESS_LEN+1];
    char  d_street_2[ADDRESS_LEN+1];
    char  d_city[ADDRESS_LEN+1];
    char  d_state[STATE_LEN+1];
    char  d_zip[ZIP_LEN+1];
    double          d_tax;
    double          d_ytd;
          char      name[20];
    long  d_next_o_id;
          long      time_start;
          int                 w_id;
          RETCODE   rc;
          DBINT     rcint;
          char      bcphint[128];

          // Seed with unique number
          seed(4);

          printf("Loading district table...\n");

          // build index before load
          if ((aptr->build_index == 1) && (aptr->index_order == 1))
                    BuildIndex("idxdiscl");

          InitString(d_name, D_NAME_LEN+1);
          InitAddress(d_street_1, d_street_2, d_city, d_state, d_zip);
          sprintf(name, "%s..%s", aptr->database, "district");

          rc = bcp_init(w_hdbc1, name, NULL, "logs\\district.err", DB_IN);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          if ((aptr->build_index == 1) && (aptr->index_order == 1))
          {
                    sprintf(bcphint, "tablock, order (d_w_id, d_id), ROWS_PER_BATCH = %u",
(aptr->num_warehouses * 10));
                    rc = bcp_control(w_hdbc1, BCPHINTS, (void*) bcphint);
                    if (rc != SUCCEED)
                              HandleErrorDBC(w_hdbc1);
          }

          rc = bcp_bind(w_hdbc1, (BYTE *) &d_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 1);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) &d_w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
2);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) d_name, 0, D_NAME_LEN, NULL, 0, 0, 3);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) d_street_1, 0, ADDRESS_LEN, NULL, 0, 0, 4);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) d_street_2, 0, ADDRESS_LEN, NULL, 0, 0, 5);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) d_city, 0, ADDRESS_LEN, NULL, 0, 0, 6);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
```

```
          rc = bcp_bind(w_hdbc1, (BYTE *) d_state, 0, STATE_LEN, NULL, 0, 0, 7);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) d_zip, 0, ZIP_LEN, NULL, 0, 0, 8);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) &d_tax, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8,
9);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) &d_ytd, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8,
10);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) &d_next_o_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 11);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          d_ytd  = 30000.0;

          d_next_o_id = orders_per_district+1;

          time_start = (TimeNow() / MILLI);

          for (w_id = aptr->starting_warehouse; w_id <= aptr->num_warehouses; w_id++)
          {
                    d_w_id = w_id;

                    for (d_id = 1; d_id <= DISTRICT_PER_WAREHOUSE; d_id++)
                    {
                              MakeAlphaString(6,10,D_NAME_LEN, d_name);

                              MakeAddress(d_street_1, d_street_2, d_city, d_state, d_zip);

                              d_tax = ((float) RandomNumber(0L,2000L))/10000.00;

                              rc = bcp_sendrow(w_hdbc1);
                              if (rc != SUCCEED)
                                        HandleErrorDBC(w_hdbc1);

                              district_rows_loaded++;
                              CheckForCommit(w_hdbc1, w_hstmt1, district_rows_loaded,
"district", &time_start);
                    }
    }

          rcint = bcp_done(w_hdbc1);
          if (rcint < 0)
                    HandleErrorDBC(w_hdbc1);

          printf("Finished loading district table.\n");

          // if build index after load...
          if ((aptr->build_index == 1) && (aptr->index_order == 0))
                    BuildIndex("idxdiscl");

    return;
}


//====================================================================
//
// Function   : Stock
//
//====================================================================
```

```
void Stock()
{
    long  s_i_id;
    short s_w_id;
    short s_quantity;
    char  s_dist_01[S_DIST_LEN+1];
    char  s_dist_02[S_DIST_LEN+1];
    char  s_dist_03[S_DIST_LEN+1];
    char  s_dist_04[S_DIST_LEN+1];
    char  s_dist_05[S_DIST_LEN+1];
    char  s_dist_06[S_DIST_LEN+1];
    char  s_dist_07[S_DIST_LEN+1];
    char  s_dist_08[S_DIST_LEN+1];
    char  s_dist_09[S_DIST_LEN+1];
    char  s_dist_10[S_DIST_LEN+1];
    long  s_ytd;
    short s_order_cnt;
    short s_remote_cnt;
    char  s_data[S_DATA_LEN+1];
    short len;
          char      name[20];
          long      time_start;
          RETCODE   rc;
          DBINT     rcint;
          char      bcphint[128];

          // Seed with unique number
          seed(3);

          // if build index before load...
          if ((aptr->build_index == 1) && (aptr->index_order == 1))
                    BuildIndex("idxstkcl");

          sprintf(name, "%s..%s", aptr->database, "stock");

          rc = bcp_init(w_hdbc1, name, NULL, "logs\\stock.err", DB_IN);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          if ((aptr->build_index == 1) && (aptr->index_order == 1))
          {
                    sprintf(bcphint, "tablock, order (s_i_id, s_w_id), ROWS_PER_BATCH =
%u", (aptr->num_warehouses * 100000));
                    rc = bcp_control(w_hdbc1, BCPHINTS, (void*) bcphint);
                    if (rc != SUCCEED)
                              HandleErrorDBC(w_hdbc1);
          }

          rc = bcp_bind(w_hdbc1, (BYTE *) &s_i_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4,
1);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          bcp_bind(w_hdbc1, (BYTE *) &s_w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 2);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) &s_quantity, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 3);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_01, 0, S_DIST_LEN, NULL, 0, 0, 4);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_02, 0, S_DIST_LEN, NULL, 0, 0, 5);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
```

```
        rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_03, 0, S_DIST_LEN, NULL, 0, 0, 6);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_04, 0, S_DIST_LEN, NULL, 0, 0, 7);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_05, 0, S_DIST_LEN, NULL, 0, 0, 8);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_06, 0, S_DIST_LEN, NULL, 0, 0, 9);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_07, 0, S_DIST_LEN, NULL, 0, 0, 10);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_08, 0, S_DIST_LEN, NULL, 0, 0, 11);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_09, 0, S_DIST_LEN, NULL, 0, 0, 12);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_10, 0, S_DIST_LEN, NULL, 0, 0, 13);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) &s_ytd, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4,
14);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) &s_order_cnt, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 15);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) &s_remote_cnt, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 16);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *) s_data, 0, S_DATA_LEN, NULL, 0, 0, 17);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        s_ytd = s_order_cnt = s_remote_cnt = 0;

        time_start = (TimeNow() / MILLI);

        printf("...Loading stock table\n");

        for (s_i_id=1; s_i_id <= max_items; s_i_id++)
        {

                for (s_w_id = (short)aptr->starting_warehouse; s_w_id <= aptr-
>num_warehouses; s_w_id++)
                {

                        s_quantity = (short)RandomNumber(10L,100L);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_01);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_02);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_03);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_04);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_05);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_06);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_07);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_08);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_09);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_10);

                        len = MakeOriginalAlphaString(26,50, S_DATA_LEN, s_data,10);

                        rc = bcp_sendrow(w_hdbc1);
                        if (rc != SUCCEED)
                                HandleErrorDBC(w_hdbc1);

                        stock_rows_loaded++;
                        CheckForCommit(w_hdbc1, w_hstmt1, stock_rows_loaded,
"stock", &time_start);

                }
        }

        rcint = bcp_done(w_hdbc1);
        if (rcint < 0)
                HandleErrorDBC(w_hdbc1);

        printf("Finished loading stock table.\n");

        SQLFreeStmt(w_hstmt1, SQL_DROP);
        SQLDisconnect(w_hdbc1);
        SQLFreeConnect(w_hdbc1);

        // if build index after load...
        if ((aptr->build_index == 1) && (aptr->index_order == 0))
                BuildIndex("idxstkcl");

        return;
}


//=============================================================================
//
// Function   : LoadCustomer
//
//=============================================================================

void LoadCustomer()
{
        LOADER_TIME_STRUCT      customer_time_start;
        LOADER_TIME_STRUCT      history_time_start;
        short                   w_id;
    short                   d_id;
        DWORD                   dwThreadID[MAX_CUSTOMER_THREADS];
        HANDLE                  hThread[MAX_CUSTOMER_THREADS];
        char                    name[20];
        RETCODE                                         rc;
        DBINT                                           rcint;
        char                                            bcphint[128];
        char                                            cmd[256];
        // SQLRETURN                                    rc_1;
        // SQLSMALLINT                                  recnum, MsgLen;
        // SQLCHAR                                                 SqlState[6],
Msg[SQL_MAX_MESSAGE_LENGTH];
        // SQLINTEGER                                   NativeError;

        // Seed with unique number
        seed(5);

        printf("Loading customer and history tables...\n");

        // if build index before load...
        if ((aptr->build_index == 1) && (aptr->index_order == 1))
                BuildIndex("idxcuscl");

        // Initialize bulk copy
```

```
        sprintf(name, "%s..%s", aptr->database, "customer");

        rc = bcp_init(c_hdbc1, name, NULL, "logs\\customer.err", DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

        if ((aptr->build_index == 1) && (aptr->index_order == 1))
        {
                sprintf(bcphint, "tablock, order (c_w_id, c_d_id, c_id),
ROWS_PER_BATCH = %u", (aptr->num_warehouses * 30000));
                rc = bcp_control(c_hdbc1, BCPHINTS, (void*) bcphint);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc1);
        }

        sprintf(name, "%s..%s", aptr->database, "history");

        rc = bcp_init(c_hdbc2, name, NULL, "logs\\history.err", DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

        sprintf(bcphint, "tablock");
        rc = bcp_control(c_hdbc2, BCPHINTS, (void*) bcphint);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

        customer_rows_loaded    = 0;
        history_rows_loaded     = 0;

        CustomerBufInit();

        customer_time_start.time_start = (TimeNow() / MILLI);
        history_time_start.time_start = (TimeNow() / MILLI);

        for (w_id = (short)aptr->starting_warehouse; w_id <= aptr->num_warehouses;
w_id++)
        {
                for (d_id = 1; d_id <= DISTRICT_PER_WAREHOUSE; d_id++)
                {

                        CustomerBufLoad(d_id, w_id);

                        // Start parallel loading threads here...

                        // Start customer table thread

                        printf("...Loading customer table for: d_id = %d, w_id =
%d\n", d_id, w_id);

                        hThread[0] = CreateThread(NULL,

0,

(LPTHREAD_START_ROUTINE) LoadCustomerTable,

&customer_time_start,

0,

&dwThreadID[0]);

                        if (hThread[0] == NULL)
                        {
                                printf("Error, failed in creating creating thread
= 0.\n");
                                exit(-1);
                        }

                        // Start History table thread

                        printf("...Loading history table for: d_id = %d, w_id =
%d\n", d_id, w_id);

                        hThread[1] = CreateThread(NULL,

0,

(LPTHREAD_START_ROUTINE) LoadHistoryTable,

&history_time_start,

0,

&dwThreadID[1]);

                        if (hThread[1] == NULL)
                        {
                                printf("Error, failed in creating creating thread
= 1.\n");
                                exit(-1);
                        }

                        WaitForSingleObject( hThread[0], INFINITE );
                        WaitForSingleObject( hThread[1], INFINITE );

                        if (CloseHandle(hThread[0]) == FALSE)
                        {
                                printf("Error, failed in closing customer thread
handle with errno: %d\n", GetLastError());
                        }

                        if (CloseHandle(hThread[1]) == FALSE)
                        {
                                printf("Error, failed in closing history thread
handle with errno: %d\n", GetLastError());
                        }


                }

        }

        // flush the bulk connection
        rcint = bcp_done(c_hdbc1);
        if (rcint < 0)
                HandleErrorDBC(c_hdbc1);

        rcint = bcp_done(c_hdbc2);
        if (rcint < 0)
                HandleErrorDBC(c_hdbc2);

        printf("Finished loading customer table.\n");

        // if build index after load...
        if ((aptr->build_index == 1) && (aptr->index_order == 0))
                BuildIndex("idxcusc1");

        // build non-clustered index
        if (aptr->build_index == 1)
                BuildIndex("idxcusnc");

        // Output the NURAND used for the loader into C_FIRST for C_ID = 1,
        // C_W_ID = 1, and C_D_ID = 1
        sprintf(cmd, "isql -S%s -U%s -P%s -d%s -e -Q\"update customer set c_first =
'C_LOAD = %d' where c_id = 1 and c_w_id = 1 and c_d_id = 1\" > logs\\nurand_load.log",
                                        aptr->server,
                                        aptr->user,
                                        aptr->password,
                                        aptr->database,
                                        LOADER_NURAND_C);
```

```
          system(cmd);

          SQLFreeStmt(c_hstmt1, SQL_DROP);
          SQLDisconnect(c_hdbc1);
          SQLFreeConnect(c_hdbc1);

          SQLFreeStmt(c_hstmt2, SQL_DROP);
          SQLDisconnect(c_hdbc2);
          SQLFreeConnect(c_hdbc2);


    return;

}


//=============================================================================
//
// Function   : CustomerBufInit
//
//=============================================================================

void CustomerBufInit()
{
          int     i;

          for (i=0;i<customers_per_district;i++)
          {
                    customer_buf[i].c_id = 0;
                    customer_buf[i].c_d_id = 0;
                    customer_buf[i].c_w_id = 0;

                    strcpy(customer_buf[i].c_first,"");
                    strcpy(customer_buf[i].c_middle,"");
                    strcpy(customer_buf[i].c_last,"");
                    strcpy(customer_buf[i].c_street_1,"");
                    strcpy(customer_buf[i].c_street_2,"");
                    strcpy(customer_buf[i].c_city,"");
                    strcpy(customer_buf[i].c_state,"");
                    strcpy(customer_buf[i].c_zip,"");
                    strcpy(customer_buf[i].c_phone,"");
                    strcpy(customer_buf[i].c_credit,"");

                    customer_buf[i].c_credit_lim = 0;
                    customer_buf[i].c_discount = (float) 0;

                    //  fix to avoid ODBC float to numeric conversion problem.
                    //        customer_buf[i].c_balance = 0;
                    strcpy(customer_buf[i].c_balance,"");

                    customer_buf[i].c_ytd_payment = 0;
                    customer_buf[i].c_payment_cnt = 0;
                    customer_buf[i].c_delivery_cnt = 0;

                    strcpy(customer_buf[i].c_data,"");

                    customer_buf[i].h_amount = 0;

                    strcpy(customer_buf[i].h_data,"");

          }


}


//=============================================================================
//
// Function   : CustomerBufLoad
//
// Fills shared buffer for HISTORY and CUSTOMER
```

```
//=============================================================================

void CustomerBufLoad(int d_id, int w_id)
{
          long                                   i;
    CUSTOMER_SORT_STRUCT    c[CUSTOMERS_PER_DISTRICT];


          for (i=0;i<customers_per_district;i++)
          {
                    if (i < 1000)
                              LastName(i, c[i].c_last);
                    else
                              LastName(NURand(255,0,999,LOADER_NURAND_C), c[i].c_last);

                    MakeAlphaString(8,16,FIRST_NAME_LEN, c[i].c_first);

                    c[i].c_id = i+1;

          }

          printf("...Loading customer buffer for: d_id = %d, w_id = %d\n",
                              d_id, w_id);

          for (i=0;i<customers_per_district;i++)
          {

                    customer_buf[i].c_d_id = d_id;
                    customer_buf[i].c_w_id = w_id;
                    customer_buf[i].h_amount = 10.0;

                    customer_buf[i].c_ytd_payment  = 10.0;

                    customer_buf[i].c_payment_cnt  = 1;
                    customer_buf[i].c_delivery_cnt = 0;

                    // Generate CUSTOMER and HISTORY data

                    customer_buf[i].c_id = c[i].c_id;

                    strcpy(customer_buf[i].c_first, c[i].c_first);
                    strcpy(customer_buf[i].c_last, c[i].c_last);

                    customer_buf[i].c_middle[0] = 'O';
                    customer_buf[i].c_middle[1] = 'E';

                    MakeAddress(customer_buf[i].c_street_1,
                                        customer_buf[i].c_street_2,
                                        customer_buf[i].c_city,
                                        customer_buf[i].c_state,
                                        customer_buf[i].c_zip);

                    MakeNumberString(16, 16, PHONE_LEN, customer_buf[i].c_phone);

                    if (RandomNumber(1L, 100L) > 10)
                              customer_buf[i].c_credit[0] = 'G';
                    else
                              customer_buf[i].c_credit[0] = 'B';
                    customer_buf[i].c_credit[1] = 'C';

                    customer_buf[i].c_credit_lim = 50000.0;
                    customer_buf[i].c_discount = ((float) RandomNumber(0L, 5000L)) /
10000.0;

                    //  fix to avoid ODBC float to numeric conversion problem.
                    //  customer_buf[i].c_balance = -10.0;
                    strcpy(customer_buf[i].c_balance,"-10.0");

                    MakeAlphaString(500, 500, C_DATA_LEN, customer_buf[i].c_data);
```

```
                    // Generate HISTORY data
                    MakeAlphaString(12, 24, H_DATA_LEN, customer_buf[i].h_data);

          }
}

//===================================================================
//
// Function   : LoadCustomerTable
//
//===================================================================

void LoadCustomerTable(LOADER_TIME_STRUCT *customer_time_start)
{
          int         i;
    long        c_id;
    short       c_d_id;
    short       c_w_id;
    char        c_first[FIRST_NAME_LEN+1];
    char        c_middle[MIDDLE_NAME_LEN+1];
    char        c_last[LAST_NAME_LEN+1];
    char        c_street_1[ADDRESS_LEN+1];
    char        c_street_2[ADDRESS_LEN+1];
    char        c_city[ADDRESS_LEN+1];
    char        c_state[STATE_LEN+1];
    char        c_zip[ZIP_LEN+1];
    char        c_phone[PHONE_LEN+1];
    char        c_credit[CREDIT_LEN+1];
    double      c_credit_lim;
    double      c_discount;

          //  fix to avoid ODBC float to numeric conversion problem.
          //  double                  c_balance;
          char                c_balance[6];

    double      c_ytd_payment;
    short       c_payment_cnt;
    short       c_delivery_cnt;
    char        c_data[C_DATA_LEN+1];
          char                c_since[C_SINCE_LEN+1];
          RETCODE             rc;

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 1);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_d_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 2);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

          rc = bcp_bind(c_hdbc1, (BYTE *) &c_w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
3);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_first, 0, FIRST_NAME_LEN, NULL, 0, 0, 4);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_middle, 0, MIDDLE_NAME_LEN,NULL, 0, 0, 5);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_last, 0, LAST_NAME_LEN, NULL, 0, 0, 6);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_street_1, 0, ADDRESS_LEN, NULL, 0, 0, 7);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_street_2, 0, ADDRESS_LEN, NULL,0,0, 8);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_city, 0, ADDRESS_LEN, NULL, 0, 0, 9);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_state, 0, STATE_LEN, NULL, 0, 0, 10);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_zip, 0, ZIP_LEN, NULL, 0, 0, 11);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_phone, 0, PHONE_LEN, NULL, 0, 0, 12);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

          rc = bcp_bind(c_hdbc1, (BYTE *) &c_since, 0, C_SINCE_LEN, NULL, 0, SQLCHARACTER,
13);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_credit, 0, CREDIT_LEN, NULL, 0, 0, 14);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_credit_lim, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8,
15);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_discount, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8,
16);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);


          //  fix to avoid ODBC float to numeric conversion problem.
    //  rc = bcp_bind(c_hdbc1, (BYTE *) &c_balance, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8,
17);
          //  if (rc != SUCCEED)
          //        HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_balance, 0, 5, NULL, 0, SQLCHARACTER, 17);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);


    rc = bcp_bind(c_hdbc1, (BYTE *) &c_ytd_payment, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8,
18);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_payment_cnt, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
19);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_delivery_cnt,0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
20);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_data, 0, 500, NULL, 0, 0, 21);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

    for (i = 0; i < customers_per_district; i++)
    {
```

```
                    c_id = customer_buf[i].c_id;
                    c_d_id = customer_buf[i].c_d_id;
                    c_w_id = customer_buf[i].c_w_id;

                    strcpy(c_first, customer_buf[i].c_first);
                    strcpy(c_middle, customer_buf[i].c_middle);
                    strcpy(c_last, customer_buf[i].c_last);
                    strcpy(c_street_1, customer_buf[i].c_street_1);
                    strcpy(c_street_2, customer_buf[i].c_street_2);
                    strcpy(c_city, customer_buf[i].c_city);
                    strcpy(c_state, customer_buf[i].c_state);
                    strcpy(c_zip, customer_buf[i].c_zip);
                    strcpy(c_phone, customer_buf[i].c_phone);
                    strcpy(c_credit, customer_buf[i].c_credit);

                    FormatDate(&c_since);

                    c_credit_lim = customer_buf[i].c_credit_lim;
                    c_discount = customer_buf[i].c_discount;

                    //  fix to avoid ODBC float to numeric conversion problem.

                    //  c_balance = customer_buf[i].c_balance;
                    strcpy(c_balance, customer_buf[i].c_balance);

                    c_ytd_payment = customer_buf[i].c_ytd_payment;
                    c_payment_cnt = customer_buf[i].c_payment_cnt;
                    c_delivery_cnt = customer_buf[i].c_delivery_cnt;

                    strcpy(c_data, customer_buf[i].c_data);

                    // Send data to server
                    rc = bcp_sendrow(c_hdbc1);
                    if (rc != SUCCEED)
                            HandleErrorDBC(c_hdbc1);

                    customer_rows_loaded++;
                    CheckForCommit(c_hdbc1, c_hstmt1, customer_rows_loaded, "customer",
&customer_time_start->time_start);
    }

}


//====================================================================
//
// Function   : LoadHistoryTable
//
//====================================================================

void LoadHistoryTable(LOADER_TIME_STRUCT *history_time_start)
{
        int         i;
    long        c_id;
    short       c_d_id;
    short       c_w_id;
        double      h_amount;
    char        h_data[H_DATA_LEN+1];
        char                h_date[H_DATE_LEN+1];
        RETCODE             rc;

    rc = bcp_bind(c_hdbc2, (BYTE *) &c_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 1);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &c_d_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 2);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &c_w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 3);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &c_d_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 4);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &c_w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 5);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

        rc = bcp_bind(c_hdbc2, (BYTE *) &h_date, 0, H_DATE_LEN, NULL, 0, SQLCHARACTER,
6);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &h_amount, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8, 7);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) h_data, 0, H_DATA_LEN, NULL, 0, 0, 8);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

    for (i = 0; i < customers_per_district; i++)
    {
                    c_id = customer_buf[i].c_id;
                    c_d_id = customer_buf[i].c_d_id;
                    c_w_id = customer_buf[i].c_w_id;
                    h_amount = customer_buf[i].h_amount;
                    strcpy(h_data, customer_buf[i].h_data);

                    FormatDate(&h_date);

                    // send to server
                    rc = bcp_sendrow(c_hdbc2);
                    if (rc != SUCCEED)
                            HandleErrorDBC(c_hdbc2);

                    history_rows_loaded++;
                    CheckForCommit(c_hdbc2, c_hstmt2, history_rows_loaded, "history",
&history_time_start->time_start);
    }

}


//=============================================================-===============
//
// Function   : LoadOrders
//
//=========================================================================

void LoadOrders()
{
        LOADER_TIME_STRUCT      orders_time_start;
        LOADER_TIME_STRUCT      new_order_time_start;
        LOADER_TIME_STRUCT      order_line_time_start;
        short                           w_id;
    short                   d_id;
        DWORD                           dwThreadID[MAX_ORDER_THREADS];
        HANDLE                          hThread[MAX_ORDER_THREADS];
        char                            name[20];
        RETCODE                                                 rc;
        char                                                    bcphint[128];

        // seed with unique number
        seed(6);

        printf("Loading orders...\n");

        // if build index before load...
```

```
        if ((aptr->build_index == 1) && (aptr->index_order == 1))
        {
                        BuildIndex("idxordcl");
                        BuildIndex("idxnodcl");
                        BuildIndex("idxodlcl");
        }

        // initialize bulk copy
        sprintf(name, "%s..%s", aptr->database, "orders");

        rc = bcp_init(o_hdbc1, name, NULL, "logs\\orders.err", DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

        if ((aptr->build_index == 1) && (aptr->index_order == 1))
        {
                sprintf(bcphint, "tablock, order (o_w_id, o_d_id, o_id),
ROWS_PER_BATCH = %u", (aptr->num_warehouses * 30000));
                rc = bcp_control(o_hdbc1, BCPHINTS, (void*) bcphint);
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc1);
        }

        sprintf(name, "%s..%s", aptr->database, "new_order");

        rc = bcp_init(o_hdbc2, name, NULL, "logs\\neword.err", DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc2);

        if ((aptr->build_index == 1) && (aptr->index_order == 1))
        {
                sprintf(bcphint, "tablock, order (no_w_id, no_d_id, no_o_id),
ROWS_PER_BATCH = %u", (aptr->num_warehouses * 9000));
                rc = bcp_control(o_hdbc2, BCPHINTS, (void*) bcphint);
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc2);
        }

        sprintf(name, "%s..%s", aptr->database, "order_line");

        rc = bcp_init(o_hdbc3, name, NULL, "logs\\ordline.err", DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc3);

    if ((aptr->build_index == 1) && (aptr->index_order == 1))
        {
                sprintf(bcphint, "tablock, order (ol_w_id, ol_d_id, ol_o_id,
ol_number), ROWS_PER_BATCH = %u", (aptr->num_warehouses * 300000));
                rc = bcp_control(o_hdbc3, BCPHINTS, (void*) bcphint);
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc3);
        }

        orders_rows_loaded      = 0;
        new_order_rows_loaded   = 0;
        order_line_rows_loaded  = 0;

        OrdersBufInit();

        orders_time_start.time_start = (TimeNow() / MILLI);
        new_order_time_start.time_start = (TimeNow() / MILLI);
        order_line_time_start.time_start = (TimeNow() / MILLI);

        for (w_id = (short)aptr->starting_warehouse; w_id <= aptr->num_warehouses;
w_id++)
        {
                for (d_id = 1; d_id <= DISTRICT_PER_WAREHOUSE; d_id++)
                {

                        OrdersBufLoad(d_id, w_id);
```

```
                        // start parallel loading threads here...

                        // start Orders table thread

                        printf("...Loading Order Table for: d_id = %d, w_id = %d\n",
d_id, w_id);

                        hThread[0] = CreateThread(NULL,

0,

(LPTHREAD_START_ROUTINE) LoadOrdersTable,

&orders_time_start,

0,

&dwThreadID[0]);

                        if (hThread[0] == NULL)
                        {
                                printf("Error, failed in creating creating thread
= 0.\n");
                                exit(-1);
                        }

                        // start NewOrder table thread

                        printf("...Loading New-Order Table for: d_id = %d, w_id =
%d\n", d_id, w_id);

                        hThread[1] = CreateThread(NULL,

0,

(LPTHREAD_START_ROUTINE) LoadNewOrderTable,

&new_order_time_start,

0,

&dwThreadID[1]);

                        if (hThread[1] == NULL)
                        {
                                printf("Error, failed in creating creating thread
= 1.\n");
                                exit(-1);
                        }

                        // start Order-Line table thread

                        printf("...Loading Order-Line Table for: d_id = %d, w_id =
%d\n", d_id, w_id);

                        hThread[2] = CreateThread(NULL,

0,

(LPTHREAD_START_ROUTINE) LoadOrderLineTable,

&order_line_time_start,

0,

&dwThreadID[2]);

                        if (hThread[2] == NULL)
                        {
                                printf("Error, failed in creating creating thread
= 2.\n");
```

```
                                        exit(-1);
                                }

                                WaitForSingleObject( hThread[0], INFINITE );
                                WaitForSingleObject( hThread[1], INFINITE );
                                WaitForSingleObject( hThread[2], INFINITE );

                                if (CloseHandle(hThread[0]) == FALSE)
                                {
                                        printf("Error, failed in closing Orders thread
handle with errno: %d\n", GetLastError());
                                }

                                if (CloseHandle(hThread[1]) == FALSE)
                                {
                                        printf("Error, failed in closing NewOrder thread
handle with errno: %d\n", GetLastError());
                                }

                                if (CloseHandle(hThread[2]) == FALSE)
                                {
                                        printf("Error, failed in closing OrderLine thread
handle with errno: %d\n", GetLastError());
                                }

                        }
                }

                printf("Finished loading orders.\n");


        return;
}


//================================================================================
//
// Function   : OrdersBufInit
//
// Clears shared buffer for ORDERS, NEWORDER, and ORDERLINE
//
//================================================================================

void OrdersBufInit()
{
        int     i;
        int             j;

        for (i=0;i<orders_per_district;i++)
        {
                orders_buf[i].o_id = 0;
                orders_buf[i].o_d_id = 0;
                orders_buf[i].o_w_id = 0;
                orders_buf[i].o_c_id = 0;
                orders_buf[i].o_carrier_id = 0;
                orders_buf[i].o_ol_cnt = 0;
                orders_buf[i].o_all_local = 0;

                for (j=0;j<=14;j++)
                {
                        orders_buf[i].o_ol[j].ol = 0;
                        orders_buf[i].o_ol[j].ol_i_id = 0;
                        orders_buf[i].o_ol[j].ol_supply_w_id = 0;
                        orders_buf[i].o_ol[j].ol_quantity = 0;
                        orders_buf[i].o_ol[j].ol_amount = 0;
                        strcpy(orders_buf[i].o_ol[j].ol_dist_info,"");
                }
        }


}
```

```
//================================================================================
//
// Function   : OrdersBufLoad
//
// Fills shared buffer for ORDERS, NEWORDER, and ORDERLINE
//
//================================================================================

void OrdersBufLoad(int d_id, int w_id)
{

        int     cust[ORDERS_PER_DIST+1];
        long    o_id;
        short   ol;

        printf("...Loading Order Buffer for: d_id = %d, w_id = %d\n",
                        d_id, w_id);

        GetPermutation(cust, ORDERS_PER_DIST);

        for (o_id=0;o_id<orders_per_district;o_id++)
        {

                // Generate ORDER and NEW-ORDER data

                orders_buf[o_id].o_d_id = d_id;
                orders_buf[o_id].o_w_id = w_id;
                orders_buf[o_id].o_id = o_id+1;
                orders_buf[o_id].o_c_id = cust[o_id+1];
                orders_buf[o_id].o_ol_cnt = (short)RandomNumber(5L, 15L);

                if (o_id < first_new_order)
                {
                        orders_buf[o_id].o_carrier_id = (short)RandomNumber(1L,
10L);
                        orders_buf[o_id].o_all_local  = 1;
                }
                else
                {
                        orders_buf[o_id].o_carrier_id = 0;
                        orders_buf[o_id].o_all_local  = 1;
                }

                for (ol=0; ol<orders_buf[o_id].o_ol_cnt; ol++)
                {

                        orders_buf[o_id].o_ol[ol].ol = ol+1;
                        orders_buf[o_id].o_ol[ol].ol_i_id = RandomNumber(1L,
max_items);
                        orders_buf[o_id].o_ol[ol].ol_supply_w_id = w_id;
                        orders_buf[o_id].o_ol[ol].ol_quantity = 5;
                        MakeAlphaString(24, 24, OL_DIST_INFO_LEN,
&orders_buf[o_id].o_ol[ol].ol_dist_info);

                        // Generate ORDER-LINE data
                        if (o_id < first_new_order)
                        {
                                orders_buf[o_id].o_ol[ol].ol_amount = 0;
                                // Added to insure ol_delivery_d set properly
during load


        FormatDate(&orders_buf[o_id].o_ol[ol].ol_delivery_d);

                        }
                        else
                        {
                                orders_buf[o_id].o_ol[ol].ol_amount =
RandomNumber(1,999999)/100.0;
```

```
                                        // Added to insure ol_delivery_d set properly
during load

                                        // odbc datetime format

          strcpy(orders_buf[o_id].o_ol[ol].ol_delivery_d,"1899-12-31 12:00:00.000");

                                }
                        }
          }
}


//========================================================================
//
// Function   : LoadOrdersTable
//
//========================================================================

void LoadOrdersTable(LOADER_TIME_STRUCT *orders_time_start)
{
          int         i;
    long        o_id;
          short     o_d_id;
          short     o_w_id;
    long        o_c_id;
    short       o_carrier_id;
    short       o_ol_cnt;
    short       o_all_local;
          char                o_entry_d[O_ENTRY_D_LEN+1];
          RETCODE             rc;
          DBINT               rcint;

          // bind ORDER data
    rc = bcp_bind(o_hdbc1, (BYTE *) &o_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 1);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *) &o_d_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 2);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *) &o_w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 3);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *) &o_c_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 4);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

          rc = bcp_bind(o_hdbc1, (BYTE *) &o_entry_d, 0, O_ENTRY_D_LEN, NULL, 0,
SQLCHARACTER, 5);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *) &o_carrier_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
6);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *) &o_ol_cnt, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 7);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *) &o_all_local, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
8);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

    for (i = 0; i < orders_per_district; i++)
    {
                    o_id         = orders_buf[i].o_id;
                    o_d_id       = orders_buf[i].o_d_id;
                    o_w_id       = orders_buf[i].o_w_id;
                    o_c_id       = orders_buf[i].o_c_id;
                    o_carrier_id = orders_buf[i].o_carrier_id;
                    o_ol_cnt     = orders_buf[i].o_ol_cnt;
                    o_all_local  = orders_buf[i].o_all_local;

                    FormatDate(&o_entry_d);

                    // send data to server
                    rc = bcp_sendrow(o_hdbc1);
                    if (rc != SUCCEED)
                              HandleErrorDBC(o_hdbc1);

                    orders_rows_loaded++;
                    CheckForCommit(o_hdbc1, o_hstmt1, orders_rows_loaded, "orders",
&orders_time_start->time_start);
    }

          // rcint = bcp_batch(o_hdbc1);
          // if (rcint < 0)
          //        HandleErrorDBC(o_hdbc1);

          if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
          {
                    rcint = bcp_done(o_hdbc1);
                    if (rcint < 0)
                              HandleErrorDBC(o_hdbc1);

                    SQLFreeStmt(o_hstmt1, SQL_DROP);
                    SQLDisconnect(o_hdbc1);
                    SQLFreeConnect(o_hdbc1);

                    // if build index after load...
                    if ((aptr->build_index == 1) && (aptr->index_order == 0))
                              BuildIndex("idxordcl");

                    // build non-clustered index
                    if (aptr->build_index == 1)
                              BuildIndex("idxordnc");
          }

}


//========================================================================
//
// Function   : LoadNewOrderTable
//
//========================================================================

void LoadNewOrderTable(LOADER_TIME_STRUCT *new_order_time_start)
{
    int         i;
    long        o_id;
    short       o_d_id;
    short       o_w_id;
          RETCODE             rc;
          DBINT               rcint;

          // Bind NEW-ORDER data

    rc = bcp_bind(o_hdbc2, (BYTE *) &o_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 1);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc2);

    rc = bcp_bind(o_hdbc2, (BYTE *) &o_d_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 2);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc2);
```

```
    rc = bcp_bind(o_hdbc2, (BYTE *) &o_w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 3);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc2);

          for (i = first_new_order; i < last_new_order; i++)
          {
                    o_id    = orders_buf[i].o_id;
                    o_d_id  = orders_buf[i].o_d_id;
                    o_w_id  = orders_buf[i].o_w_id;

                    rc = bcp_sendrow(o_hdbc2);
                    if (rc != SUCCEED)
                              HandleErrorDBC(o_hdbc2);

                    new_order_rows_loaded++;
                    CheckForCommit(o_hdbc2, o_hstmt2, new_order_rows_loaded, "new_order",
&new_order_time_start->time_start);
          }

          // rcint = bcp_batch(o_hdbc2);
          // if (rcint < 0)
          //        HandleErrorDBC(o_hdbc2);

          if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
          {
                    rcint = bcp_done(o_hdbc2);
                    if (rcint < 0)
                              HandleErrorDBC(o_hdbc2);

                    SQLFreeStmt(o_hstmt2, SQL_DROP);
                    SQLDisconnect(o_hdbc2);
                    SQLFreeConnect(o_hdbc2);

                    // if build index after load...
                    if ((aptr->build_index == 1) && (aptr->index_order == 0))
                              BuildIndex("idxnodcl");

          }

}


//====================================================================
//
// Function   : LoadOrderLineTable
//
//====================================================================

void LoadOrderLineTable(LOADER_TIME_STRUCT *order_line_time_start)
{
          int         i,j;
    long        o_id;
          short     o_d_id;
          short     o_w_id;
    long        ol;
          long      ol_i_id;
    short       ol_supply_w_id;
    short       ol_quantity;
    double      ol_amount;
    char        ol_dist_info[DIST_INFO_LEN+1];
          char              ol_delivery_d[OL_DELIVERY_D_LEN+1];
          RETCODE           rc;
          DBINT             rcint;

          // bind ORDER-LINE data
    rc = bcp_bind(o_hdbc3, (BYTE *) &o_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 1);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *) &o_d_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 2);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *) &o_w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, 3);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *) &ol, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 4);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *) &ol_i_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 5);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *) &ol_supply_w_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
6);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

          rc = bcp_bind(o_hdbc3, (BYTE *) &ol_delivery_d, 0, OL_DELIVERY_D_LEN, NULL, 0,
SQLCHARACTER, 7);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *) &ol_quantity, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
8);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *) &ol_amount, 0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8, 9);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *) ol_dist_info, 0, DIST_INFO_LEN, NULL, 0, 0, 10);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

          for (i = 0; i < orders_per_district; i++)
    {
                    o_id    = orders_buf[i].o_id;
                    o_d_id  = orders_buf[i].o_d_id;
                    o_w_id  = orders_buf[i].o_w_id;

                    for (j=0; j < orders_buf[i].o_ol_cnt; j++)
                    {
                              ol             = orders_buf[i].o_ol[j].ol;
                              ol_i_id        = orders_buf[i].o_ol[j].ol_i_id;
                              ol_supply_w_id = orders_buf[i].o_ol[j].ol_supply_w_id;
                              ol_quantity    = orders_buf[i].o_ol[j].ol_quantity;
                              ol_amount      = orders_buf[i].o_ol[j].ol_amount;

                              strcpy(ol_delivery_d,orders_buf[i].o_ol[j].ol_delivery_d);

                              strcpy(ol_dist_info,orders_buf[i].o_ol[j].ol_dist_info);

                              rc = bcp_sendrow(o_hdbc3);
                              if (rc != SUCCEED)
                                        HandleErrorDBC(o_hdbc3);

                              order_line_rows_loaded++;
                              CheckForCommit(o_hdbc3, o_hstmt3, order_line_rows_loaded,
"order_line", &order_line_time_start->time_start);
                    }

          }

          // rcint = bcp_batch(o_hdbc3);
          // if (rcint < 0)
          //        HandleErrorDBC(o_hdbc3);

          if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
```

```
        {
                rcint = bcp_done(o_hdbc3);
                if (rcint < 0)
                        HandleErrorDBC(o_hdbc3);

                SQLFreeStmt(o_hstmt3, SQL_DROP);
                SQLDisconnect(o_hdbc3);
                SQLFreeConnect(o_hdbc3);

                // if build index after load...
                if ((aptr->build_index == 1) && (aptr->index_order == 0))
                        BuildIndex("idxodlcl");

        }

}


//====================================================================
//
// Function   : GetPermutation
//
//====================================================================

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

        for (i=1;i<=n;i++)
                perm[i] = i;

        for (i=1;i<=n;i++)
        {
                r = RandomNumber(i,n);
                t = perm[i];
                perm[i] = perm[r];
                perm[r] = t;
        }

}


//====================================================================
//
// Function   : CheckForCommit
//
//====================================================================

void CheckForCommit(HDBC hdbc,
                                        HSTMT hstmt,
                                int rows_loaded,
                                        char *table_name,
                                long *time_start)
{

    long  time_end, time_diff;
          // DBINT  rcint;

    if ( !(rows_loaded % aptr->batch) )
    {

                // rcint = bcp_batch(hdbc);
                // if (rcint < 0)
                //        HandleErrorDBC(hdbc);

                time_end = (TimeNow() / MILLI);
                time_diff = time_end - *time_start;

                printf("-> Loaded %ld rows into %s in %ld sec - Total = %d (%.2f
rps)\n",
                        aptr->batch,
                                table_name,
                                        time_diff,
                                rows_loaded,
                                        (float) aptr->batch / (time_diff ? time_diff :
1L));

                *time_start = time_end;
    }

    return;
}


//====================================================================
//
// Function   : OpenConnections
//
//====================================================================

void OpenConnections()
{

        RETCODE         rc;

        char                            szDriverString[300];
        char                            szDriverStringOut[1024];
        SQLSMALLINT                     cbDriverStringOut;


        SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv );

        SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3, 0 );

        SQLAllocHandle(SQL_HANDLE_DBC, henv , &i_hdbc1);
        SQLAllocHandle(SQL_HANDLE_DBC, henv , &w_hdbc1);
        SQLAllocHandle(SQL_HANDLE_DBC, henv , &c_hdbc1);
        SQLAllocHandle(SQL_HANDLE_DBC, henv , &c_hdbc2);
        SQLAllocHandle(SQL_HANDLE_DBC, henv , &o_hdbc1);
        SQLAllocHandle(SQL_HANDLE_DBC, henv , &o_hdbc2);
        SQLAllocHandle(SQL_HANDLE_DBC, henv , &o_hdbc3);

        SQLSetConnectAttr(i_hdbc1, SQL_COPT_SS_BCP, (void *)SQL_BCP_ON, SQL_IS_INTEGER
);
        SQLSetConnectAttr(w_hdbc1, SQL_COPT_SS_BCP, (void *)SQL_BCP_ON, SQL_IS_INTEGER
);
        SQLSetConnectAttr(c_hdbc1, SQL_COPT_SS_BCP, (void *)SQL_BCP_ON, SQL_IS_INTEGER
);
        SQLSetConnectAttr(c_hdbc2, SQL_COPT_SS_BCP, (void *)SQL_BCP_ON, SQL_IS_INTEGER
);
        SQLSetConnectAttr(o_hdbc1, SQL_COPT_SS_BCP, (void *)SQL_BCP_ON, SQL_IS_INTEGER
);
        SQLSetConnectAttr(o_hdbc2, SQL_COPT_SS_BCP, (void *)SQL_BCP_ON, SQL_IS_INTEGER
);
        SQLSetConnectAttr(o_hdbc3, SQL_COPT_SS_BCP, (void *)SQL_BCP_ON, SQL_IS_INTEGER
);

        // Open connections to SQL Server

        // Connection 1

        sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,
                                                                aptr->server,
                                                                aptr->user,
                                                                aptr->password,
                                                                aptr->database );

        rc = SQLSetConnectOption (i_hdbc1, SQL_PACKET_SIZE, aptr->pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);
```

```
        rc = SQLDriverConnect ( i_hdbc1,
                                                NULL,
                                                      (SQLCHAR*)&szDriverString[0] ,
                                                      SQL_NTS,

        (SQLCHAR*)&szDriverStringOut[0],
                                                      sizeof(szDriverStringOut),
                                                      &cbDriverStringOut,
                                                      SQL_DRIVER_NOPROMPT );
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        // Connection 2

        sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,
                                                        aptr->server,
                                                        aptr->user,
                                                        aptr->password,
                                                        aptr->database );

        rc = SQLSetConnectOption (w_hdbc1, SQL_PACKET_SIZE, aptr->pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = SQLDriverConnect ( w_hdbc1,
                                                                NULL,

        (SQLCHAR*)&szDriverString[0] ,
                                                                SQL_NTS,

        (SQLCHAR*)&szDriverStringOut[0],

        sizeof(szDriverStringOut),
                                                                &cbDriverStringOut,
                                                                SQL_DRIVER_NOPROMPT
);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        // Connection 3

        sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,
                                                        aptr->server,
                                                        aptr->user,
                                                        aptr->password,
                                                        aptr->database );

        rc = SQLSetConnectOption (c_hdbc1, SQL_PACKET_SIZE, aptr->pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

        rc = SQLDriverConnect ( c_hdbc1,
                                                                NULL,

        (SQLCHAR*)&szDriverString[0] ,
                                                                SQL_NTS,

        (SQLCHAR*)&szDriverStringOut[0],

        sizeof(szDriverStringOut),
                                                                &cbDriverStringOut,
                                                                SQL_DRIVER_NOPROMPT
);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

        // Connection 4

        sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,
                                                                aptr->server,
                                                                aptr->user,
                                                                aptr->password,
                                                                aptr->database );

        rc = SQLSetConnectOption (c_hdbc2, SQL_PACKET_SIZE, aptr->pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

        rc = SQLDriverConnect ( c_hdbc2,
                                                                NULL,

        (SQLCHAR*)&szDriverString[0] ,
                                                                SQL_NTS,

        (SQLCHAR*)&szDriverStringOut[0],

        sizeof(szDriverStringOut),
                                                                &cbDriverStringOut,
                                                                SQL_DRIVER_NOPROMPT
);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

        // Connection 5

        sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,
                                                                aptr->server,
                                                                aptr->user,
                                                                aptr->password,
                                                                aptr->database );

        rc = SQLSetConnectOption (o_hdbc1, SQL_PACKET_SIZE, aptr->pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

        rc = SQLDriverConnect ( o_hdbc1,
                                                                NULL,

        (SQLCHAR*)&szDriverString[0] ,
                                                                SQL_NTS,

        (SQLCHAR*)&szDriverStringOut[0],

        sizeof(szDriverStringOut),
                                                                &cbDriverStringOut,
                                                                SQL_DRIVER_NOPROMPT
);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

        // Connection 6

        sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,
                                                                aptr->server,
                                                                aptr->user,
                                                                aptr->password,
                                                                aptr->database );

        rc = SQLSetConnectOption (o_hdbc2, SQL_PACKET_SIZE, aptr->pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc2);

        rc = SQLDriverConnect ( o_hdbc2,
                                                                NULL,

        (SQLCHAR*)&szDriverString[0] ,
```

```
                                                                SQL_NTS,

          (SQLCHAR*)&szDriverStringOut[0],

          sizeof(szDriverStringOut),
                                                                &cbDriverStringOut,
                                                                SQL_DRIVER_NOPROMPT
);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc2);

          // Connection 7

          sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,
                                                          aptr->server,
                                                          aptr->user,
                                                          aptr->password,
                                                          aptr->database );

          rc = SQLSetConnectOption (o_hdbc3, SQL_PACKET_SIZE, aptr->pack_size);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

          rc = SQLDriverConnect ( o_hdbc3,
                                                                NULL,

          (SQLCHAR*)&szDriverString[0] ,
                                                                SQL_NTS,

          (SQLCHAR*)&szDriverStringOut[0],

          sizeof(szDriverStringOut),
                                                                &cbDriverStringOut,
                                                                SQL_DRIVER_NOPROMPT
);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

}

//====================================================================
//
// Function name: BuildIndex
//
//====================================================================

void BuildIndex(char          *index_script)
{
          char      cmd[256];

          printf("Starting index creation:  %s\n",index_script);

          sprintf(cmd, "isql -S%s -U%s -P%s -e -i%s\\%s.sql > logs\\%s.log",
                                    aptr->server,
                                    aptr->user,
                                    aptr->password,
                                    aptr->index_script_path,
                                    index_script,
                                    index_script);

          system(cmd);

          printf("Finished index creation:  %s\n",index_script);
}


void HandleErrorDBC (SQLHDBC  hdbc1)
{

          SQLCHAR             SqlState[6], Msg[SQL_MAX_MESSAGE_LENGTH];
          SQLINTEGER  NativeError;
          SQLSMALLINT i, MsgLen;
          SQLRETURN   rc2;
          char                timebuf[128];
          char                datebuf[128];
          FILE                *fp1;

          i = 1;
          while (( rc2 = SQLGetDiagRec(SQL_HANDLE_DBC , hdbc1, i, SqlState , &NativeError,
                                          Msg, sizeof(Msg) , &MsgLen )) != SQL_NO_DATA )
          {

                    sprintf( szLastError , "%s" ,  Msg );

                    _strtime(timebuf);
                    _strdate(datebuf);

                    printf( "[%s : %s] %s\n" , datebuf, timebuf, szLastError);

                    fp1 = fopen("logs\\tpccldr.err","w");
                    if (fp1 == NULL)
                              printf("ERROR:  Unable to open errorlog file.\n");
                    else
                    {
                              fprintf(fp1, "[%s : %s] %s\n" , datebuf, timebuf,
szLastError);
                              fclose(fp1);
                    }

                    i++;

          }
}


void FormatDate ( char* szTimeCOutput )
{

          struct tm when;
          time_t now;

          time( &now );
          when  = *localtime( &now );

          mktime( &when );

          // odbc datetime format
          strftime( szTimeCOutput , 30 , "%Y-%m-%d %H:%M:%S.000", &when );

          return;

}
```

getargs.c

```
//        File:               GETARGS.C
//                                      Microsoft TPC-C Kit Ver. 4.00
//                                      Copyright Microsoft, 1996, 1997, 1998
//        Purpose:  Source file for command line processing


// Includes
#include "tpcc.h"

//=====================================================================
//
// Function name: GetArgsLoader
//
//=====================================================================

void GetArgsLoader(int argc, char **argv, TPCCLDR_ARGS *pargs)
{
    int             i;
    char  *ptr;

#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsLoader()\n", (int) GetCurrentThreadId());
#endif

    /* init args struct with some useful values */
    pargs->server                       =  SERVER;
    pargs->user                         =  USER;
    pargs->password                     =  PASSWORD;
    pargs->database                     =  DATABASE;
    pargs->batch                        =  BATCH;
    pargs->num_warehouses               =  UNDEF;
          pargs->tables_all                       =  TRUE;
          pargs->table_item                       =  FALSE;
          pargs->table_warehouse                  =  FALSE;
          pargs->table_customer                   =  FALSE;
          pargs->table_orders                     =  FALSE;
          pargs->loader_res_file                  =  LOADER_RES_FILE;
          pargs->pack_size                        =  DEFLDPACKSIZE;
          pargs->starting_warehouse     =  DEF_STARTING_WAREHOUSE;
          pargs->build_index                      =  BUILD_INDEX;
          pargs->index_order                      =  INDEX_ORDER;
          pargs->index_script_path      =  INDEX_SCRIPT_PATH;
          pargs->scale_down                       =  SCALE_DOWN;

    /* check for zero command line args */
    if ( argc == 1 )
                    GetArgsLoaderUsage();

    for (i = 1; i < argc; ++i)
    {
                    if (argv[i][0] != '-' && argv[i][0] != '/')
                    {
                    printf("\nUnrecognized command");
                    GetArgsLoaderUsage();
                    exit(1);
                    }

                    ptr = argv[i];

                    switch (ptr[1])
                    {
                    case 'h':     /* Fall throught */
                    case 'H':
                                        GetArgsLoaderUsage();
                                        break;
case 'D':
                    pargs->database = ptr+2;
                    break;

case 'P':
                    pargs->password = ptr+2;
                    break;

case 'S':
                    pargs->server = ptr+2;
                    break;

case 'U':
                    pargs->user = ptr+2;
                    break;

case 'b':
                    pargs->batch = atol(ptr+2);
                    break;

case 'W':
                    pargs->num_warehouses = atol(ptr+2);
                    break;

case 's':
                    pargs->starting_warehouse = atol(ptr+2);
                    break;

case 't':
                    {
                              pargs->tables_all = FALSE;
                              if (strcmp(ptr+2,"item") == 0)
                                        pargs->table_item = TRUE;
                              else if (strcmp(ptr+2,"warehouse") == 0)
                                        pargs->table_warehouse = TRUE;
                              else if (strcmp(ptr+2,"customer") == 0)
                                        pargs->table_customer = TRUE;
                              else if (strcmp(ptr+2,"orders") == 0)
                                        pargs->table_orders = TRUE;
                              else
                              {
                              printf("\nUnrecognized command");
                              GetArgsLoaderUsage();
                              exit(1);
                              }

                              break;
                    }
          case 'f':
                    pargs->loader_res_file = ptr+2;
                    break;

          case 'p':
                    pargs->pack_size = atol(ptr+2);
                    break;

          case 'i':
                    pargs->build_index = atol(ptr+2);
                    break;

          case 'o':
                    pargs->index_order = atol(ptr+2);
                    break;

          case 'c':
                    pargs->scale_down = atol(ptr+2);
                    break;
case 'd':
                    pargs->index_script_path = ptr+2;
```

```
                                        break;

                    default:
                              GetArgsLoaderUsage();
                              exit(-1);
                              break;
                    }
    }

    /* check for required args */
    if (pargs->num_warehouses == UNDEF )
    {
                    printf("Number of Warehouses is required\n");
                    exit(-2);
    }

    return;
}


//======================================================================
//
// Function name: GetArgsLoaderUsage
//
//======================================================================

void GetArgsLoaderUsage()
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsLoaderUsage()\n", (int) GetCurrentThreadId());
#endif


    printf("TPCCLDR:\n\n");
          printf("Parameter                                                    Default\n");
          printf("-------------------------------------------------------------\n");
    printf("-W Number of Warehouses to Load                      Required \n");
    printf("-S Server                                            %s\n", SERVER);
    printf("-U Username                                          %s\n", USER);
    printf("-P Password                                          %s\n", PASSWORD);
    printf("-D Database                                          %s\n", DATABASE);
          printf("-b Batch Size                                        %ld\n", (long)
BATCH);
          printf("-p TDS packet size                                   %ld\n", (long)
DEFLDPACKSIZE);
          printf("-f Loader Results Output Filename                    %s\n",
LOADER_RES_FILE);
          printf("-s Starting Warehouse                                %ld\n", (long)
DEF_STARTING_WAREHOUSE);
          printf("-i Build Option (data = 0, data and index = 1)       %ld\n", (long)
BUILD_INDEX);
          printf("-o Cluster Index Build Order (before = 1, after = 0) %ld\n", (long)
INDEX_ORDER);
          printf("-c Build Scaled Database (normal = 0, tiny = 1)      %ld\n", (long)
SCALE_DOWN);
          printf("-d Index Script Path                                 %s\n",
INDEX_SCRIPT_PATH);
          printf("-t Table to Load                                     all tables \n");
    printf("   [item|warehouse|customer|orders]\n");
    printf("   Notes: \n");
    printf("     - the '-t' parameter may be included multiple times to \n");
    printf("       specify multiple tables to be loaded \n");
    printf("     - 'item' loads ITEM table \n");
    printf("     - 'warehouse' loads WAREHOUSE, DISTRICT, and STOCK tables \n");
    printf("     - 'customer' loads CUSTOMER and HISTORY tables \n");
    printf("     - 'orders' load NEW-ORDER, ORDERS, ORDER-LINE tables \n");

          printf("\nNote:  Command line switches are case sensitive.\n");

    exit(0);
}
```

random.c

```
//        File:               RANDOM.C
//                                      Microsoft TPC-C Kit Ver. 4.00
//                                      Copyright Microsoft, 1996, 1997, 1998
//        Purpose:  Random number generation routines for database loader


// 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. 4.00
//                                    Copyright Microsoft, 1996, 1997, 1998
//        Purpose:  Source file for database loader string functions


// 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)
{
    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:
//4.3.2.2 The 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)
{
            int                 len;
            int                 i;
            static    char chArray[] =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
            static    int       chArrayMax = 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;
}


//=========================================================================
//
// Function name: MakeOriginalAlphaString
//
//=========================================================================

int MakeOriginalAlphaString(int  x,
                                                        int  y,
                                                             int  z,
                                                        char *str,
                                                    int  percent)
{
    int             len;
    int             val;
    int             start;

#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;
}


//=========================================================================
//
// 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;
}


//=========================================================================
//
// Function name: InitString
//
//=========================================================================
void InitString(char *str, int len)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering InitString()\n", (int) GetCurrentThreadId());
#endif

          memset(str, ' ', len);
    str[len] = 0;
}


//=========================================================================
// Function name: InitAddress
//
// Description:
//
//=========================================================================

void InitAddress(char *street_1, char *street_2, char *city, char *state, char *zip)
{
          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;
}


//==================================================================
//
// Function name: PaddString
//
//==================================================================

void PaddString(int max, char *name)
{
        int                 len;

        len = strlen(name);
        if ( len < max )
                memset(name+len, ' ', max - len);
        name[max] = 0;

        return;
}
```

time.c

```
//      File:               TIME.C
//                                  Microsoft TPC-C Kit Ver. 4.00
//                                  Copyright Microsoft, 1996, 1997, 1998
//      Purpose:  Source file for time functions


// Includes
#include "tpcc.h"

// Globals
static long start_sec;


//==================================================================
//
// Function name: TimeNow
//
//==================================================================

long TimeNow()
{
   long             time_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;
}
```

Appendix C - Tunable Parameters

Server Configuration Parameters

Microsoft Windows NT Server Version 4.0 Tunable Parameters

No Windows NT registry parameters were modified for this benchmark.

Microsoft Windows NT Server Version 4.0 Configuration

The following services were disabled on the server:

- Computer Browser
- License Logging Service
- Messenger
- NT LM Security Support Provider
- Plug and Play
- RPC Locator Service
- TCP/IP Netbios Helper
- Spooler
- Server

Microsoft SQL Server Version 7.0 Startup Parameters

Microsoft SQL Server was started with the following command line options

sqlservr -c -x -T3502 –g38

where

-c	Start SQL Server independently of the Microsoft Windows NT Service Control Manager.
-x	Disable the keeping of CPU time and cache-hit ratio statistics.
-T3502	Prints a message to the log at the beginning and end of each checkpoint.
-g 38	Reserve 38 MB for non-buffer pool allocations

LOG Drives Layout

The log drives were placed on one PERC2 controller across four channels with eight 18G drives. Hardware RAID 10 was used to mirror the two 4-disk stripe sets. Write Caching was disabled on the controller.

Microsoft SQL Server Stack Size

The default stack size of Microsoft SQL Server was changed using the EDITBIN utility. The EDITBIN utility ships with Microsoft Visual C++ V5.0. The command used was editbin /stack:65536 sqlservr.exe.

Microsoft SQL Server 7.0 Configuration Parameters

name	minimum	maximum	config_value	run_value
affinity mask	0	2147483647	15	15
allow updates	0	1	0	0
cost threshold for parallelism	0	32767	5	5
cursor threshold	-1	2147483647	-1	-1
default language	0	9999	0	0
default sortorder id	0	255	50	50

Appendix C – Tunable Parameters

extended memory size (MB)	0	2147483647	0	0
fill factor (%)	0	100	0	0
index create memory (KB)	704	1600000	0	0
language in cache	3	100	3	3
language neutral full-text	0	1	0	0
lightweight pooling	0	1	1	1
locks	5000	2147483647	0	0
max async IO	1	255	255	255
max degree of parallelism	0	32	1	1
max server memory (MB)	4	2147483647	2950	2950
max text repl size (B)	0	2147483647	65536	65536
max worker threads	10	1024	190	190
media retention	0	365	0	0
min memory per query (KB)	512	2147483647	512	512
min server memory (MB)	0	2147483647	2950	2950
nested triggers	0	1	1	1
network packet size (B)	512	65535	4096	4096
open objects	0	2147483647	0	0
priority boost	0	1	1	1
query governor cost limit	0	2147483647	0	0
query wait (s)	-1	2147483647	-1	-1
recovery interval (min)	0	32767	32767	32767
remote access	0	1	0	0
remote login timeout (s)	0	2147483647	5	5
remote proc trans	0	1	0	0
remote query timeout (s)	0	2147483647	0	0
resource timeout (s)	5	2147483647	10	10
scan for startup procs	0	1	0	0
set working set size	0	1	1	1
show advanced options	0	1	1	1
spin counter	1	2147483647	10000	10000
time slice (ms)	50	1000	100	100
two digit year cutoff	1753	9999	2049	2049
Unicode comparison style	0	2147483647	0	0
Unicode locale id	0	2147483647	33280	33280
user connections	0	32767	190	190
user options	0	4095	0	0

Appendix C – Tunable Parameters

Disk Controller Configuration Parameters

The disk controller used during this benchmark were configured as follows:

```
C:\Dell Perc2>afacli @list.cli
DELL PowerEdge Expandable RAID Controller 2, Command Line Interface   V1.0
Copyright (c) 1997 Adaptec, Inc.
--------------------------------------------------------------------------------

Executing: open "afa0"

Executing: controller details
Controller Information
----------------------
             Device Name: afa0
         Controller type: PERC 2
Controller serial number: Last six digits = A7E105
         Number of Buses: 4
         Devices per bus: 15
          Controller CPU: Strong Arm 110
    Controller CPU speed: 233 Mhz
       Controller Memory: 48 Mbytes

Component Revisions
-------------------
                CLI: 1.0-0 (Build #1588)
                API: 1.0-0 (Build #1588)
 File System Driver: 1.0-2 (Build #1801)
    Miniport Driver: 1.0-2 (Build #1801)
Controller Software: 1.0-2 (Build #1801)
Controller Firmware: (Build #1801)
Controller Hardware: 1.0

Executing: close

Executing: open "afa1"

Executing: controller details
Controller Information
----------------------
             Device Name: afa1
         Controller type: PERC 2
Controller serial number: Last six digits = 5B90E0
         Number of Buses: 4
         Devices per bus: 15
          Controller CPU: Strong Arm 110
    Controller CPU speed: 233 Mhz
       Controller Memory: 48 Mbytes

Component Revisions
-------------------
                CLI: 1.0-0 (Build #1588)
                API: 1.0-0 (Build #1588)
 File System Driver: 1.0-2 (Build #1801)
    Miniport Driver: 1.0-2 (Build #1801)
Controller Software: 1.0-2 (Build #1801)
Controller Firmware: (Build #1801)
Controller Hardware: 1.0

Executing: close

Executing: open "afa2"

Executing: controller details
Controller Information
----------------------
             Device Name: afa2
         Controller type: PERC 2
Controller serial number: Last six digits = 5B9DAA
         Number of Buses: 4
```

```
         Devices per bus: 15
          Controller CPU: Strong Arm 110
    Controller CPU speed: 233 Mhz
       Controller Memory: 48 Mbytes

Component Revisions
-------------------
                CLI: 1.0-0 (Build #1588)
                API: 1.0-0 (Build #1588)
 File System Driver: 1.0-2 (Build #1801)
    Miniport Driver: 1.0-2 (Build #1801)
Controller Software: 1.0-2 (Build #1801)
Controller Firmware: (Build #1801)
Controller Hardware: 1.0

Executing: close

Executing: open "afa3"

Executing: controller details
Controller Information
----------------------
             Device Name: afa3
         Controller type: PERC 2
Controller serial number: Last six digits = 7A2B8A
         Number of Buses: 4
         Devices per bus: 15
          Controller CPU: Strong Arm 110
    Controller CPU speed: 233 Mhz
       Controller Memory: 48 Mbytes

Component Revisions
-------------------
                CLI: 1.0-0 (Build #1588)
                API: 1.0-0 (Build #1588)
 File System Driver: 1.0-2 (Build #1801)
    Miniport Driver: 1.0-2 (Build #1801)
Controller Software: 1.0-2 (Build #1801)
Controller Firmware: (Build #1801)
Controller Hardware: 1.0

Executing: close

Executing: open "afa4"

Executing: controller details
Controller Information
----------------------
             Device Name: afa4
         Controller type: PERC 2
Controller serial number: Last six digits = 5B9057
         Number of Buses: 4
         Devices per bus: 15
          Controller CPU: Strong Arm 110
    Controller CPU speed: 233 Mhz
       Controller Memory: 48 Mbytes

Component Revisions
-------------------
                CLI: 1.0-0 (Build #1588)
                API: 1.0-0 (Build #1588)
 File System Driver: 1.0-2 (Build #1801)
    Miniport Driver: 1.0-2 (Build #1801)
Controller Software: 1.0-2 (Build #1801)
Controller Firmware: (Build #1801)
Controller Hardware: 1.0

Executing: close

Executing: open "afa5"

Executing: controller details
Controller Information
----------------------
```

```
            Device Name: afa5
         Controller type: PERC 2
Controller serial number: Last six digits = 7A33A8
         Number of Buses: 4
         Devices per bus: 15
          Controller CPU: Strong Arm 110
    Controller CPU speed: 233 Mhz
       Controller Memory: 48 Mbytes

Component Revisions
-------------------
               CLI: 1.0-0 (Build #1588)
               API: 1.0-0 (Build #1588)
 File System Driver: 1.0-2 (Build #1801)
    Miniport Driver: 1.0-2 (Build #1801)
Controller Software: 1.0-2 (Build #1801)
Controller Firmware: (Build #1801)
Controller Hardware: 1.0

Executing: close

Executing: exit

DELL PowerEdge Expandable RAID Controller 2, Command Line Interface   V1.0
Copyright (c) 1997 Adaptec, Inc.
--------------------------------------------------------------------------------

Executing: open "afa0"

Executing: container show cache 0

Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status        : DISABLED
Write Cache Status       : DISABLED
Stream Detection Status  : DISABLED

Executing: container show cache 1

Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status        : DISABLED
Write Cache Status       : DISABLED
Stream Detection Status  : DISABLED

Executing: close

Executing: open "afa1"

Executing: container show cache 0

Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status        : DISABLED
Write Cache Status       : DISABLED
Stream Detection Status  : DISABLED

Executing: container show cache 1

Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status        : DISABLED
Write Cache Status       : DISABLED
Stream Detection Status  : DISABLED

Executing: close

Executing: open "afa2"

Executing: container show cache 0
```

```
Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status         : DISABLED
Write Cache Status        : DISABLED
Stream Detection Status   : DISABLED

Executing: container show cache 1

Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status         : DISABLED
Write Cache Status        : DISABLED
Stream Detection Status   : DISABLED

Executing: close

Executing: open "afa3"

Executing: container show cache 0

Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status         : DISABLED
Write Cache Status        : DISABLED
Stream Detection Status   : DISABLED

Executing: container show cache 1

Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status         : DISABLED
Write Cache Status        : DISABLED
Stream Detection Status   : DISABLED

Executing: close

Executing: open "afa4"

Executing: container show cache 0

Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status         : DISABLED
Write Cache Status        : DISABLED
Stream Detection Status   : DISABLED

Executing: container show cache 1

Global Container Read Cache Size  : 13549568
Global Container Write Cache Size : 8085504

Read Cache Status         : DISABLED
Write Cache Status        : DISABLED
Stream Detection Status   : DISABLED

Executing: close

Executing: open "afa5"

Executing: container show cache 0

Global Container Read Cache Size  : 0
Global Container Write Cache Size : 0

Read Cache Status         : DISABLED
Write Cache Status        : DISABLED
Stream Detection Status   : DISABLED
```

```
Executing: container show cache 1

Global Container Read Cache Size  : 0
Global Container Write Cache Size : 0

Read Cache Status        : DISABLED
Write Cache Status       : DISABLED
Stream Detection Status  : DISABLED

Executing: close

Executing: exit


C:\Dell Perc2>list

C:\Dell Perc2>call list_fsu 0

C:\Dell Perc2>fsu init nucleus 0; config symbols aac364.fsg; config get
InterruptDeferTime; config get AvgResponsesQueued; config
get AvgNonDeferredInterrupts; config get AvgDeferredInterrupts; config get
AvgAdapCmdQueDepth; config get AvgHostRespQueDepth; e
xit
Connect comm link to NT kernel driver "\\.\afa0"
Opening FS device \\.\afa0 to send Fib to.
Opening FS device \\.\afa0 to send Fib to.
3644 symbols loaded
   0x000f4fac:0x00000008 8
   0x000f4f58:0x00000a3d 2621
   0x000f4f60:0x000000f6 246
   0x000f4f68:0x00000947 2375
   0x000f4f7c:0x00000000 0
   0x000f4f84:0x00000003 3

C:\Dell Perc2>call list_fsu 1

C:\Dell Perc2>fsu init nucleus 1; config symbols aac364.fsg; config get
InterruptDeferTime; config get AvgResponsesQueued; config
get AvgNonDeferredInterrupts; config get AvgDeferredInterrupts; config get
AvgAdapCmdQueDepth; config get AvgHostRespQueDepth; e
xit
Connect comm link to NT kernel driver "\\.\afa1"
Opening FS device \\.\afa1 to send Fib to.
Opening FS device \\.\afa1 to send Fib to.
3644 symbols loaded
   0x000f4fac:0x00000008 8
   0x000f4f58:0x00000a43 2627
   0x000f4f60:0x000000d1 209
   0x000f4f68:0x00000972 2418
   0x000f4f7c:0x00000000 0
   0x000f4f84:0x00000003 3

C:\Dell Perc2>call list_fsu 2

C:\Dell Perc2>fsu init nucleus 2; config symbols aac364.fsg; config get
InterruptDeferTime; config get AvgResponsesQueued; config
get AvgNonDeferredInterrupts; config get AvgDeferredInterrupts; config get
AvgAdapCmdQueDepth; config get AvgHostRespQueDepth; e
xit
Connect comm link to NT kernel driver "\\.\afa2"
Opening FS device \\.\afa2 to send Fib to.
Opening FS device \\.\afa2 to send Fib to.
3644 symbols loaded
   0x000f4fac:0x00000008 8
   0x000f4f58:0x00000a3e 2622
   0x000f4f60:0x000000a8 168
   0x000f4f68:0x00000995 2453
   0x000f4f7c:0x00000000 0
   0x000f4f84:0x00000003 3

C:\Dell Perc2>call list_fsu 3

C:\Dell Perc2>fsu init nucleus 3; config symbols aac364.fsg; config get
InterruptDeferTime; config get AvgResponsesQueued; config
```

```
get AvgNonDeferredInterrupts; config get AvgDeferredInterrupts; config get
AvgAdapCmdQueDepth; config get AvgHostRespQueDepth; e
xit
Connect comm link to NT kernel driver "\\.\afa3"
Opening FS device \\.\afa3 to send Fib to.
Opening FS device \\.\afa3 to send Fib to.
3644 symbols loaded
   0x000f4fac:0x00000008 8
   0x000f4f58:0x00000a46 2630
   0x000f4f60:0x0000008e 142
   0x000f4f68:0x000009b7 2487
   0x000f4f7c:0x00000000 0
   0x000f4f84:0x00000003 3

C:\Dell Perc2>call list_fsu 4

C:\Dell Perc2>fsu init nucleus 4; config symbols aac364.fsg; config get
InterruptDeferTime; config get AvgResponsesQueued; config
get AvgNonDeferredInterrupts; config get AvgDeferredInterrupts; config get
AvgAdapCmdQueDepth; config get AvgHostRespQueDepth; e
xit
Connect comm link to NT kernel driver "\\.\afa4"
Opening FS device \\.\afa4 to send Fib to.
Opening FS device \\.\afa4 to send Fib to.
3644 symbols loaded
   0x000f4fac:0x00000008 8
   0x000f4f58:0x00000a49 2633
   0x000f4f60:0x00000070 112
   0x000f4f68:0x000009d9 2521
   0x000f4f7c:0x00000000 0
   0x000f4f84:0x00000003 3

C:\Dell Perc2>call list_fsu 5

C:\Dell Perc2>fsu init nucleus 5; config symbols aac364.fsg; config get
InterruptDeferTime; config get AvgResponsesQueued; config
get AvgNonDeferredInterrupts; config get AvgDeferredInterrupts; config get
AvgAdapCmdQueDepth; config get AvgHostRespQueDepth; e
xit
Connect comm link to NT kernel driver "\\.\afa5"
Opening FS device \\.\afa5 to send Fib to.
Opening FS device \\.\afa5 to send Fib to.
3644 symbols loaded
   0x000f4fac:0x00000006 6
   0x000f4f58:0x000008b0 2224
   0x000f4f60:0x00000106 262
   0x000f4f68:0x000007a9 1961
   0x000f4f7c:0x00000000 0
   0x000f4f84:0x00000002 2

C:\Dell Perc2>
```

Appendix C – Tunable Parameters

Microsoft Diagnostics Report For PE6300

```
Microsoft Diagnostics Report For \\PE6300
------------------------------------------------------------------------

OS Version Report
------------------------------------------------------------------------
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381: Service Pack 4) x86 Multiprocessor Free
Registered Owner: tpcc, dell computer corp.
Product Number: 02196-OEM-0123456-01234
------------------------------------------------------------------------

System Report
------------------------------------------------------------------------
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: MPS 1.4 - APIC platform
BIOS Date: 02/10/99
BIOS Version: Phoenix ROM BIOS PLUS Version 1.

Processor list:
   0:  x86 Family 6 Model 7 Stepping 2 GenuineIntel ~500 Mhz
   1:  x86 Family 6 Model 7 Stepping 2 GenuineIntel ~500 Mhz
   2:  x86 Family 6 Model 7 Stepping 2 GenuineIntel ~500 Mhz
   3:  x86 Family 6 Model 7 Stepping 2 GenuineIntel ~500 Mhz
------------------------------------------------------------------------

Video Display Report
------------------------------------------------------------------------
BIOS Date: <unavailable>
Adapter:
   Setting: 1024 x 768 x 65536
            75 Hz
   Type: ati compatible display adapter
   String: ATI Graphics Accelerator
   Memory: 2 MB
   Chip Type: ATI 3D RAGE PRO PCI (GT-C2U2)
   DAC Type: ATI Internal DAC
Driver:
   Vendor: ATI Technologies Inc.
   File(s): ati.sys, ati.dll, 8514a.dll
   Version: 4.3.92, 4.0.0


Drives Report
------------------------------------------------------------------------
C:\  (Local - NTFS)  Total: 1,044,193 KB, Free: 443,817 KB
   Serial Number: D49A - 2743
   Bytes per cluster: 512
   Sectors per cluster: 1
   Filename length: 255
D:\  (Local - NTFS)  Total: 7,839,680 KB, Free: 4,403,968 KB
   Serial Number: B05A - 3B51
   Bytes per cluster: 512
   Sectors per cluster: 128
   Filename length: 255

Memory Report
------------------------------------------------------------------------
Handles: 1,000
Threads: 118
Processes: 16

Physical Memory (K)
   Total: 3,997,100
   Available: 3,820,892
   File Cache: 13,748
```

Appendix C – Tunable Parameters

```
Kernel Memory (K)
   Total: 19,264
   Paged: 8,628
   Nonpaged: 10,636

Commit Charge (K)
   Total: 27,204
   Limit: 6,199,040
   Peak: 3,102,120

Pagefile Space (K)
   Total: 2,358,272
   Total in use: 4,268
   Peak: 4,276

   C:\pagefile.sys
      Total: 262,144
      Total in use: 2,208
      Peak: 2,216

   D:\pagefile.sys
      Total: 2,096,128
      Total in use: 2,060
      Peak: 2,060



Services Report
----------------------------------------------------------------------
FAST Remote Services Agent                    Stopped   (Manual)
   C:\Program Files\PERC2\System\AfaAgent.Exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
   Service Dependencies:
      RpcSs
Alerter                                       Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
Computer Browser                              Stopped   (Manual)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
      LmHosts
ClipBook Server                               Stopped   (Manual)
   C:\WINNT\system32\clipsrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      NetDDE
DHCP Client (TDI)                             Stopped   (Disabled)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      Tcpip
      Afd
      NetBT
EventLog (Event log)                          Running   (Automatic)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Server                                        Running   (Automatic)
```

```
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
Workstation (NetworkProvider)                  Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
License Logging Service                        Stopped   (Manual)
   C:\WINNT\System32\llssrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TCP/IP NetBIOS Helper                          Stopped   (Manual)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      NetworkProvider
Messenger                                      Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      NetBios
MSDTC (MS Transactions)                        Stopped   (Manual)
   C:\WINNT\System32\msdtc.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      RPCSS
      NTLMSSP
MSSQLServer                                    Stopped   (Manual)
   D:\MSSQL7\binn\sqlservr.exe
   Service Account Name: .\Administrator
   Error Severity: Normal
   Service Flags: Own Process
Network DDE (NetDDEGroup)                      Stopped   (Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      NetDDEDSDM
Network DDE DSDM                               Stopped   (Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Net Logon (RemoteValidation)                   Stopped   (Manual)
   C:\WINNT\System32\lsass.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LmHosts
Network Monitor Agent                          Stopped   (Manual)
   C:\WINNT\System32\nmagent.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      bh
NT LM Security Support Provider                Running   (Manual)
```

```
   C:\WINNT\System32\SERVICES.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Plug and Play (PlugPlay)                        Stopped   (Manual)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Protected Storage                               Running   (Automatic)
   c:\winnt\system32\pstores.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
   Service Dependencies:
      RpcSs
Directory Replicator                            Stopped   (Manual)
   C:\WINNT\System32\lmrepl.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
Remote Procedure Call (RPC) Locator             Stopped   (Manual)
   C:\WINNT\System32\LOCATOR.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      LanmanWorkstation
      Rdr
Remote Procedure Call (RPC) Service             Running   (Automatic)
   C:\WINNT\system32\RpcSs.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Schedule                                        Stopped   (Manual)
   C:\WINNT\System32\AtSvc.Exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Simple TCP/IP Services                          Stopped   (Manual)
   C:\WINNT\system32\tcpsvcs.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      Afd
   Group Dependencies:
      TDI
Spooler (SpoolerGroup)                          Stopped   (Manual)
   C:\WINNT\system32\spoolss.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
SQLServerAgent                                  Stopped   (Manual)
   D:\MSSQL7\binn\sqlagent.exe
   Service Account Name: .\Administrator
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      MSSQLServer
Telephony Service                               Stopped   (Manual)
   C:\WINNT\system32\tapisrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
UPS                                             Stopped   (Manual)
   C:\WINNT\System32\ups.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
```

```
Drivers Report
--------------------------------------------------------------------------
Abiosdsk (Primary disk)                          Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
afacomm (SCSI miniport)                          Stopped   (Disabled)
   System32\DRIVERS\afacomm.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
afadisk (Primary Disk)                           Stopped   (Disabled)
   System32\DRIVERS\afadisk.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
afasa (SCSI miniport)                            Stopped   (Disabled)
   System32\DRIVERS\afasa.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
afascsi (port)                                   Stopped   (Disabled)
   System32\DRIVERS\afascsi.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
AFD Networking Support Environment (TDI)         Running   (Automatic)
   C:\WINNT\System32\drivers\afd.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Aha154x (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Aha174x (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
aic78u2 (SCSI miniport)                          Running   (Boot)
   C:\WINNT\system32\drivers\aic78u2.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
aic78xx (SCSI miniport)                          Running   (Boot)
   C:\WINNT\System32\DRIVERS\aic78xx.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Always (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
ami0nt (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
amsint (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Arrow (SCSI miniport)                            Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
atapi (SCSI miniport)                            Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Atdisk (Primary disk)                            Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
ati (Video)                                      Running   (System)
   System32\DRIVERS\ati.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Beep (Base)                                      Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Network Monitor Tools and Agent Drivers          Running   (Manual)
   C:\WINNT\System32\drivers\bhnt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      NDIS
```

```
BusLogic (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Busmouse (Pointer Port)                         Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cdaudio (Filter)                                Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cdfs (File system)                              Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      SCSI CDROM Class
Cdrom (SCSI CDROM Class)                        Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Changer (Filter)                                Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
cirrus (Video)                                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cpqarray (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
cpqfws2e (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dac960nt (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dce376nt (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Delldsa (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Dell_DGX (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Disk (SCSI Class)                               Running   (Boot)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Diskperf (Filter)                               Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
DptScsi (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dtc329x (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Intel(R) PRO NDIS Driver (NDIS)                 Running   (Automatic)
   C:\WINNT\System32\drivers\E100BNT.SYS
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
et4000 (Video)                                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
FAfaSa (SCSI miniport)                          Running   (System)
   System32\DRIVERS\FAfaSa.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
FastAfa                                         Running   (Automatic)
   System32\DRIVERS\FastAfa.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Fastfat (Boot file system)                      Running   (Disabled)
   Error Severity: Normal
```

```
   Service Flags: File System Driver, Shared Process
Fd16_700 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Fd7000ex (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Fd8xx (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
flashpnt (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Floppy (Primary disk)                           Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Ftdisk (Filter)                                 Stopped   (Manual)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
i8042 Keyboard and PS/2 Mouse Port Driver (Keyboard Port) Running   (System)
   System32\DRIVERS\i8042prt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Inport (Pointer Port)                           Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jazzg300 (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jazzg364 (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jzvxl484 (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Keyboard Class Driver (Keyboard Class)          Running   (System)
   System32\DRIVERS\kbdclass.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
KSecDD (Base)                                   Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mga (Video)                                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mga_mil (Video)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mitsumi (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mkecr5xx (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Modem (Extended base)                           Stopped   (Manual)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Mouse Class Driver (Pointer Class)              Running   (System)
   System32\DRIVERS\mouclass.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Msfs (File system)                              Running   (System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Mup (Network)                                   Running   (Manual)
   C:\WINNT\System32\drivers\mup.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Ncr53c9x (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
ncr77c22 (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
```

```
Ncrc700 (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ncrc710 (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Microsoft NDIS System Driver (NDIS)             Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
NetBIOS Interface (NetBIOSGroup)                Running   (Manual)
   C:\WINNT\System32\drivers\netbios.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      TDI
WINS Client(TCP/IP) (PNP_TDI)                   Running   (Automatic)
   C:\WINNT\System32\drivers\netbt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Tcpip
NetDetect                                       Stopped   (Manual)
   C:\WINNT\system32\drivers\netdtect.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Npfs (File system)                              Running   (System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Ntfs (File system)                              Running   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Null (Base)                                     Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Oliscsi (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Parallel (Extended base)                        Stopped   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Parport
   Group Dependencies:
      Parallel arbitrator
Parport (Parallel arbitrator)                   Stopped   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
ParVdm (Extended base)                          Stopped   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Parport
   Group Dependencies:
      Parallel arbitrator
PCIDump (PCI Configuration)                     Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Pcmcia (System Bus Extender)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
PnP ISA Enabler Driver (Base)                   Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
psidisp (Video)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Ql10wnt (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
qv (Video)                                      Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Rdr (Network)                                   Running   (Manual)
   C:\WINNT\System32\drivers\rdr.sys
```

```
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
s3 (Video)                                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Scsiprnt (Extended base)                    Stopped   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Scsiscan (SCSI Class)                       Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Serial (Extended base)                      Stopped   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sermouse (Pointer Port)                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sfloppy (Primary disk)                      Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Simbad (Filter)                             Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
slcd32 (SCSI miniport)                      Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Sparrow (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Spock (SCSI miniport)                       Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Srv (Network)                               Running   (Manual)
   C:\WINNT\System32\drivers\srv.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
symc810 (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T128 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T13B (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
TCP/IP Service (PNP_TDI)                    Running   (Automatic)
   C:\WINNT\System32\drivers\tcpip.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
tga (Video)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
tmv1 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra124 (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra14f (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra24f (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
update (Base)                               Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
v7vram (Video)                              Stopped   (Disabled)
```

```
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaSave (Video Save)                          Running   (System)
   C:\WINNT\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaStart (Video Init)                         Stopped   (System)
   C:\WINNT\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Wd33c93 (SCSI miniport)                       Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
wd90c24a (Video)                              Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
wdvga (Video)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
weitekp9 (Video)                              Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Xga (Video)                                   Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process


IRQ and Port Report
-------------------------------------------------------------------------
Devices                         Vector Level  Affinity
-------------------------------------------------------------------------
FAfaSa                              32    32 0x00000000
FAfaSa                              40    40 0x00000000
FAfaSa                               4     4 0x00000000
FAfaSa                              12    12 0x00000000
FAfaSa                              20    20 0x00000000
FAfaSa                              28    28 0x00000000
MPS 1.4 - APIC platform              8     8 0x0000000f
MPS 1.4 - APIC platform              0     0 0x0000000f
MPS 1.4 - APIC platform              1     1 0x0000000f
MPS 1.4 - APIC platform              2     2 0x0000000f
MPS 1.4 - APIC platform              3     3 0x0000000f
MPS 1.4 - APIC platform              4     4 0x0000000f
MPS 1.4 - APIC platform              5     5 0x0000000f
MPS 1.4 - APIC platform              6     6 0x0000000f
MPS 1.4 - APIC platform              7     7 0x0000000f
MPS 1.4 - APIC platform              8     8 0x0000000f
MPS 1.4 - APIC platform              9     9 0x0000000f
MPS 1.4 - APIC platform             10    10 0x0000000f
MPS 1.4 - APIC platform             11    11 0x0000000f
MPS 1.4 - APIC platform             12    12 0x0000000f
MPS 1.4 - APIC platform             13    13 0x0000000f
MPS 1.4 - APIC platform             14    14 0x0000000f
MPS 1.4 - APIC platform             15    15 0x0000000f
MPS 1.4 - APIC platform             16    16 0x0000000f
MPS 1.4 - APIC platform             17    17 0x0000000f
MPS 1.4 - APIC platform             18    18 0x0000000f
MPS 1.4 - APIC platform             19    19 0x0000000f
MPS 1.4 - APIC platform             20    20 0x0000000f
MPS 1.4 - APIC platform             21    21 0x0000000f
MPS 1.4 - APIC platform             22    22 0x0000000f
MPS 1.4 - APIC platform             23    23 0x0000000f
MPS 1.4 - APIC platform             24    24 0x0000000f
MPS 1.4 - APIC platform             25    25 0x0000000f
MPS 1.4 - APIC platform             26    26 0x0000000f
MPS 1.4 - APIC platform             27    27 0x0000000f
MPS 1.4 - APIC platform             28    28 0x0000000f
MPS 1.4 - APIC platform             29    29 0x0000000f
MPS 1.4 - APIC platform             30    30 0x0000000f
MPS 1.4 - APIC platform             31    31 0x0000000f
MPS 1.4 - APIC platform             32    32 0x0000000f
MPS 1.4 - APIC platform             33    33 0x0000000f
MPS 1.4 - APIC platform             34    34 0x0000000f
```

```
MPS 1.4 - APIC platform           35    35 0x0000000f
MPS 1.4 - APIC platform           36    36 0x0000000f
MPS 1.4 - APIC platform           37    37 0x0000000f
MPS 1.4 - APIC platform           38    38 0x0000000f
MPS 1.4 - APIC platform           39    39 0x0000000f
MPS 1.4 - APIC platform           40    40 0x0000000f
MPS 1.4 - APIC platform           41    41 0x0000000f
MPS 1.4 - APIC platform           42    42 0x0000000f
MPS 1.4 - APIC platform           43    43 0x0000000f
MPS 1.4 - APIC platform           44    44 0x0000000f
MPS 1.4 - APIC platform           45    45 0x0000000f
MPS 1.4 - APIC platform           46    46 0x0000000f
MPS 1.4 - APIC platform           47    47 0x0000000f
MPS 1.4 - APIC platform           61    61 0x0000000f
MPS 1.4 - APIC platform           65    65 0x0000000f
MPS 1.4 - APIC platform           80    80 0x0000000f
MPS 1.4 - APIC platform          193   193 0x0000000f
MPS 1.4 - APIC platform          225   225 0x0000000f
MPS 1.4 - APIC platform          253   253 0x0000000f
MPS 1.4 - APIC platform          254   254 0x0000000f
MPS 1.4 - APIC platform          255   255 0x0000000f
i8042prt                           1     1 0xffffffff
i8042prt                          12    12 0xffffffff
E100B                             24    24 0x00000000
Floppy                             6     6 0x00000000
aic78u2                           16    16 0x00000000
aic78u2                           24    24 0x00000000
aic78xx                           32    32 0x00000000
------------------------------------------------------------------------
Devices                        Physical Address  Length
------------------------------------------------------------------------
FAfaSa                          0x0000d400  0x0000000100
FAfaSa                          0x0000dc00  0x0000000100
FAfaSa                          0x0000fc00  0x0000000100
FAfaSa                          0x0000f900  0x0000000100
FAfaSa                          0x0000f800  0x0000000100
FAfaSa                          0x0000f400  0x0000000100
MPS 1.4 - APIC platform         0x00000000  0x0000000010
MPS 1.4 - APIC platform         0x00000020  0x0000000002
MPS 1.4 - APIC platform         0x00000040  0x0000000004
MPS 1.4 - APIC platform         0x00000048  0x0000000004
MPS 1.4 - APIC platform         0x00000061  0x0000000001
MPS 1.4 - APIC platform         0x00000070  0x0000000002
MPS 1.4 - APIC platform         0x00000080  0x0000000010
MPS 1.4 - APIC platform         0x00000092  0x0000000001
MPS 1.4 - APIC platform         0x000000a0  0x0000000002
MPS 1.4 - APIC platform         0x000000c0  0x0000000010
MPS 1.4 - APIC platform         0x000000f0  0x0000000010
i8042prt                        0x00000060  0x0000000001
i8042prt                        0x00000064  0x0000000001
E100B                           0x0000d8e0  0x000000001e
Floppy                          0x000003f0  0x0000000006
Floppy                          0x000003f7  0x0000000001
aic78u2                         0x0000ec00  0x0000000100
aic78u2                         0x0000e800  0x0000000100
aic78xx                         0x0000e400  0x0000000100
ati                             0x000003b0  0x000000000c
ati                             0x000003c0  0x0000000020
ati                             0x000003c4  0x0000000002
ati                             0x000003c5  0x0000000001
ati                             0x000003ce  0x0000000002
ati                             0x000003cf  0x0000000001
ati                             0x00002000  0x0000000100
VgaSave                         0x000003b0  0x000000000c
VgaSave                         0x000003c0  0x0000000020
VgaSave                         0x000001ce  0x0000000002


DMA and Memory Report
------------------------------------------------------------------------
Devices                        Channel    Port
------------------------------------------------------------------------
Floppy                             2     0
```

```
------------------------------------------------------------------------
Devices                          Physical Address  Length
------------------------------------------------------------------------
FAfaSa                           0xf8102000  0x00002000
FAfaSa                           0xf81fe000  0x00002000
FAfaSa                           0xfe0fe000  0x00002000
FAfaSa                           0xfe0fc000  0x00002000
FAfaSa                           0xfe0fa000  0x00002000
FAfaSa                           0xfe0f8000  0x00002000
MPS 1.4 - APIC platform          0xfec00000  0x00000400
MPS 1.4 - APIC platform          0xfee00000  0x00000400
E100B                            0xf6000000  0x0000001e
aic78u2                          0xfb002000  0x00001000
aic78u2                          0xfb001000  0x00001000
aic78xx                          0xfb000000  0x00001000
ati                              0x000a0000  0x00020000
ati                              0xf5000000  0x00800000
VgaSave                          0x000a0000  0x00020000


Environment Report
------------------------------------------------------------------------



System Environment Variables
   ComSpec=C:\WINNT\system32\cmd.exe
   HOME=C:/
   NUMBER_OF_PROCESSORS=4
   OS=Windows_NT
   Os2LibPath=C:\WINNT\system32\os2\dll;

Path=C:\mks\mksnt;C:\WINNT\system32;C:\WINNT;C:\PROGRA~1\PERC2\System;D:\mstpcc.401\setup;
D:\MSSQL7\BINN
   PROCESSOR_ARCHITECTURE=x86
   PROCESSOR_IDENTIFIER=x86 Family 6 Model 7 Stepping 2, GenuineIntel
   PROCESSOR_LEVEL=6
   PROCESSOR_REVISION=0702
   ROOTDIR=C:/mks
   SHELL=C:/mks/mksnt/sh.exe
   TMPDIR=C:/TEMP
   windir=C:\WINNT



Environment Variables for Current User

include=d:\devstudio\vc\include;d:\devstudio\vc\atl\include;d:\devstudio\vc\mfc\include;d:
\devstudio\vc\include;d:\devstudio\vc\atl\include;d:\devstudio\vc\mfc\include;%include%

lib=d:\devstudio\vc\lib;d:\devstudio\vc\mfc\lib;d:\devstudio\vc\lib;d:\devstudio\vc\mfc\li
b;%lib%
   MSDevDir=d:\DevStudio\SharedIDE
   path=d:\devstudio\sharedide\bin\ide;d:\devstudio\sharedide\bin;d:\devstudio\vc\bin
   TEMP=C:\TEMP
   TMP=C:\TEMP



Network Report
------------------------------------------------------------------------
Your Access Level: Admin & Local
Workgroup or Domain: TPCGROUP
Network Version: 4.0
LanRoot: TPCGROUP
Logged On Users: 1
Current User (1): Administrator
  Logon Domain: PE6300
  Logon Server: PE6300

Transport: NetBT_E100B1, 00-A0-C9-D5-DF-9D, VC's: 0, Wan: Wan

Character Wait: 3,600
```

Appendix C – Tunable Parameters

```
Collection Time: 250
Maximum Collection Count: 16
Keep Connection: 600
Maximum Commands: 5
Session Time Out: 45
Character Buffer Size: 512
Maximum Threads: 17
Lock Quota: 6,144
Lock Increment: 10
Maximum Locks: 500
Pipe Increment: 10
Maximum Pipes: 500
Cache Time Out: 40
Dormant File Limit: 45
Read Ahead Throughput: 4,294,967,295
Mailslot Buffers: 3
Server Announce Buffers: 20
Illegal Datagrams: 5
Datagram Reset Frequency: 60
Log Election Packets: False
Use Opportunistic Locking: True
Use Unlock Behind: True
Use Close Behind: True
Buffer Pipes: True
Use Lock, Read, Unlock: True
Use NT Caching: True
Use Raw Read: True
Use Raw Write: True
Use Write Raw Data: True
Use Encryption: True
Buffer Deny Write Files: True
Buffer Read Only Files: True
Force Core Creation: True
512 Byte Max Transfer: False
Bytes Received: 0
SMB's Received: 0
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 0
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Bytes Transmitted: 0
SMB's Transmitted: 0
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 0
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Initally Failed Operations: 0
Failed Completion Operations: 0
Read Operations: 0
Random Read Operations: 0
Read SMB's: 0
Large Read SMB's: 0
Small Read SMB's: 0
Write Operations: 0
Random Write Operations: 0
Write SMB's: 0
Large Write SMB's: 0
Small Write SMB's: 0
Raw Reads Denied: 0
Raw Writes Denied: 0
Network Errors: 0
Sessions: 0
Failed Sessions: 0
Reconnects: 0
Core Connects: 0
LM 2.0 Connects: 0
LM 2.x Connects: 0
Windows NT Connects: 0
Server Disconnects: 0
Hung Sessions: 0
Use Count: 0
Failed Use Count: 0
Current Commands: 0
Server File Opens: 1
```

```
Server Device Opens: 0
Server Jobs Queued: 0
Server Session Opens: 1
Server Sessions Timed Out: 0
Server Sessions Errored Out: 0
Server Password Errors: 0
Server Permission Errors: 0
Server System Errors: 0
Server Bytes Sent: 50,125,762
Server Bytes Received: 1,472,410
Server Average Response Time: 0
Server Request Buffers Needed: 0
Server Big Buffers Needed: 0
```

Appendix C – Tunable Parameters

Client Configuration Parameters

Microsoft Windows NT Server Version 4.0 Configuration Parameters

No Windows NT parameters were changed on the client machines.

The following services were disabled:

- Computer Browser
- License Logging Service
- Messenger
- Net logon
- NT LM Security Support Provider
- Plug and Play
- RPC Locator
- Schedule
- Spooler
- TCP/IP Netbios Helper
- UPS

Appendix C – Tunable Parameters

TPCC Application Registry Parameters

```
Key Name:           SOFTWARE\TPCC
Class Name:         <NO CLASS>
Last Write Time:    3/20/99 - 5:33 PM
Value 0
  Name:             DatabaseName
  Type:             REG_SZ
  Data:             tpcc

Value 1
  Name:             DatabasePassword
  Type:             REG_SZ
  Data:

Value 2
  Name:             DatabaseServer
  Type:             REG_SZ
  Data:             pe6300

Value 3
  Name:             DatabaseUser
  Type:             REG_SZ
  Data:             sa

Value 4
  Name:             DLLPath
  Type:             REG_SZ
  Data:             http:/scripts/tpcc/tpcc.dll

Value 5
  Name:             LogPath
  Type:             REG_EXPAND_SZ
  Data:             d:\logs

Value 6
  Name:             MaxUsersThisClient
  Type:             REG_DWORD
  Data:             0x2710

Value 7
  Name:             NumberOfWarehousesTotal
  Type:             REG_DWORD
  Data:             0x76c
```

Microsoft Internet Information Server Registry Parameters

```
Key Name:           SYSTEM\CurrentControlSet\Services\InetInfo\Parameters
Class Name:         <NO CLASS>
Last Write Time:    3/20/99 - 5:34 PM
Value 0
  Name:             BandwidthLevel
  Type:             REG_DWORD
  Data:             0xffffffff

Value 1
  Name:             ListenBackLog
  Type:             REG_DWORD
  Data:             0x800

Value 2
  Name:             PoolThreadLimit
  Type:             REG_DWORD
  Data:             0x180

Value 3
  Name:             ThreadTimeout
```

```
  Type:            REG_DWORD
  Data:            0x1c20


Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\Filter


Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\Filter
(IIS 3.0 defaults)

Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\MimeMap
(IIS 3.0 defaults)
```

World Wide Web Service Registry Parameters

```
Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Parameters
Class Name:        <NO CLASS>
Last Write Time:   3/20/99 - 5:37 PM
Value 0
  Name:            AcceptExOutstanding
  Type:            REG_DWORD
  Data:            0x800

Value 1
  Name:            AccessDeniedMessage
  Type:            REG_SZ
  Data:            Error: Access is Denied.

Value 2
  Name:            AdminEmail
  Type:            REG_SZ
  Data:            Admin@corp.com

Value 3
  Name:            AdminName
  Type:            REG_SZ
  Data:            Administrator

Value 4
  Name:            AnonymousUserName
  Type:            REG_SZ
  Data:            IUSR_CLIENT6

Value 5
  Name:            Authorization
  Type:            REG_DWORD
  Data:            0x5

Value 6
  Name:            CacheExtensions
  Type:            REG_DWORD
  Data:            0x1

Value 7
  Name:            CheckForWAISDB
  Type:            REG_DWORD
  Data:            0

Value 8
  Name:            ConnectionTimeOut
  Type:            REG_DWORD
  Data:            0x1c20

Value 9
  Name:            DebugFlags
  Type:            REG_DWORD
  Data:            0x8

Value 10
  Name:            Default Load File
  Type:            REG_SZ
```

```
  Data:            Default.htm

Value 11
  Name:            Dir Browse Control
  Type:            REG_DWORD
  Data:            0x4000001e

Value 12
  Name:            Filter DLLs
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv\sspifilt.dll

Value 13
  Name:            GlobalExpire
  Type:            REG_DWORD
  Data:            0xffffffff

Value 14
  Name:            InstallPath
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv

Value 15
  Name:            LogFileDirectory
  Type:            REG_EXPAND_SZ
  Data:            D:\logs

Value 16
  Name:            LogFileFormat
  Type:            REG_DWORD
  Data:            0

Value 17
  Name:            LogFilePeriod
  Type:            REG_DWORD
  Data:            0x1

Value 18
  Name:            LogFileTruncateSize
  Type:            REG_DWORD
  Data:            0x1388000

Value 19
  Name:            LogSqlDataSource
  Type:            REG_SZ
  Data:            HTTPLOG

Value 20
  Name:            LogSqlPassword
  Type:            REG_SZ
  Data:            sqllog

Value 21
  Name:            LogSqlTableName
  Type:            REG_SZ
  Data:            Internetlog

Value 22
  Name:            LogSqlUserName
  Type:            REG_SZ
  Data:            InternetAdmin

Value 23
  Name:            LogType
  Type:            REG_DWORD
  Data:            0

Value 24
  Name:            MajorVersion
  Type:            REG_DWORD
  Data:            0x2

Value 25
  Name:            MaxConnections
```

```
  Type:            REG_DWORD
  Data:            0x186a0

Value 26
  Name:            MinorVersion
  Type:            REG_DWORD
  Data:            0

Value 27
  Name:            NTAuthenticationProviders
  Type:            REG_SZ
  Data:            NTLM

Value 28
  Name:            ScriptTimeout
  Type:            REG_DWORD
  Data:            0x384

Value 29
  Name:            SecurePort
  Type:            REG_DWORD
  Data:            0x1bb

Value 30
  Name:            ServerSideIncludesEnabled
  Type:            REG_DWORD
  Data:            0x1

Value 31
  Name:            ServerSideIncludesExtension
  Type:            REG_SZ
  Data:            .stm


Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\ADCLaunch
Class Name:        <NO CLASS>
Last Write Time:   3/20/99 - 5:12 PM

Key Name:
SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\ADCLaunch\AdvancedDataFactory
Class Name:        <NO CLASS>
Last Write Time:   3/20/99 - 5:12 PM

Key Name:
SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\ADCLaunch\RDSServer.DataFactory
Class Name:        <NO CLASS>
Last Write Time:   3/20/99 - 5:12 PM

Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Script Map
Class Name:        <NO CLASS>
Last Write Time:   3/20/99 - 5:52 AM
Value 0
  Name:            .idc
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv\httpodbc.dll


Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots
Class Name:        <NO CLASS>
Last Write Time:   3/20/99 - 5:12 PM
Value 0
  Name:            /
  Type:            REG_SZ
  Data:            C:\InetPub\wwwroot,,1

Value 1
  Name:            /iisadmin
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv\iisadmin,,1

Value 2
  Name:            /MSADC
  Type:            REG_SZ
  Data:            C:\Program Files\Common Files\System\MSADC,,5
```

```
Value 3
  Name:            /Scripts
  Type:            REG_SZ
  Data:            C:\InetPub\scripts,,4
```

Tuxedo Registry Parameters

```
Key Name:          SOFTWARE\BEA Systems\TUXEDO\6.3\IPCResources\tpcc
Class Name:        <NO CLASS>
Last Write Time:   3/20/99 - 5:32 PM
Value 0
  Name:            TUXIPC_MSG_BYTES
  Type:            REG_DWORD
  Data:            0x10000

Value 1
  Name:            TUXIPC_MSG_HDRS
  Type:            REG_DWORD
  Data:            0x1fc0

Value 2
  Name:            TUXIPC_MSG_QUEUE_BYTES
  Type:            REG_DWORD
  Data:            0x100000

Value 3
  Name:            TUXIPC_MSG_QUEUES
  Type:            REG_DWORD
  Data:            0x400

Value 4
  Name:            TUXIPC_MSG_SEG_BYTES
  Type:            REG_DWORD
  Data:            0x40

Value 5
  Name:            TUXIPC_MSG_SEGS
  Type:            REG_DWORD
  Data:            0x7fff

Value 6
  Name:            TUXIPC_PROC
  Type:            REG_DWORD
  Data:            0x400

Value 7
  Name:            TUXIPC_SEM
  Type:            REG_DWORD
  Data:            0x400

Value 8
  Name:            TUXIPC_SEM_IDS
  Type:            REG_DWORD
  Data:            0x400

Value 9
  Name:            TUXIPC_SEM_UNDO
  Type:            REG_DWORD
  Data:            0x400

Value 10
  Name:            TUXIPC_SHM_PROCS
  Type:            REG_DWORD
  Data:            0x400

Value 11
  Name:            TUXIPC_SHM_SEGS
  Type:            REG_DWORD
  Data:            0x32
```

Appendix C – Tunable Parameters

Tuxedo Configuration File

```
#ident  "@(#)apps:simpapp/ubbsimple   60.3"
#       tux.ubb: Tuxedo 6.3 configuration File - one file for each client
#       Dell Computer Corp.
#       modified by James Jordan
#
#       Environment Variables that must be set at system level:
#       TMCONTEXTS=1
#       TUXDIR=c:\tuxedo
#       TUXCONFIG=c:\tpcctux\tuxconfig
#
#       to load text config file (this file) to binary tuxedo config file:
#         run-> c:\tuxedo\bin\tmloadcf -y < tux.ubb
#       to boot servers:
#         run-> c:\tuxedo\bin\tmboot -y
#       to shutdown servers:
#         run-> c:\tuxedo\bin\tmshutdown -y
#

*RESOURCES
IPCKEY          133133

DOMAINID        CLIENT1
MASTER          CLIENT1
MAXACCESSERS    800
MAXSERVERS      65
MAXSERVICES     230
MODEL           SHM
LDBAL           N
SCANUNIT        60
BLOCKTIME       15
BBLQUERY        60
SANITYSCAN      240

*MACHINES
DEFAULT:

"CLIENT1"        LMID=CLIENT1
                 APPDIR="d:\tpcctux"
                 TUXCONFIG="d:\tpcctux\tuxconfig"
                 TUXDIR="d:\tuxedo"
                 ULOGPFX="d:\logs\tuxserv_ULOG"
                 TYPE="WinNT"
                 UID= 0
                 GID= 0
*GROUPS
GROUPT4
        LMID=CLIENT1   GRPNO=1 OPENINFO=NONE
GROUPDL
        LMID=CLIENT1   GRPNO=2 OPENINFO=NONE

*SERVERS
DEFAULT:
TMT4             SRVGRP=GROUPT4 SRVID=100
                MIN=52 MAX=55
                CLOPT="-A"
                RQADDR=t4q REPLYQ=Y

TMDL             SRVGRP=GROUPDL SRVID=200
                MIN=5 MAX=10
                CLOPT="-A"
                RQADDR=deliveryq REPLYQ=N

*SERVICES
```

Appendix C – Tunable Parameters

Microsoft Diagnostics Report For PE2300

```
Microsoft Diagnostics Report For \\CLIENT6
------------------------------------------------------------------------

OS Version Report
------------------------------------------------------------------------
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381: Service Pack 4) x86 Multiprocessor Free
Registered Owner: SPA, Dell Computer Corporation
Product Number: 01296-OEM-0123456-01234
------------------------------------------------------------------------

System Report
------------------------------------------------------------------------
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: MPS 1.4 - APIC platform
BIOS Date: 05/05/98
BIOS Version: Rage Pro v3.071
              Phoenix ROM BIOS

Processor list:
   0:  x86 Family 6 Model 5 Stepping 1 GenuineIntel ~397 Mhz
   1:  x86 Family 6 Model 5 Stepping 1 GenuineIntel ~397 Mhz
------------------------------------------------------------------------

Video Display Report
------------------------------------------------------------------------
BIOS Date: <unavailable>
Adapter:
   Setting: 1024 x 768 x 256
            72 Hz
   Type: ati compatible display adapter
   String: ATI Graphics Accelerator
   Memory: 2 MB
   Chip Type: ATI 3D RAGE PRO AGP (GT-C2U2)
   DAC Type: ATI Internal DAC
Driver:
   Vendor: ATI Technologies Inc.
   File(s): ati.sys, ati.dll, 8514a.dll
   Version: 4.3.92, 4.0.0


Drives Report
------------------------------------------------------------------------
C:\  (Local - FAT)  Total: 1,019,824 KB, Free: 354,016 KB
D:\  (Local - NTFS) NTFS Total: 3,172,836 KB, Free: 2,536,564 KB

Memory Report
------------------------------------------------------------------------
Handles: 27,428
Threads: 545
Processes: 71

Physical Memory (K)
   Total: 523,700
   Available: 245,036
   File Cache: 16,244



Services Report
------------------------------------------------------------------------
Alerter                                    Running   (Automatic)
EventLog (Event log)                       Running   (Automatic)
Server                                     Running   (Automatic)
Workstation (NetworkProvider)              Running   (Automatic)
NT LM Security Support Provider            Running   (Manual)
Remote Procedure Call (RPC) Service        Running   (Automatic)
```

```
TUXEDO IPC Helper                                Running   (Automatic)
World Wide Web Publishing Service                Running   (Automatic)


Drivers Report
--------------------------------------------------------------------------
AFD Networking Support Environment (TDI)         Running   (Automatic)
aic78u2 (SCSI miniport)                          Running   (Boot)
aic78xx (SCSI miniport)                          Running   (Boot)
ati (Video)                                      Running   (System)
Beep (Base)                                      Running   (System)
Cdfs (File system)                               Running   (Disabled)
Cdrom (SCSI CDROM Class)                         Running   (System)
Disk (SCSI Class)                                Running   (Boot)
Intel EtherExpress PRO Adapter (NDIS)            Running   (Automatic)
Fastfat (Boot file system)                       Running   (Disabled)
Floppy (Primary disk)                            Running   (System)
i8042 Keyboard and PS/2 Mouse Port Driver (Keyboard Port) Running   (System)
Keyboard Class Driver (Keyboard Class)           Running   (System)
KSecDD (Base)                                    Running   (System)
Mouse Class Driver (Pointer Class)               Running   (System)
Msfs (File system)                               Running   (System)
Mup (Network)                                    Running   (Manual)
Microsoft NDIS System Driver (NDIS)              Running   (System)
WINS Client(TCP/IP) (PNP_TDI)                    Running   (Automatic)
Npfs (File system)                               Running   (System)
Ntfs (File system)                               Running   (Disabled)
Null (Base)                                      Running   (System)
Parallel (Extended base)                         Running   (Automatic)
Parport (Parallel arbitrator)                    Running   (Automatic)
ParVdm (Extended base)                           Running   (Automatic)
Rdr (Network)                                    Running   (Manual)
Scsiscan (SCSI Class)                            Running   (System)
Serial (Extended base)                           Running   (Automatic)
Srv (Network)                                    Running   (Manual)
TCP/IP Service (PNP_TDI)                         Running   (Automatic)
VgaSave (Video Save)                             Running   (System)


IRQ and Port Report
--------------------------------------------------------------------------
Devices                          Vector Level  Affinity
--------------------------------------------------------------------------
MPS 1.4 - APIC platform             8     8 0x00000003
MPS 1.4 - APIC platform             0     0 0x00000003
MPS 1.4 - APIC platform             1     1 0x00000003
MPS 1.4 - APIC platform             2     2 0x00000003
MPS 1.4 - APIC platform             3     3 0x00000003
MPS 1.4 - APIC platform             4     4 0x00000003
MPS 1.4 - APIC platform             5     5 0x00000003
MPS 1.4 - APIC platform             6     6 0x00000003
MPS 1.4 - APIC platform             7     7 0x00000003
MPS 1.4 - APIC platform             8     8 0x00000003
MPS 1.4 - APIC platform             9     9 0x00000003
MPS 1.4 - APIC platform            10    10 0x00000003
MPS 1.4 - APIC platform            11    11 0x00000003
MPS 1.4 - APIC platform            12    12 0x00000003
MPS 1.4 - APIC platform            13    13 0x00000003
MPS 1.4 - APIC platform            14    14 0x00000003
MPS 1.4 - APIC platform            15    15 0x00000003
MPS 1.4 - APIC platform            16    16 0x00000003
MPS 1.4 - APIC platform            17    17 0x00000003
MPS 1.4 - APIC platform            18    18 0x00000003
MPS 1.4 - APIC platform            19    19 0x00000003
MPS 1.4 - APIC platform            20    20 0x00000003
MPS 1.4 - APIC platform            21    21 0x00000003
MPS 1.4 - APIC platform            22    22 0x00000003
MPS 1.4 - APIC platform            23    23 0x00000003
MPS 1.4 - APIC platform            24    24 0x00000003
MPS 1.4 - APIC platform            25    25 0x00000003
MPS 1.4 - APIC platform            26    26 0x00000003
MPS 1.4 - APIC platform            27    27 0x00000003
```

```
MPS 1.4 - APIC platform          28    28 0x00000003
MPS 1.4 - APIC platform          29    29 0x00000003
MPS 1.4 - APIC platform          30    30 0x00000003
MPS 1.4 - APIC platform          31    31 0x00000003
MPS 1.4 - APIC platform          32    32 0x00000003
MPS 1.4 - APIC platform          33    33 0x00000003
MPS 1.4 - APIC platform          34    34 0x00000003
MPS 1.4 - APIC platform          35    35 0x00000003
MPS 1.4 - APIC platform          36    36 0x00000003
MPS 1.4 - APIC platform          37    37 0x00000003
MPS 1.4 - APIC platform          38    38 0x00000003
MPS 1.4 - APIC platform          39    39 0x00000003
MPS 1.4 - APIC platform          40    40 0x00000003
MPS 1.4 - APIC platform          41    41 0x00000003
MPS 1.4 - APIC platform          42    42 0x00000003
MPS 1.4 - APIC platform          43    43 0x00000003
MPS 1.4 - APIC platform          44    44 0x00000003
MPS 1.4 - APIC platform          45    45 0x00000003
MPS 1.4 - APIC platform          46    46 0x00000003
MPS 1.4 - APIC platform          47    47 0x00000003
MPS 1.4 - APIC platform          61    61 0x00000003
MPS 1.4 - APIC platform          65    65 0x00000003
MPS 1.4 - APIC platform          80    80 0x00000003
MPS 1.4 - APIC platform         193   193 0x00000003
MPS 1.4 - APIC platform         225   225 0x00000003
MPS 1.4 - APIC platform         253   253 0x00000003
MPS 1.4 - APIC platform         254   254 0x00000003
MPS 1.4 - APIC platform         255   255 0x00000003
i8042prt                          1     1 0xffffffff
i8042prt                         12    12 0xffffffff
Serial                            4     4 0x00000000
Serial                            3     3 0x00000000
E100B                            40    40 0x00000000
E100B                            48    48 0x00000000
E100B                            56    56 0x00000000
E100B                            40    40 0x00000000
E100B                            52    52 0x00000000
E100B                            32    32 0x00000000
Floppy                            6     6 0x00000000
aic78u2                          16    16 0x00000000
aic78xx                          24    24 0x00000000
------------------------------------------------------------------------
Devices                       Physical Address  Length
------------------------------------------------------------------------
MPS 1.4 - APIC platform         0x00000000  0x0000000010
MPS 1.4 - APIC platform         0x00000020  0x0000000002
MPS 1.4 - APIC platform         0x00000040  0x0000000004
MPS 1.4 - APIC platform         0x00000048  0x0000000004
MPS 1.4 - APIC platform         0x00000061  0x0000000001
MPS 1.4 - APIC platform         0x00000070  0x0000000002
MPS 1.4 - APIC platform         0x00000080  0x0000000010
MPS 1.4 - APIC platform         0x00000092  0x0000000001
MPS 1.4 - APIC platform         0x000000a0  0x0000000002
MPS 1.4 - APIC platform         0x000000c0  0x0000000010
MPS 1.4 - APIC platform         0x000000f0  0x0000000010
i8042prt                        0x00000060  0x0000000001
i8042prt                        0x00000064  0x0000000001
Parport                         0x00000378  0x0000000003
Serial                          0x000003f8  0x0000000007
Serial                          0x000002f8  0x0000000007
E100B                           0x0000dcc0  0x000000001c
E100B                           0x0000dca0  0x000000001c
E100B                           0x0000dc80  0x000000001c
E100B                           0x0000e4e0  0x000000001c
E100B                           0x0000e4c0  0x000000001c
E100B                           0x0000dce0  0x000000001c
Floppy                          0x000003f0  0x0000000006
Floppy                          0x000003f7  0x0000000001
aic78u2                         0x0000ec00  0x0000000100
aic78xx                         0x0000e800  0x0000000100
ati                             0x000003b0  0x000000000c
ati                             0x000003c0  0x0000000020
ati                             0x000003c4  0x0000000002
ati                             0x000003c5  0x0000000001
```

```
ati                             0x000003ce  0x0000000002
ati                             0x000003cf  0x0000000001
ati                             0x0000fc00  0x0000000100
VgaSave                         0x000003b0  0x000000000c
VgaSave                         0x000003c0  0x0000000020
VgaSave                         0x000001ce  0x0000000002


DMA and Memory Report
-------------------------------------------------------------------------
Devices                         Channel    Port
-------------------------------------------------------------------------
Floppy                             2     0
-------------------------------------------------------------------------
Devices                         Physical Address  Length
-------------------------------------------------------------------------
MPS 1.4 - APIC platform         0xfec00000  0x00000400
MPS 1.4 - APIC platform         0xfee00000  0x00000400
E100B                           0xf4002000  0x0000001c
E100B                           0xf4001000  0x0000001c
E100B                           0xf4000000  0x0000001c
E100B                           0xf2fff000  0x0000001c
E100B                           0xf2ffe000  0x0000001c
E100B                           0xf4003000  0x0000001c
aic78u2                         0xf9fff000  0x00001000
aic78xx                         0xf9ffe000  0x00001000
ati                             0x000a0000  0x00020000
ati                             0xfc000000  0x00800000
VgaSave                         0x000a0000  0x00020000


Environment Report
-------------------------------------------------------------------------


System Environment Variables
   ComSpec=C:\WINNT\system32\cmd.exe
   HOME=D:/
   NUMBER_OF_PROCESSORS=2
   OS=Windows_NT
   Os2LibPath=C:\WINNT\system32\os2\dll;
   Path=d:\mks\mksnt;C:\WINNT\system32;C:\WINNT;d:\TUXEDO\bin;d:\MSSQL7\BINN;
   PROCESSOR_ARCHITECTURE=x86
   PROCESSOR_IDENTIFIER=x86 Family 6 Model 5 Stepping 1, GenuineIntel
   PROCESSOR_LEVEL=6
   PROCESSOR_REVISION=0501
   ROOTDIR=d:/mks
   SHELL=d:/mks/mksnt/sh.exe
   TMCONTEXTS=1
   TMPDIR=C:/TEMP
   TUXCONFIG=D:\TPCCTUX\TUXCONFIG
   TUXDIR=D:\TUXEDO
   windir=C:\WINNT


Environment Variables for Current User
   TEMP=C:\TEMP
   TMP=C:\TEMP


Network Report
-------------------------------------------------------------------------
Your Access Level: Admin & Local
Workgroup or Domain: TPCGROUP
Network Version: 4.0
LanRoot: TPCGROUP
Logged On Users: 1
Current User (1): Administrator
  Logon Domain: CLIENT6
```

```
  Logon Server: CLIENT6

Transport: NetBT_E100B2, 00-A0-C9-D1-C4-CF, VC's: 0, Wan: Wan
Transport: NetBT_E100B3, 00-A0-C9-D1-C4-CE, VC's: 0, Wan: Wan
Transport: NetBT_E100B4, 00-A0-C9-D1-80-00, VC's: 0, Wan: Wan
Transport: NetBT_E100B5, 00-A0-C9-D1-C4-D3, VC's: 0, Wan: Wan
Transport: NetBT_E100B6, 00-A0-C9-9D-BB-EB, VC's: 0, Wan: Wan
Transport: NetBT_E100B1, 00-A0-C9-D1-C4-D0, VC's: 0, Wan: Wan

Character Wait: 3,600
Collection Time: 250
Maximum Collection Count: 16
Keep Connection: 600
Maximum Commands: 5
Session Time Out: 45
Character Buffer Size: 512
Maximum Threads: 17
Lock Quota: 6,144
Lock Increment: 10
Maximum Locks: 500
Pipe Increment: 10
Maximum Pipes: 500
Cache Time Out: 40
Dormant File Limit: 45
Read Ahead Throughput: 4,294,967,295
Mailslot Buffers: 3
Server Announce Buffers: 20
Illegal Datagrams: 5
Datagram Reset Frequency: 60
```

Appendix C – Tunable Parameters

RTE Input Parameters

rcc_1875_15_5

```
rcc_1875_15_5
rte config file - 1875 Warehouses across 15 rteslave processes on 5 slaves
Client/Server
80 0 9000 5 1 2 1
/rte/trans_params.1
/scripts/tpcc/tpcc.dll
208 1875 1300000
*rte2 3
client4_4    1   125
client4_5  126   250
client5_4  251   375
*rte3 3
client5_5  376   500
client6_4  501   625
client6_5  626   750
*rte4 3
client4_1  751   875
client4_2  876  1000
client4_3 1001  1125
*rte5 3
client5_1 1126  1250
client5_2 1251  1375
client5_3 1376  1500
*rte6 3
client6_1 1501  1625
client6_2 1626  1750
client6_3 1751  1875
*
```

trans_params.1

```
NewOrder     44.86   12.08  18.00   0.10   5.00   0.10
Payment      43.035  12.08   3.00   0.10   5.00   0.10
OrderStatus   4.035  10.08   2.00   0.10   5.00   0.10
Delivery      4.035   5.08   2.00   0.10   5.00   0.10
StockLevel    4.035   5.08   2.00   0.10  20.00   0.10
```

Appendix D – Disk Storage

180 Day Space

TPC-C 180 Day Space Requirements						
Warehouses	**1900**				**TpmC**	**23400**
Table	**Rows**	**Data KB**	**Index KB**	**Extra 5% KB**	**8hr Space**	**Total Space KB**
Warehouse	1900	208	40	12		260
District	19000	2112	40	108		2260
Customer	57000000	2661672	104	133,089		2794865
History	57000000	3166680	16		935,797	3166696
NewOrder	17100000	270360	680			271040
Orders	57000000	1747128	964912		5,642,260	2712040
OrderLine	570001928	35625128	88664		9,267,773	35713792
Item	100000	9528	64	480		10072
Stock	190000000	60800000	136064	3,046,803		63982867
Total		104,282,816	1,190,584	3,180,492	15,845,830	108,653,892
DB File Group	**Count**	**Size MB**	**MB Allocated**	**MB Loaded+5%**	**Total for 8 hrs**	
Master,model,msdb	1	30	30	30	30	
mssql70_tpcc_root	1	8	8	8	8	
Mssql70_CS_FG	8	128,000	127,200	65,213	65,213	
Mssql70_misc_FG	8	65,600	64,000	40,895	56,369	
			191,238	106,145	121,620	
	MB					
Dynamic Space	39,589	Sum of Data for Order, Orderline and History				
Static Space	66,519	Sum of Data+Index+5%-Dynamic Space				
Free Space	85,131	Total Allocated Spac - (Dynamic + Static Space)				
Daily Growth	7,801	(Dynamic Space/(W*62.5))*tpmc				
Daily Spread	73,429	(Free Space -1.5*Daily Growth) Zero Assumed				
180 Day Space MB	1,470,713		18 GB Drive	17.300	GB	
180 Day Space GB	**1,436.24**	**GB**	9 GB Drive	8.474	GB	
Log Size	48,000	MB				
KB Per New Order	5.3596	KB				
8 hr log MB	58,788	MB				
8 hr log GB	**57.4098**	**GB**				
Space Usage	**GB Needed**	**Disks Priced**	**GB Priced**			
180 Day Space DB	1,436.24	8	138.40	18GB		

Appendix D – Disk Storage

		184	1559.15	9GB		
Total DB		**192**	**1697.55**	**GB**		
8-hr log + mirror	114.8196	0	0.00	GB		
OS, Swap	6	1	8.475	GB		
Total Storage	**1,557.06**	**GB**	**1,706.02**	**GB**		
Log Space OK						
Total Space OK						
				72		
0:00:00	Container	64KB:	609			
0:00:00	Container	609MB	500			
0:00:00	Free	1109MB	894			
0:00:00	Container	2004MB	6674			
			8677	MB		
			8.473632813	GB		

Appendix E - Price Quotations

Appendix E – Price Quotations

One Microsoft Way
Redmond, WA 98052-6399
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

May 10, 1999

Dave Jaffe
Dell Computer Corporation
1 Dell Way
Round Rock, TX 78682
(512) 728-4543
dave_jaffe@dell.com

Dear Mr. Jaffe:

Here is the information you requested regarding U.S. pricing of several Microsoft products that were used in a recent TPC-C benchmark:

Microsoft SQL Server 7.0, Enterprise Edition (one server plus unlimited CALs)	$28,999
Microsoft Windows NT Server 4.0, Enterprise Edition (one server plus 25 CALs)	$3,999
Windows NT Server 4.0 (one server plus 5 CALs)	$809
Visual C++ Professional 5.0 (single copy)	$499
5-year maintenance for above software @ $2095/yr	$10,475

This quote is valid for the next 60 days.

If I can be of any further assistance, please contact me at 425-936-5301 or tomkr@microsoft.com.

Yours truly,

Thomas Kreyche
Product Manager
SQL Server Marketing

03/23/99 TUE 16:24 FAX 408 570 8901 BEA SYSTEMS, INC. 002

BEA ENTERPRISE MIDDLEWARE SOLUTIONS

March 23, 1999

Mr. David Jaffe
Dell Computer
Austin, Texas
FAX: 512-728-4084

Dear Mr. Jaffe;

Per your request I am enclosing the pricing information regarding TUXEDO 6.x that you requested. This pricing applies to Tuxedo 6.1, 6.2, 6.3 and 6.4. Please note that Tuxedo 6.4 is our most recent version of Tuxedo but that all 6.x releases are generally available. Core functionality services pricing is appropriate for your activities. As per the table below, server systems are classified in one of 5 tiers based on CPU type and capacity. The Dell servers with 4 CPU capacity are tier 2 systems. This quote is valid for 90 days from the date of issue of this letter.

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

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.x. Prices range from $3,000 for Tier 1 to $250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

BEA Tux/CFS Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 8) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers (Class 1and Class 2)	Unlimited	$3,000.00	$480.00	$690.00
Tier 2 -- PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations (class 3)	Unlimited	$12,000.00	$1,920.00	$2,760.00
Tier 3 -- Midrange Multiprocessors, up to 8 CPUs per system capacity (Class 4 and 5)	Unlimited	$30,000.00	$4,800.00	$6,900.00
Tier 4 -- Large (more than 8, less than 32 CPUs) and Mainframe	Unlimited	$100,000.00	$16,000.00	$23,000.00

1

Appendix E – Price Quotations

Systems (Class 6)				
Tier 5 -- Massively Parallel Systems, > 32 processors	Unlimited	$250,000.00	$40,000.00	$57,500.00

Intel based server tier classifications:

Platform	Operating System	Tier 1	Tier 1	Tier 2	Tier 3	Tier 3
Intel Pentium/ Pentium Pro PCs	Interactive R3.2 ESIX SVR 4.0 SCO UNIX 3.2.2 and 3.2.4 SCO ODT 2.x,3.x Solaris x86 2.X UnixWare, Windows NT 3.5/4.0	All 386/486 PCs are Class 1	ALL Pentium and Pentium Pro PCs with 1 or 2 CPUs capacity are Tier 1	ALL Pentium and Pentium Pro PCs with 3 or 4 CPUs capacity are Tier 2		ALL Pentium and Pentium Pro PCs with 5,6,7, or 8 CPUs are Tier 3

Very Truly Yours,

Lewis D. Brentano,
Director, Market Planning

Appendix E – Price Quotations