TPC Benchmark™ C Full Disclosure Report Using Sun Microsystems Enterprise 4500 Server and Sybase Adaptive Server Enterprise 11.9.3 RDBMS

Sun Microsystems, Inc
901 San Antonio Road
Palo Alto, CA 94303
U.S.A.

Revision 1,November 1999
Submitted for review
Compliant with Revision 3.5 of the TPC-C specification

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark C™ test conducted on the Sun Enterprise 4500 Server system, running Sybase Adaptive Server Enterprise 11.9.3 RDBMS and BEA Systems, Inc. Tuxedo 6.3.

TPC Benchmark C Metrics

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

Executive Summary Statements

Pages v-vii contain the executive summary of the benchmark result for the Sun Microsystems Ultra Enterprise 4500.

First Printing

Sun Microsystems, Inc believes that the information in this document is accurate as of its publication date. The information in this document is subject to change without notice. Sun Microsystems, Inc assumes no responsibility for any errors that may appear in this document.

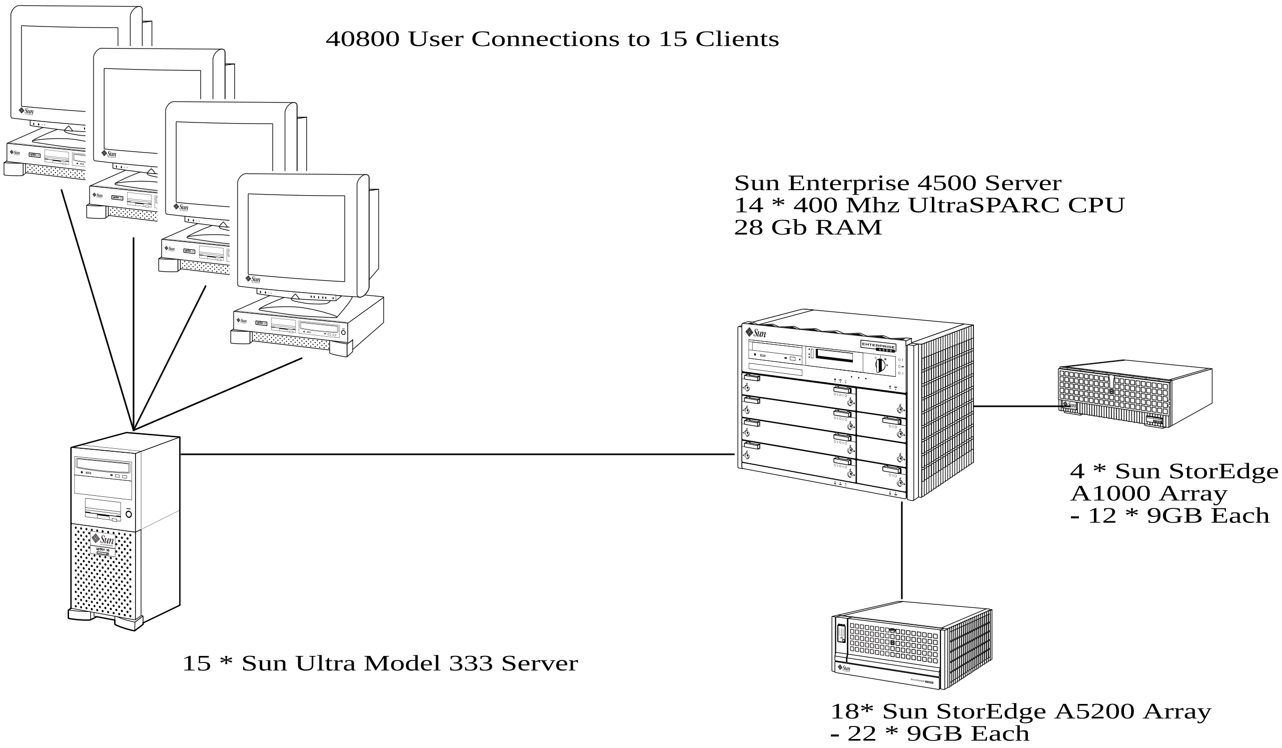
The pricing information in this document is believed to accurately reflect prices in effect on November 22, 1999. However, Sun Microsystems, Inc provides no warranty on the pricing information in this document.

The performance information in this document is for guidance only. System performance is highly dependent on many factors including system hardware, system and user software, and user application characteristics. Customer applications must be carefully evaluated before estimating performance. Sun Microsystems Computer Company does not warrant or represent that a user can or will achieve a similar performance expressed in tpmC or normalized price/performance ($/tpmC). No warranty on system performance or price/performance is expressed or implied in this document.

Sun **Sybase, Inc.**	**Sun Enterprise 4500 C/S w/15 Front-Ends**	TPC-C 3.5 Report Date: **November 22, 1999**

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$2,507,453	**50268.07**	**$49.88**	**March 30, 2000**

Processors	Database Manager	Operating System	Other Software	Number of Users
14 * 400MHz UltraSPARC II	**Sybase ASE 11.9.3 GA**	**Solaris 7**	**BEA Tuxedo 6.3**	**40800**


40800 User Connections to 15 Clients
Sun Enterprise 4500 Server
14 * 400 Mhz UltraSPARC CPU
28 Gb RAM
4 * Sun StorEdge
A1000 Array
- 12 * 9GB Each
15 * Sun Ultra Model 333 Server
18* Sun StorEdge A5200 Array
- 22 * 9GB Each

Configuration

	Server System	Front End Systems
Database Nodes:	1 Sun Enterprise 4500 Servers	15 * Ultra 10 Model 333
Processors	14* 400 MHz UltraSPARC II	1 * 333 MHz UltraSPARC II each
Cache memory	32KB (D+I), 8MB external	32KB (D+I), 512KB external, each
Main memory	28 GB	1 GB each
Disk controllers	3 Dualport FC-AL, 2 Fast/Wide SCSI-2	1 * SSCSI-2 each
Disk Drives	444 * 9GB FC-AL (18 * A5200, 4 * A1000), 6 * 9GB SCSI-2 (1 * MultiPack)	1 * 9 GB disk
Total Disk Storage	4050 GB	9 GB each
10 BaseT Hub	None	5100* 9-Port Hubs
100 Base T Hubs	None	None
100 BaseT Switches	1 * 16-Port	None

Sun **Sybase, Inc.**	**Sun Enterprise 4500 C/S w/15 Front-Ends**	TPC-C 3.5 **Report Date: November 22, 1999**

Pricing Summary

Description	Part Number	Source	Unit Price	Qty	Ext. Price	5 Yr. Maint.
Server Hardware						
E4500	E4501	2	23,360	1	23,360	30,967
CPU/Memory board	2602A	2	4,380	7	30,660	71,098
400MHz/8MB UltraSPARC II	2580A	2	10,950	14	153,300	
2GB memory (8 * 256MB)	7026A	2	12,045	14	168,630	
PS/300W for Ex000	954A	2	1,314	2	2,628	
SBus I/O board	2612A	2	4,745	1	4,745	
FCAL SBus Host adapter	6730A	2	1,971	2	3,942	
DWIS/S SBus Host Adapter	1062A	2	945	1	945	
200GB StorEdge A5200 Array	SG-XARY520A-200G	2	51,120	18	920,160	254,016
109-Gbyte StorEdge A1000 Array	SG-XARY144A-109G	2	18,036	4	72,142	40,320
54.6-Gbyte StorEdge MultiPack	SG-XDSK060C-54G	2	6512	1	6,512	
SCSI Cable 68 pin	3856A	2	40	1	40	
Opt Int Tape 12Gb 4MM	6283A	2	986	1	986	
Wyse Terminal	WYSE-WY55-A	1	430	1	430	
Server Hardware Subtotal					1,388,480	396,401
Server Software						
Solaris Server Software	SOLMS-26ZW9999	1	100	1	100	
SPARC Compiler C/C++ 5.0	WCCIS-500-T999	1	995	1	995	1,080
Sybase Adaptive Server Enterprise 11.9.3		3	295000	1	295,000	236,000
Sybase Development		3	9600	1	9,600	7,680
Sybase Open Client		3	795	1	795	636
Sybase Discount (10%)					(30,540)	
Server Software Subtotal					275,951	245,396
Client Hardware						
Ultra 10 Server Model 333	A22UHC1Z9S-B512CP	2	5,496	15	82,434	64,584
512MB Memory for Ultra 10	7039A	2	1,580	15	23,694	
PCI QEF Card	1034A	2	1,310	15	19,655	
Color Monitor	x7126	2	343	15	5,148	
Client Hardware Subtotal					130,931	64,584
Client Software						
BEA Tuxedo CFS 6.3		4	3,000	15	45,000	36,000
Client Software Subtotal					45,000	36,000
User Connectivity						
16-port 10/100Mbps Fast Ethernet Hub	Z179489	5	420	3	1,260	
9-port 10Mbps Ethernet Hub	Z179489	5	27	5610	151,470	
User Connectivity Subtotal					152,730	
Sun Enterprise Services discounts						(228,020)
			Total		1,993,092	514,361
			5Yr. cost		2,507,453	
			tpmC Rating		50,268	
			$/tpmC		$49.88	

Service for all Sun products is from Sun Microsystems, Inc.

Notes:

1. Sun Microsystems Inc. 2. CAT Technology Inc 3. Sybase 4. BEA Systems, Inc. 5. Software House Int.

Audited by: Francois Raab, Infosizing Inc.

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 purchase 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 the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.

Numerical Quantity Summary

MQTH, Computed Maximum Qualified Throughput tpmC	**= 50268.07**
% throughput difference, reported & reproducibility runs	= < 0.1%

Response Times (in secs)	**90th Percentile**	**Average**	**Maximum**
Menu	0.50	0.25	0.65
New-Order	0.80	0.34	24.14
Payment	1.00	0.47	25.52
Order-Status	1.00	0.51	21.63
Delivery(interactive)	0.26	0.25	0.35
Delivery(deferred)	2.00	0.64	5.00
Stock-level	2.00	0.67	5.44

Transaction Mix, in percent of total transactions

New-Order	44.85%
Payment	43.06%
Order-Status	4.02%
Delivery	4.04%
Stock-level	4.03%

Keying/Think Times (in secs)	**Average.**	**Min.**	**Maximum**
New-Order	18.02 / 12.20	18.01 / 0	18.17 / 122.00
Payment	3.02 / 12.20	3.01 / 0	3.13 / 122.00
Order-Status	2.02 / 10.25	2.01 / 0	2.14 / 102.50
Delivery	2.02 / 5.19	2.01 / 0	2.10 / 52.00
Stock-level	2.02 / 5.20	2.01 / 0	2.10 / 52.00

Test Duration	
Ramp-up time	25 minutes
Measurement Interval	60 minutes
Number of checkpoints	4
Checkpoint Interval	15 minutes
Number of transactions (all types) completed in measurement interval	6724854

Contents

Preface

This report documents the compliance of the Sun Microsystems TPC Benchmark ™C testing on the Enterprise 4500 Server running Sybase Adaptive Server Enterprise 11.9.3 with the *TPC Benchmark* ^{TM}*C Standard Revision 3.5.*

These tests were run using the Sybase Adaptive Server Enterprise 11.9.3 RDBMS running with Solaris 7 on the Enterprise 4500 Server and BEA Tuxedo 6.3 on the Ultra 10 Model 333 clients.

Document Structure

The *TPC Benchmark ™C Full Disclosure Report* is organized as follows:

- The main body of the document lists each item in Clause 8 of the TPC Benchmark ™C Standard and explains how each specification is satisfied.
- Appendix A contains the application source code that implements the transactions and forms modules.
- Appendix B contains the code used to create and load the database.
- Appendix C contains the configuration information for the operating system, the RDBMS and Tuxedo.
- Appendix D contains the 180-day space calculations.
- Appendix E contains the code used to generate transactions and measure response times.
- Appendix F contains the screen layouts of all the forms.
- Appendix G contains the price quotes.

Additional Copies

To request additional copies of this report, write to the following address:

Shanley P.R.
777 N First Street, Suite 600
San Jose, CA 95112-6311
(408) 295-8894
FAX (408) 295-2613

Enterprise 4500 TPC Benchmark ™C Full Disclosure

Introduction

The *TPC Benchmark ™C Standard Specification* requires test sponsors to publish, and make available to the public, a full disclosure report for the results to be considered compliant with the Standard. The required contents of the full disclosure report are specified in Clause 8.

This report is intended to satisfy the Standard's requirement for full disclosure. It documents the compliance of the benchmark tests reported in the *TPC Benchmark ™C* results for the Sun Microsystems Enterprise 4500 Server running the Sybase Adaptive Server Enterprise 11.9.3.

In the *Standard Specification*, the main headings in Clause 8 are keyed to the other clauses. The headings in this report use the same sequence, so that they correspond to the titles or subjects referred to in Clause 8.

Each section in this report begins with the text of the corresponding item from Clause 8 of the *Standard Specification*, printed in italic type. The plain type text that follows explains how the tests comply with the TPC C™ Benchmark requirement. In sections where Clause 8 requires extensive listings, the section refers to the appropriate appendix at the end of this report.

1- General Items

1.1 Application Code and Definition Statements

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

Appendix A contains the application source code that implements the transactions and forms modules.

1.2 Sponsor

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

This benchmark test was sponsored by Sun Microsystems, Inc. and Sybase, Inc.

1.3 Parameter Settings

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

- *Database tuning options*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and application configuration parameters*
- *Compilation and linkage options and run-time optimizations used to create/install applications, OS, and/or databases.*

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

Appendix C contains all the required parameter settings.

1.4 Configuration Diagrams

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

The measured configuration was the same as the priced configuration, with the exception that six 4.2 Gb system disks were use in place of 9.1Gb. Figure 1 is a diagram of the configuration.

Configuration Items for the Enterprise 4500

For the configuration, the server machine was a Sun Enterprise 4500 which consisted of the following:

- 7 CPU/Memory Boards
- 14 UltraSPARC-II 400 MHz Processors with 8MB External Cache each
- 28 Gb of main memory
- 1 Sbus I/O board
- 18 Sun StorEdge A5200 Array (22 x 9 GB FCAL disks in each)
- 4 Sun StorEdge A1000 Array (12 x 9 GB SCSI disks in each)
- 2 FCAL 100MB/s Sbus Host Adaptor
- 6 FCAL GBIC Modules
- 54.6GB Disk Multipack (6 x 9GB UltraSCSI disks)*
- Internal CD-ROM
- 12-24GB 4mm DDS-3 Backup Tape Device
- 2 additional PS/300W power supply modules

* In the benchmark configuration, a 25.2 Gb Disk Multipack (6 x 4.2 GB SCSCI disks in each) was used.

The fifteen client machines were Ultra 10 Model 333's and each contained:

- One UltraSPARC-II 333 MHz Processor.
- 1024 MB of Main Memory.
- One Internal SCSI-2 controller.

- One Internal 9 GB SCSI disk.
- Internal CD-ROM.
- Quad FastEthernet Controller

The benchmark configuration used a Remote Terminal Emulator (RTE) to emulate TPC-C user sessions. The driver systems were directly connected through ethernet to the clients which emulated the database client sessions.

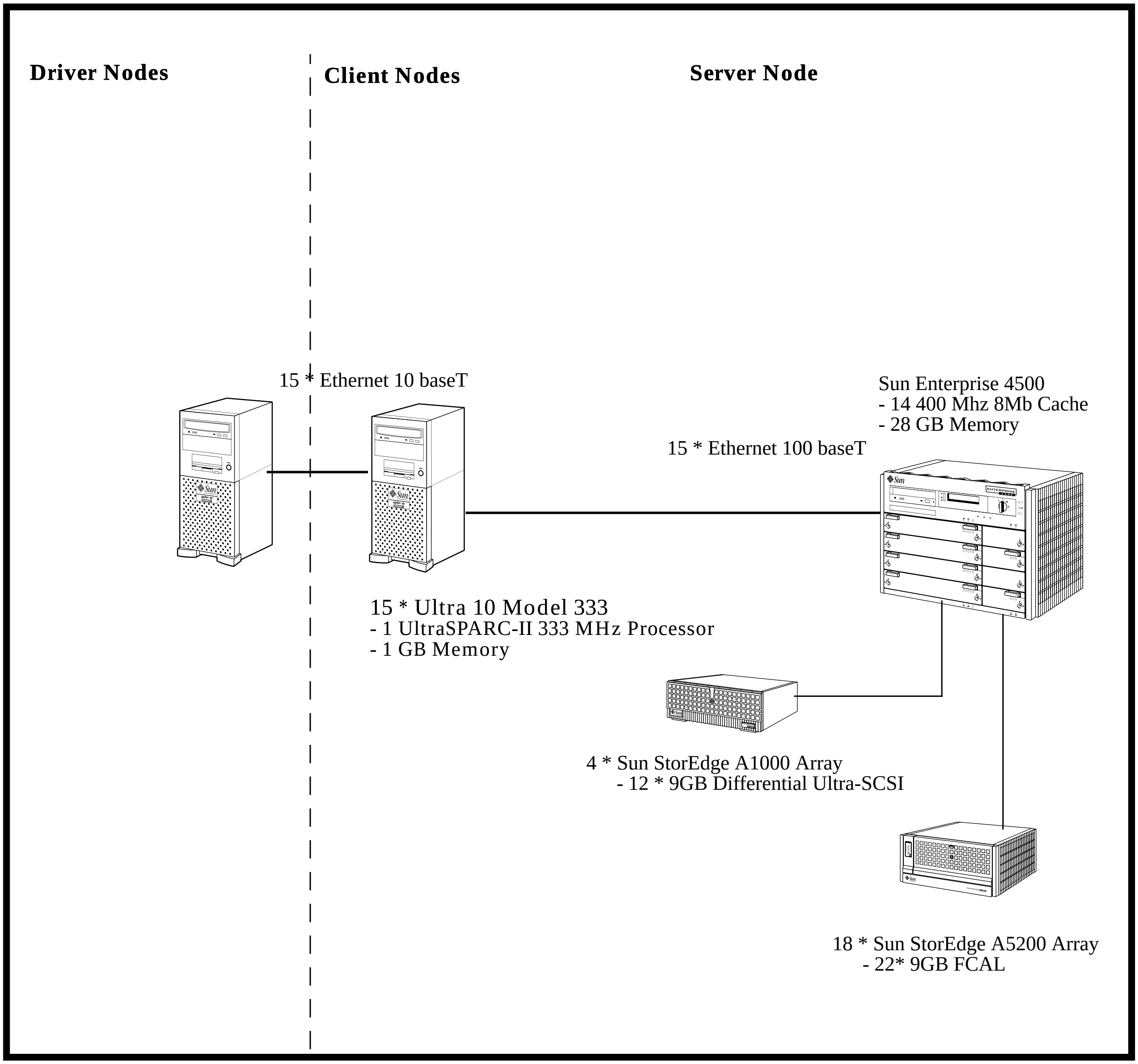


Figure 1: The Sun Enterprise 4500 Benchmark Configuration

2 - Clause 1 Related Items

2.1 Table Definitions

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

Appendix B describes the programs that define, create, and populate a Sybase Adaptive Server Enterprise 11.9.3 database for TPC-C testing.

2.2 Physical Organization of Database

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

Space was allocated to Sybase Adaptive Server Enterprise 11.9.3 on the server according to the data in section 5.2. The size of the database devices on each disk drive was calculated to provide even distribution of load across the disk drives.

2.3 Insert and Delete Operations

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

There were no restrictions on insert and delete operations to any tables beyond the limits defined in Clause 1.4.11.

2.4 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.

Partitioning was not used in this implementation.

2.5 Table Replication

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

No tables were replicated in this implementation.

2.6 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).

No additional or duplicate attributes were added to any of the tables.

3 - Clause 2 Related Items

3.1 Random Number Generation

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

The Random Number Generator used was the one that appeared in the article titled “Random Number Generators: Good Ones Are Hard To Find” in the communications of the ACM - October 1988, Volume 31, Number 10. The properties of this random number generator are well-known and are documented in the article as producing a uniformly distributed pseudo-random sequence. To generate a random number, the driver programs first use a seed based on the host address, current time and the process-id of the respective session. This guarantees that each emulated user on all the RTE machines is mathematically independent of others.

3.2 Input/Output Screen Layouts

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

The screen layouts are shown in Appendix F.

3.3 Terminal Feature Verification

The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained.

The terminal features were verified by the Audit manually exercising each specification on a representative Enterprise 1 running Solaris 2.5.1.

3.4 Presentation Manager or Intelligent Terminal

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

The TPC-C forms module was implemented using the capabilities of an xterm terminal emulator.

3.5 Transaction Statistics

Table 1 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 requires.

Table 1: Transaction Statistics

Transaction Type	Statistics	Percentage
New Order	Home warehouse	99.00
	Remote warehouse	1.00
	Rolled back transactions	1.00
	Average items per order	10.00
Payment	Home warehouse	85.04
	Remote warehouse	14.96
	Non-primary key access	60.00
Order Status	Non-primary key access	60.06
Delivery	Skipped transactions	0.00
Transaction Mix	New order	44.85
	Payment	43.06
	Order status	4.02
	Delivery	4.04
	Stock level	4.03

3.6 Queueing Mechanism

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

Delivery transactions were submitted to servers using the same Tuxedo call mechanism that other transactions used. The only difference was that the call was asynchronous - i.e., control returned to the client process immediately and the deferred delivery completed asynchronously.

4 - Clause 3 Related Items

4.1 Transaction System Properties (ACID)

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

The TPC Benchmark C Standard Specification defines a set of transaction processing system properties that a system under test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation, and Durability (ACID). This section defines each of these properties, describes the steps taken to ensure that they were present during the test and describes a series of tests done to demonstrate compliance with the standard.

4.2 Atomicity

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

4.2.1 Completed Transaction

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

The test was performed by first retrieving, through the *isql* utility, the balances from a set of randomly picked warehouse, district, and customer rows (selected by customer number). Then a Payment transaction was submitted through the TPC Benchmark C application. Upon completion of the transaction, the balances of the selected warehouse, district, and customer rows were once again retrieved to verify that the changes had been made.

4.2.2 Aborted Transaction

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

For this test, the same warehouse, district, and customer ids used above were used to issue a transaction to a modified version of the TPC Benchmark C application in which the COMMIT command had been replaced by a ROLLBACK command. After the transaction was aborted, the balances of the warehouse, district, and customer rows were retrieved to verify that no changes had been made to the database.

4.3 Consistency

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

The TPC Benchmark C standard requires System Under Test to meet the 12 consistency conditions listed in Clause 3.3.2.

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

In order to demonstrate the consistency of the application, the following steps were taken:

1. Prior to the start of the benchmark run, the consistency of the database was verified by applying the consistency conditions 1-4 described above.

2. After each measurement runs, all consistency tests 1-4 were performed.

Upon the completion of the benchmark, the consistency of the database was determined by applying the same consistency conditions used in step 1.

4.4 Isolation

Isolation can be defined in terms of phenomena that can occur during the execution of concurrent transactions. These phenomena are P0 (“Dirty Write”), P1 (“Dirty Read”), P2 (“Non-repeatable Read”) and P3 (“Phantom”). The table in Clause 3.4.1 of *the TPC-C specifications defines the isolation requirements which must be met by the TPC-C transactions. Sufficient conditions must be enabled at either the system or application level to ensure the required isolation is maintained.*

Sybase Adaptive Server Enterprise 11.9.3 ensures isolation and full serializability by locking strategies that preserve data integrity when multiple users are accessing a database.

The TPC Benchmark C Standard Specification defines a set of required tests to be performed on the system under test to demonstrate that transaction isolation was present in the system configuration. These tests involve the execution of two transactions on the system and examining the interaction when the results of the transactions are committed to the database and when the results are aborted.

4.4.1 Isolation Test 1

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

1. At terminal 1, a New-Order transaction was started but not COMMITted.

2. At terminal 2, an Order-Status transaction was started for the same customer used on terminal 1. This transaction attempted to read the data for the order on terminal 1.

3. Terminal 2 transaction waited for the terminal 1 transaction to complete.

4. COMMITted the transaction on terminal 1. The transaction on terminal 2 now completed.

5. The results from the Order-Status transaction matched the data entered for the New-Order.

4.4.2 Isolation Test 2

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

1. Completed an Order-Status transaction.

2. At terminal 1, a New-Order transaction was started for the same customer used in step 1, but not COMMITted.

3. At terminal 2, an Order-Status transaction was started for the same customer used at terminal 1. This transaction attempted to read the data for the order at terminal 1.

4. The terminal 2 transaction waited for the terminal 1 transaction.

5. A ROLLBACK was executed for the transaction at terminal 1. The transaction at terminal 2 now completed.

6. The results from the Order-Status transaction matched the data returned in step 1.

4.4.3 Isolation Test 3

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

1. At terminal 1, a New-Order transaction was started but not committed.

2. At terminal 2, another New-Order transaction was started for the same customer used at terminal 1.

3. The terminal 2 transaction waited for the terminal 1 transaction to complete.

4. COMMITted the transaction at terminal 1. The transaction at terminal 2 completed.

5. The order number for the terminal 2 transaction was one greater than the terminal 1 transaction. The Next Order Number for the district reflected the results from both transactions.

4.4.4 Isolation Test 4

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

1. At terminal 1, a New-Order transaction was started but not COMMlTted.

2. At terminal 2, another New-Order transaction was started for the same customer used at terminal 1.

3. The terminal 2 transaction waited for the terminal 1 transaction to complete.

4. A ROLLBACK was executed for the transaction on terminal 1. The transaction on terminal 2 completed.

5. The order number for the terminal 2 transaction was one greater than the previous transaction. The Next Order Number for the district reflected the results from only the terminal 2 transaction. In other words, it had been incremented by one.

4.4.5 Isolation Test 5

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

1. At terminal 1, a Delivery transaction was started but not COMMITted.

2. At terminal 2, a Payment transaction was started for the same customer used at terminal 1.

3. Terminal 2 transaction waited for the terminal 1 transaction to complete.

4. COMMITted the transaction at terminal 1. The transaction at terminal 2 completed.

5. The customer balance reflected the results from both transactions.

4.4.6 Isolation Test 6

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

1. At terminal 1, a Delivery transaction was started but not COMMITed.

2. At terminal 2, a Payment transaction was started for the same customer used on terminal 1.

3. Terminal 2 transaction waited for terminal 1 transaction to complete.

4. A ROLLBACK was executed for the transaction on terminal 1. The transaction on terminal 2 completed.

5. The customer balance reflected the result of the Payment transaction only.

4.4.7 Isolation Test 7

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

1. At terminal 1, New-Order transaction Tl was started and was queried for the price of two items, item x and item y. The transaction completed.

2. At terminal 2, New-Order transaction T2 was started for a group of items including item x twice and item y. The transaction was stopped immediately after querying the price of item x for the first time and before querying the price of x for the second time or item y.

3. At terminal 1, an interactive SQL transaction,T3, was started to increase the price of items x and y by ten percent. The transaction failed to complete.

The transaction on terminal 2 locks in read mode the row with the price of item x. This would force the transaction on terminal 1 (step 3) to stall. Therefore, Case A of Clause 3.4.2.7 occurred.

4. Continued terminal 2 transaction, The price of items x (the second time) and y matched the values read by step 1. Completed the transaction.

5. Transaction T3 completed and COMMITted.

6. At terminal 1, another transaction was started to query the prices of items x and y. Completed the transaction.

7. The prices read by the transaction in step 6 matched those set by transaction T3.

4.5 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.

List of single failures:

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

Instantaneous interruption (system crash/system hang) in processing which requires system reboot to recover.

Failure of all or part of memory (loss of contents).

Sun Microsystems executed three durability tests to satisfy the durability requirements for this implementation of TPC Benchmark C. The combined test for loss of memory and instantaneous interruption was performed with a fully scaled database under the full load of terminals. The test for loss of data and the test for loss of log was performed with a 15 warehouse database.

4.5.1 Permanent Irrecoverable Failure

Two tests were performed in this case: one for loss of data disk, the other for loss of a recovery log disk. Consistency condition 3 as specified in Clause 3.3.2.3 was verified after these tests.

4.5.2 Loss of Data Disk

The following steps were taken to demonstrate durability in case of loss of a data disk:

1. The database was loaded with 15 warehouses.

2. The database was backed up (database dump) to disks.

3. The D_NEXT_O_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).

4. A test was executed with 150 terminals. On the driver system, the committed and rolled back New-Order transactions were recorded in a "success" file.

5. After a few minutes into the measurement period, one of the data disks was powered down.

6. The test was aborted on the driver.

7. The transaction log is dumped to a file.

8. The original database was restored from the backup copy in step 2.

9. Sybase was restarted and its transaction log, that was dumped in step 7, was used to roll forward the transactions that had completed since the backup.

10. The contents of the “success” file on the driver and the ORDERS table were compared to verify that records in the “success” file for committed New-Order transaction had corresponding records in the ORDERS table.

11. Step 3 was repeated to determine the total number of orders (count2). Count2-count1 was the same as the number of committed records in the “success” file.

4.5.3 Loss of Log Disk

The following steps were taken to demonstrate durability in case of loss of a recovery log disk:

1. The D_NEXT_ O_ID fields for all rows in district table were added to determine the initial count of the total number of orders (count1).

2. A test was executed with 150 terminals. On the driver system, the committed and rolled back New-Order transactions were recorded in a “success” file.

3. After few minutes into the measurement period, one of the log disks was powered down.

4. Since the log disk is mirrored, the system continues to process transactions despite the missing disk.

5. The test was allowed to run till completion.

6. The contents of the “success” file on the driver and the ORDERS table were compared to verify that records in the “success” file for committed New-Order transaction had corresponding records in the ORDERS table.

7. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 was the same as the number of committed records in the “success” file.

8. The missing log disk was replaced and was successfully re-synchronized with the disk present during the run.

4.5.4 Instantaneous Interruption and Loss of Memory

Instantaneous interruption and loss of memory tests were combined because the loss of power erases the contents of memory. This failure was induced by removing the SUT's primary power while the benchmark was running.

1. The D_NEXT_O_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).

2. A fully-scaled test was executed. On the driver system, the committed and rolled back New-Order transactions were recorded in a “success” file.

3. After 5 minutes into the measurement period, the SUT's primary power was removed.

4. The test was aborted on the driver.

5. Power was restored to the SUT and a normal system recovery was done. A recovery was automatically performed by Sybase Adaptive Server Enterprise 11.9.3 when the database was restarted and brought on-line. The recovery restored the database to the consistent point just after the last committed transaction had occurred before the induced failure.

6. The contents of the “success” file on the driver and the ORDERS table were compared to verify that records in the “success” file for committed New-Order transactions had corresponding records in the ORDERS table. The number of transactions missed “in flight” were less than the number of users.

7. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 was compared with the number of committed records in the “success” file.

5 - Clause 4 Related Items

5.1 Initial Cardinality of Tables

The Cardinality (e.g. number of rows) of each table, as it existed at the start of the benchmark run (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.

The TPC-C database for this test was configured with 4080 warehouses.

Table 2: Cardinality of Tables

Table	Occurrences
Warehouse	4080
District	40800
Customer	100000
History	36720000
Orders	122400000
New order	122400000
Order line	122400000
Stock	1224000000
Item	40800000

5.2 Database Layout

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

The distribution of database tables over the 396 disks of the system is an extension of the distribution of the tested system, 180 day storage growth requirements are made with the unused space plus additional disks. Figure 2 shows the configuration of the system disks.

5.2.1 Database Layout of Benchmark System.

Table 3: Disk Layout

Device Name	No. of Devices	Physical Disks
Log Devices / Mirrors (8 hr)	3	24 * A1000 disks
customer	23	Each device striped across 42 disks
history	2	Each device striped across 42 disks
stock	41	Each device striped across 42 disks
order-line	17	Each device striped across 42 disks
order	5	Each device striped across 42 disks
master	1	Striped across 45 disks

The data was striped evenly across a total of 396 disks. All of these were located in 18 Sun SPARCstorage Arrays (9 GB disks). An additional 6 x 4.2GB disks were used for the Operating System, swap disks and Sybase binaries.

The logs and mirrors were located on 24 physical A1000 disks (9GB).

5.3 Type of Database

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.

Sybase Adaptive Server Enterprise 11.9.3 is a relational database management system.The interface we used was Sybase DB-Library and stored procedures.

5.4 Mapping of Database

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

No table partitioning or replication was done.

5.5 180 Day Space Computation

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).

The 180 day space computation is shown in Appendix D.

6 - Clause 5 Related Items

6.1 Measured tpmC

Measured tpmC must be reported.

The measured tpmC was *50268.07*

6.2 Response Times

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

Table 4: Response Times

Type	Average	Maximum	90% percentile
New-Order	0.34	24.14	.80
Payment	0.47	25.52	1.00
Order-Status	0.51	21.63	1.00
Interactive Delivery	0.25	0.35	0.26
Deferred Delivery	0.64	5.00	2.00
Stock-Level	.67	5.44	2.00
Menu	0.25	0.65	0.50

6.3 Keying and Think Times

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

Table 5: Keying Times

Type	Average	Minimum	Maximum
New-Order	18.02	18.01	18.17
Payment	3.02	3.01	3.13
Order-Status	2.02	2.01	2.14
Interactive Delivery	2.02	2.01	2.10
Stock-Level	2.02	2.01	2.10

Table 6: Think Times

Type	Average	Minimum	Maximum
New-Order	12.20	0.00	122.0
Payment	12.20	0.00	122.0
Order-Status	10.25	0.00	102.5
Interactive Delivery	5.19	0.00	52.00
Stock-Level	5.20	0.00	52.00

6.4 Response Time Frequency Distribution Curves

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

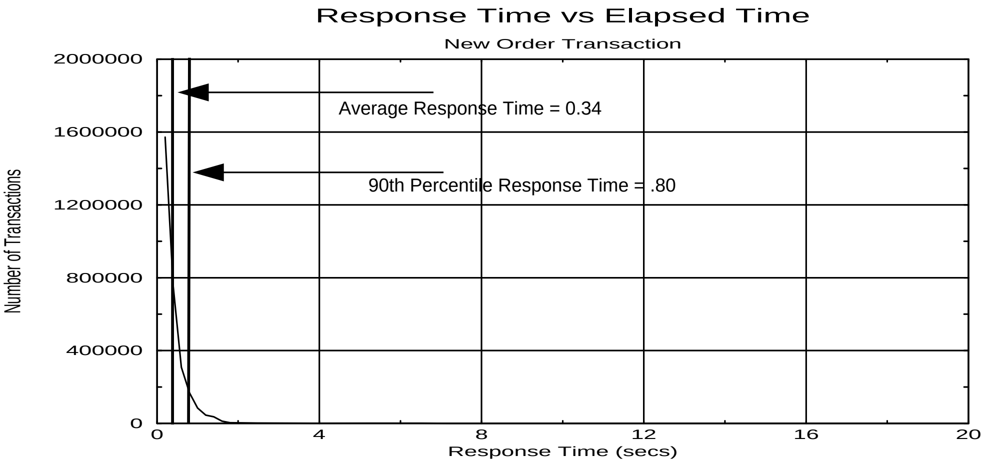


Figure 5: New Order Response Time Distribution

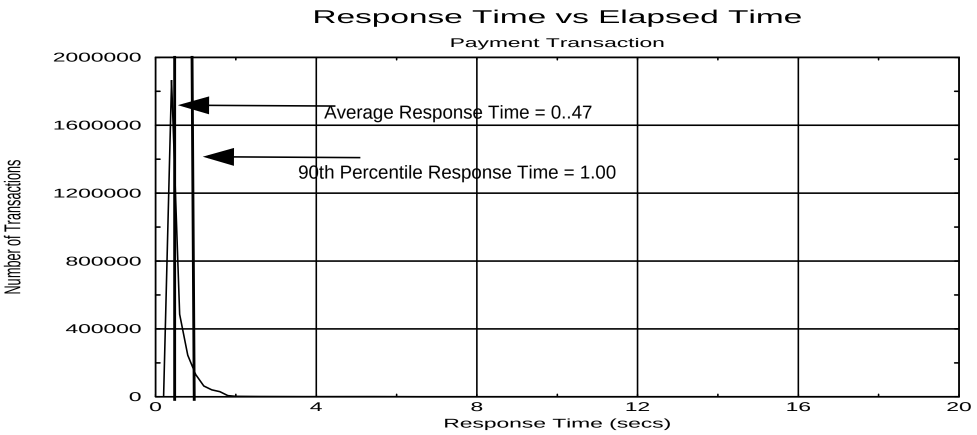


Figure 6: Payment Response Time Distribution

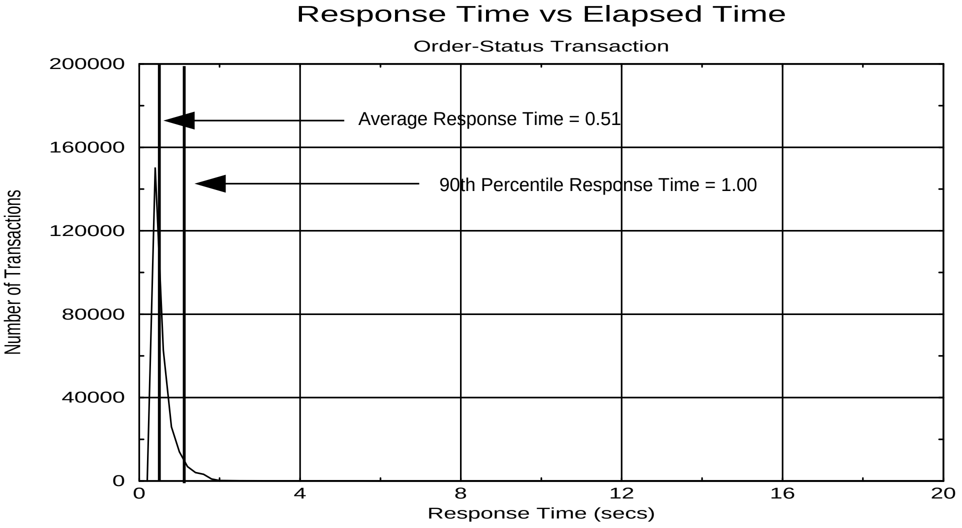


Figure 7: Order Status Response Time Distribution

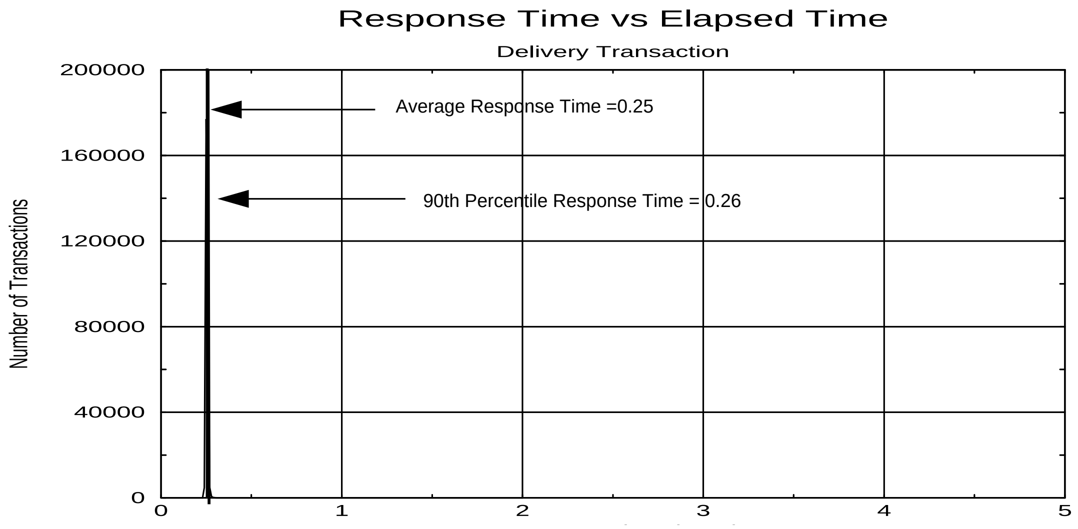


Figure 8: Delivery Response Time Distribution

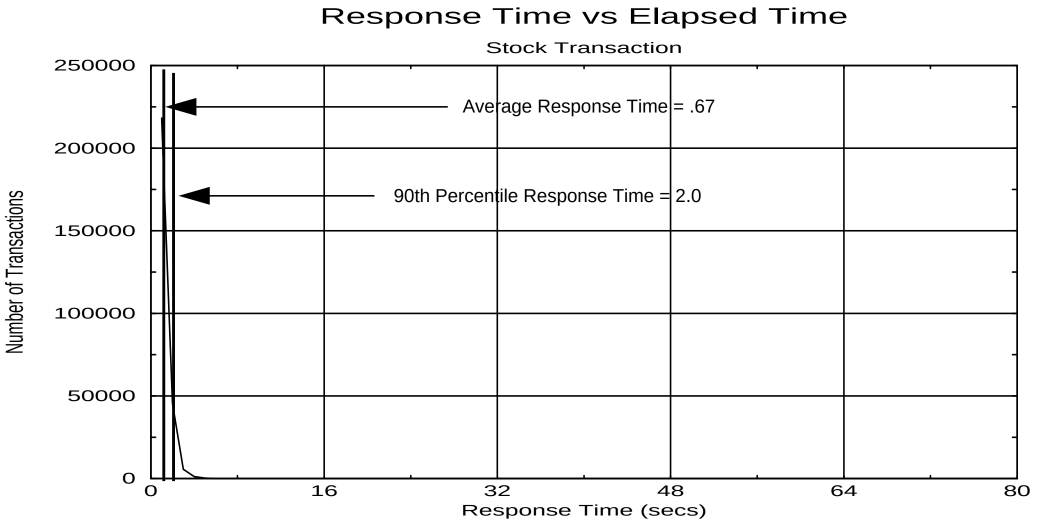


Figure 9: Stock Level Response Time Distribution

6.5 Response time versus throughput

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

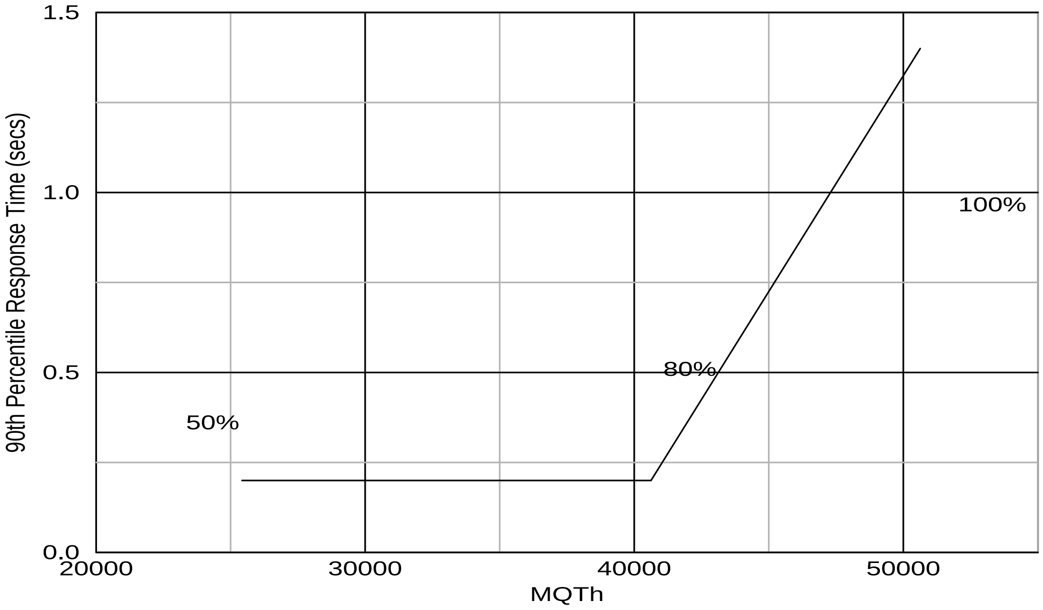


Figure 10: Response Time versus Throughput

6.6 Think Time distribution curves

Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type.

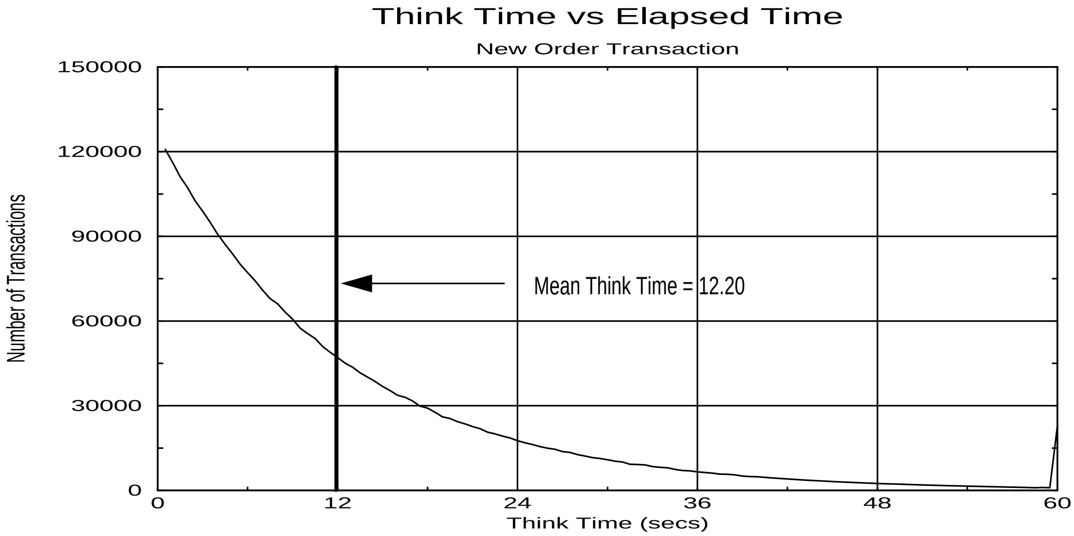


Figure 11: New Order Think Time Distribution

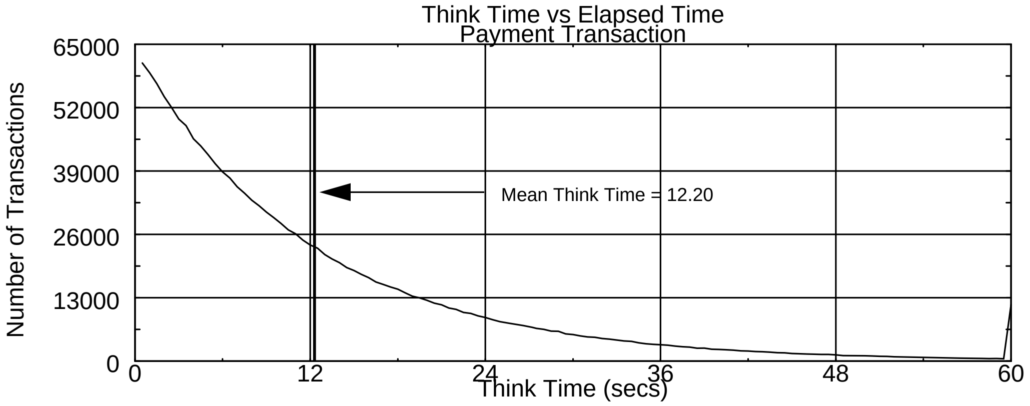


Figure 12: Payment Think Time Distribution

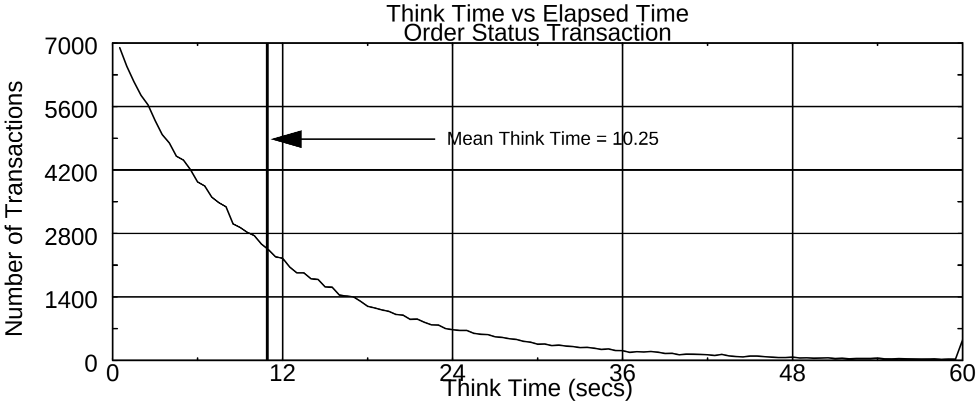


Figure 13: Order Status Think Time Distribution

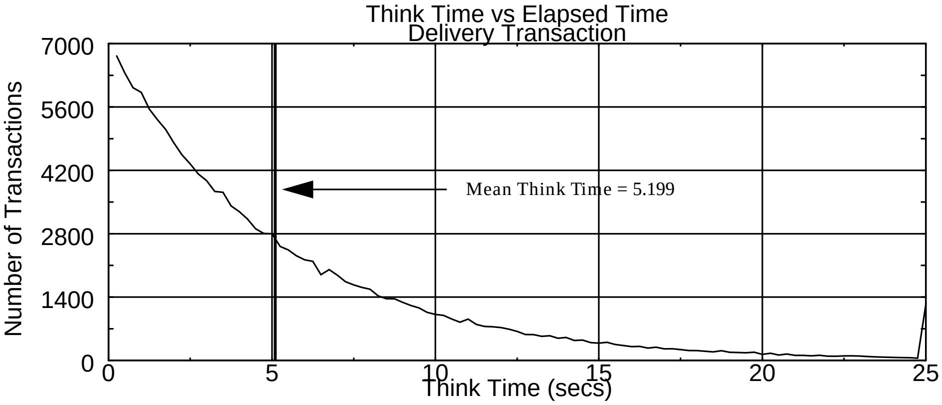


Figure 14: Delivery Think Time Distribution

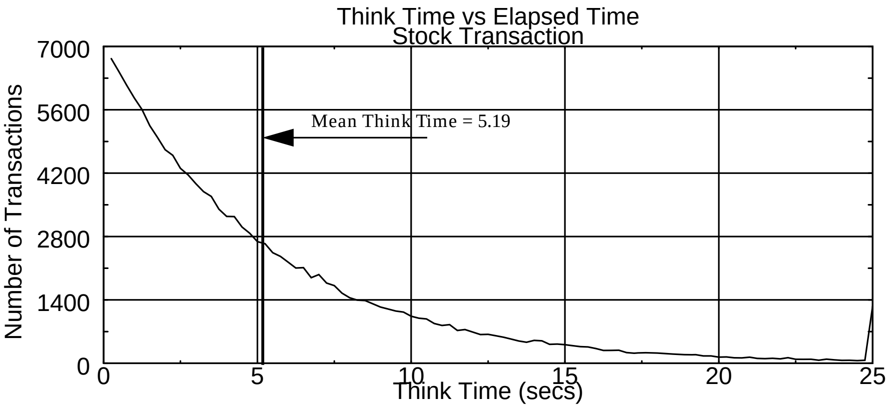


Figure 15: Stock Level Think Time Distribution

6.8 Throughput versus Elapsed Time

A graph of throughput versus elapsed time (see Clause 6.6.5) must be reported for the New-Order transaction.

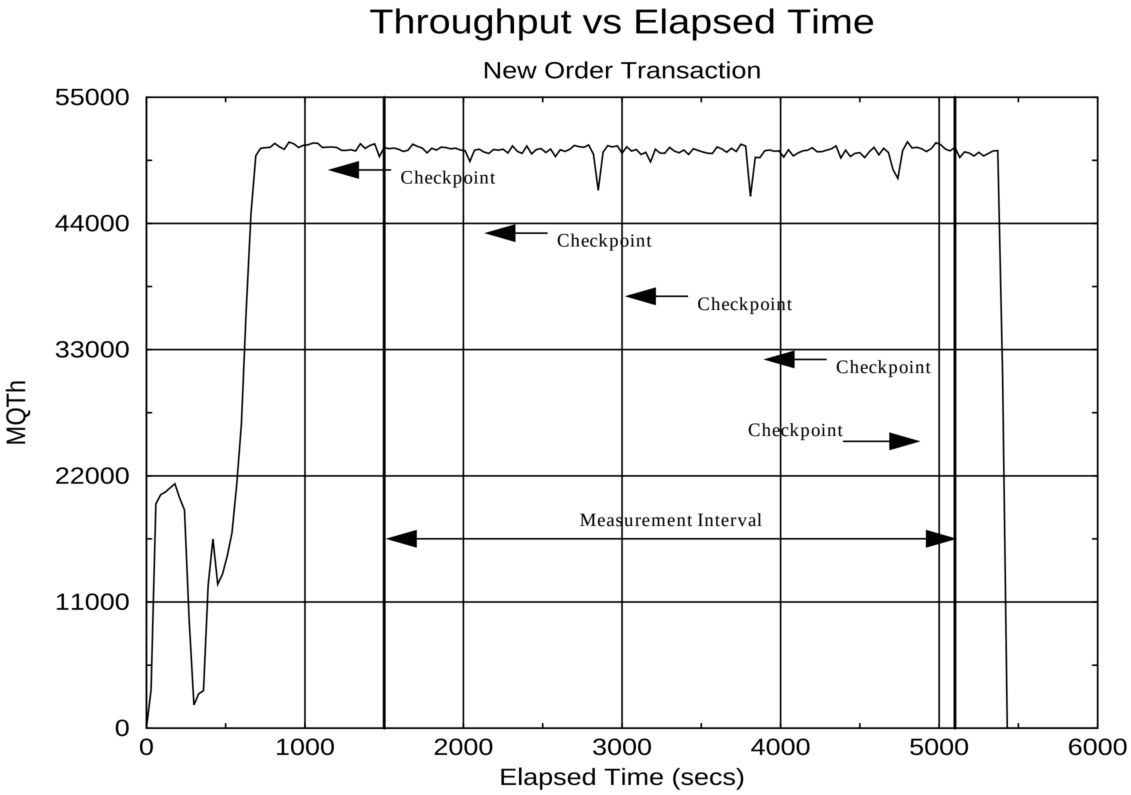


Figure 16: Throughput vs Elapsed Time

6.9 Steady State Determination

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.

The transaction throughput rate (tpmC) and response times were relatively constant after the initial ‘ramp up’ period. The throughput and response time were verified by examining the throughput (tpmC) graph reported at 30 second intervals for the duration of the benchmark. Ramp up, steady state, and ramp down are clearly discernible in the graph, Figure 16.

6.10 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.

6.10.1 Checkpoint

A Sybase Adaptive Server Enterprise 11.9.3 checkpoint writes all buffers in memory to disk so that data on disk matches what is in memory. Checkpoints are marked by a special record written into the logs. One checkpoint was implemented in the measurement run.

6.11 Reproducibility

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

In a repeat run, a throughput of 50,216.12 tpmC was achieved.

6.12 Measurement Period Duration

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

The measurement interval was 60 minutes.

6.13 Transaction Mix Regulation

The method of regulation of the transaction mix (e.g., card decks or weighted random distribution) must be described. If weighted distribution is used and the RTE adjusts the weights associated with each transaction type, the maximum adjustments to the weight from the initial value must be disclosed.

The weighted distribution algorithm as described in Clause 5.2.4.1 of the TPC-C specification was used to regulate the transaction mix. Weights for the various transactions were statically assigned.

6.14 Numerical Results

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

See Table 1 for results.

6.15 New-Orders Rolled-Back

The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed.

See Table 1 for results.

6.16 Order-Line Average

The average number of order-lines entered per New-Order transaction must be disclosed.

See Table 1 for results.

6.17 Remote Order-Lines

The percentage of remote order-lines entered per New-Order transaction must be disclosed.

See Table 1 for results.

6.18 Remote Payments

The percentage of remote payment transactions must be disclosed.

See Table 1 for results.

6.19 Customer Lastname

The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed.

See Table 1 for results.

6.20 Deliverys Skipped

The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.

See Table 1 for results.

6.21 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.

One checkpoint occurred at 1200 seconds after the start of ramp-up and 4 after the start of the measurement interval. The interval between the checkpoints was 15 minutes.

7 - Clause 6 Related Items

7.1 RTE Description

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

The RTE used was developed by Sun Microsystems and is proprietary. It consists of a *master_rte* program which forks off the individual RTE processes and controls the run. After the run completes, a separate report generator program collects all the log files and generates the final statistics of a run.

Inputs to the RTE include the names of the RTE machines to run on, client machines to attach to, the database scale, the ramp-up, measurement and ramp-down times. The script used to set these values is shown below:

```
setenv  ramp_up         1500    # ramp_up interval (secs)
setenv  stdy_state      3600    # steady-state/measurement interval
(secs)
setenv  ramp_down       300      # ramp_down interval (secs)
setenv  trigger_time    1600    # Trigger time for users to login
setenv  scale           4080    # of warehouses
setenv  comment         ""

# Number of users on each machine

set users = ( 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750 2750
2750 2750 2750 2750 )

# Number of users on each machine

set rte_machines = ( r01 r02 r03 r04 r05 r06 r07 r08 r09 r10 r11 r12
r13 r14 r15 )      # Names of rte machines

set clnt_machines = ( c01 c02 c03 c04 c05 c06 c07 c08 c09 c10 c11
c12 c13 c14 c15 )     # Names of client machines (same # as #rtes)

set mix = ( 404 807 1209 5514 10000 )    # %Mix of transactions
(stock,del,ords,paym,newo)

set think = ( 5200 5200 10250 12200 12200 ) # Think times in ms for
above tx
```

The code used to generate the transactions and record response times is shown in Appendix E.

7.2 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.

** In the configuration, workstations are connected to the clients via telnet in the same way as the emulated system. The driver system emulates the workstations by making a direct connection to the SUT for each terminal.

7.3 Configuration Diagrams

A complete funcitional 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).

Figure 1 is a diagram of the benchmarked configuration and Figure 2 shows the configuration of the priced configuration. Section 1.4 of this Full Disclosure Report gives details on both configurations.

7.4 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).

The configuration used one 10BaseT LAN for each driver system, connecting the driver system to the corresponding client and one 100BaseT LAN connecting all the 15 client systems to the server. There were about *** workstations “terminals” on each.

7.6 Operator Intervention

If the configuration requires operator intervention, the mechanism and the frequency of this intervention must be disclosed.

The Enterprise 4500 Server configuration reported does not require any operator intervention to sustain the reported throughput.

8 - Clause 7 Related Items

8.1 System Pricing

A detailed list of hardware and software used in the priced system must be reported. Each separately orderable item must have vendor part number, description, 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.

The Executive Summary on page vi lists pricing information for all components. All Sun pricing is from CAT Technology, Inc. The hub pricing was obtained from Software House International, Inc. The Tuxedo pricing is from BEA. Please refer to appendix G for the quotes.

8.2 Support Pricing

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.

8.2.1 Sun Hardware and Software Support

The Silver Program of the SunService Support Program was used in all Sun pricing calculations. This program provides complete service with both on-site and telephone assistance. Features of this program include telephone assistance from 8:00 am to 5:00 pm, Monday - Friday; and on-site service assistance from 8:00 am to 5:00 pm, Monday - Friday; and Solaris maintenance releases. This service provides live telephone transfer of software fixes and 4 hour on-site response for urgent problems.

All Sun hardware has a one year warranty. During the warranty period, the monthly price for the Silver Program is 60% of the usual monthly price.

8.2.2 Sybase Standard Technical Support

Sybase Standard Technical Support includes:

- Product updates.
- Unlimited, toll-free telephone service to assist in product installation, syntax, and usage that is available from 6:00 a.m. to 5:00 p.m., Monday through Friday.

8.3 Discounts

The following generally available discounts to any buyer with like conditions were applied to the priced configurations:

- a 10% Sun support 3 year contract discount
- a 5% Sun support pre-payment discount
- a 10% Sybase volume discount on Sybase Adaptive Server products.

8.4 Availability

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.

All Sun products will be available by March 30, 2000. Sybase Adaptive Server Enterprise 11.9.3 is available now.

8.5 TpmC, Price/TpmC

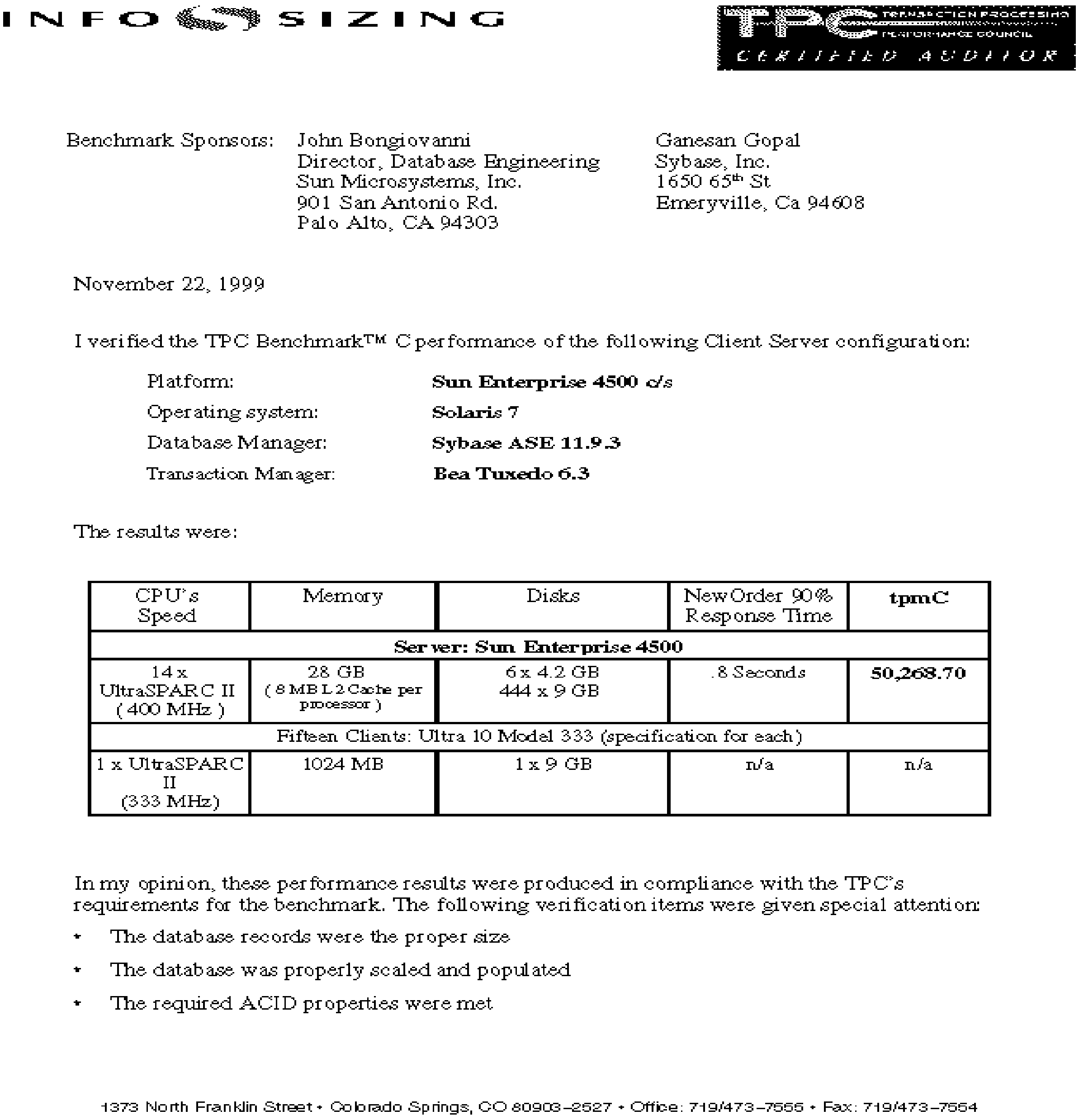
A statement of the measured tpmC, as well as the respective calculations for 5-year pricing, price/performance (price/tpmC), and the availability date must be included.

The Maximum Qualified Throughput for the Ultra Enterprise 4500 was 50268.07 tpmC at $49.88 per tpmC.

9 - Clause 8 Related Items

9.1 Auditor's Report

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



INFOSIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL
CERTIFIED AUDITOR

Benchmark Sponsors:

John Bongiovanni
Director, Database Engineering
Sun Microsystems, Inc.
901 San Antonio Rd.
Palo Alto, CA 94303

Ganesan Gopal
Sybase, Inc.
1650 65th St
Emeryville, Ca 94608

November 22, 1999

I verified the TPC Benchmark™ C performance of the following Client Server configuration:

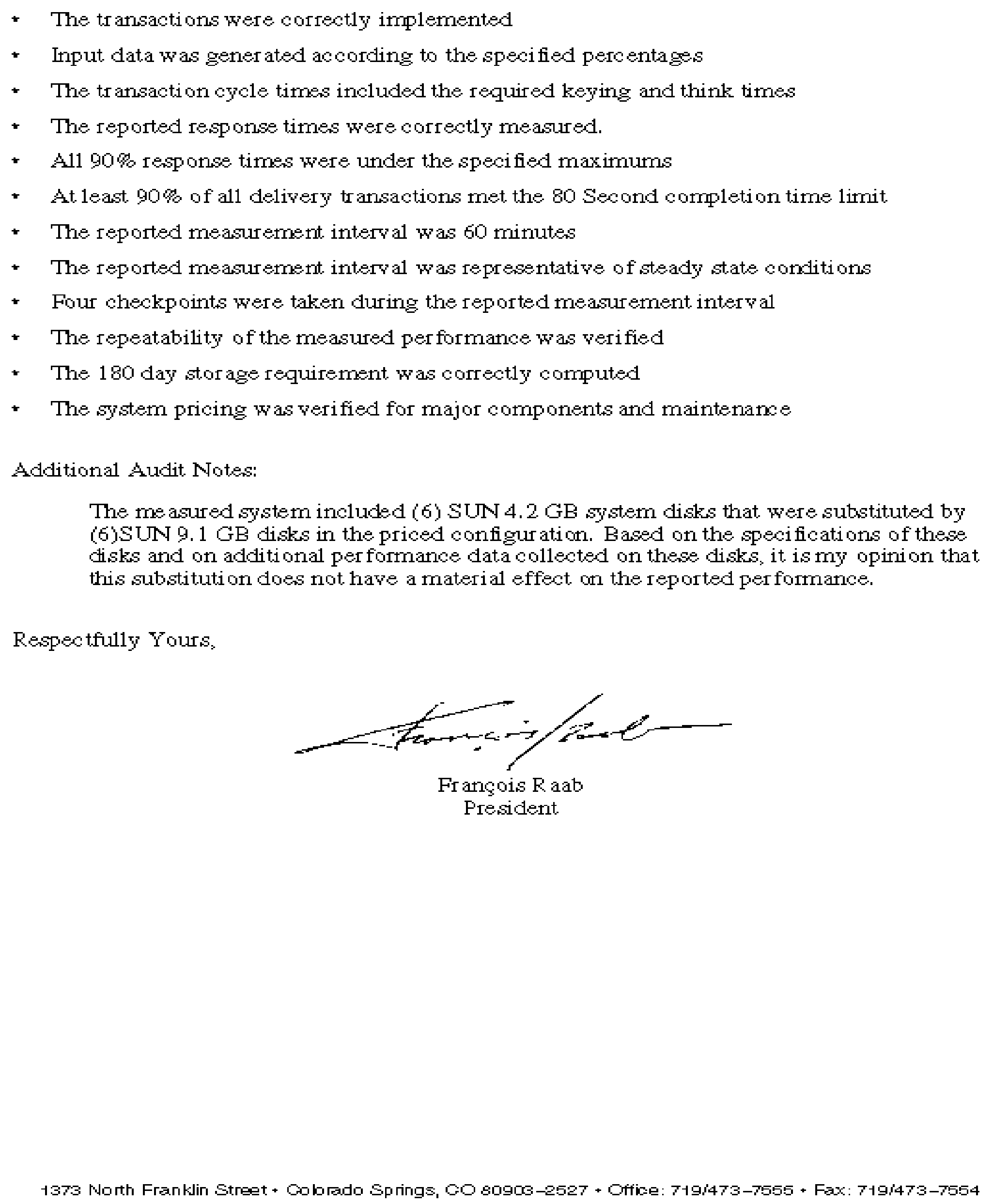
Platform:	**Sun Enterprise 4500 c/s**
Operating system:	**Solaris 7**
Database Manager:	**Sybase ASE 11.9.3**
Transaction Manager:	**Bea Tuxedo 6.3**

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	**tpmC**
Server: Sun Enterprise 4500				
14 x UltraSPARC II (400 MHz)	28 GB (8 MB L2 Cache per processor)	6 x 4.2 GB 444 x 9 GB	.8 Seconds	**50,268.70**
Fifteen Clients: Ultra 10 Model 333 (specification for each)				
1 x UltraSPARC II (333 MHz)	1024 MB	1 x 9 GB	n/a	n/a

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

- The database records were the proper size
- The database was properly scaled and populated
- The required ACID properties were met



- The transactions were correctly implemented
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.
- All 90% response times were under the specified maximums
- At least 90% of all delivery transactions met the 80 Second completion time limit
- The reported measurement interval was 60 minutes
- The reported measurement interval was representative of steady state conditions
- Four checkpoints were taken during the reported measurement interval
- The repeatability of the measured performance was verified
- The 180 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

> The measured system included (6) SUN 4.2 GB system disks that were substituted by (6)SUN 9.1 GB disks in the priced configuration. Based on the specifications of these disks and on additional performance data collected on these disks, it is my opinion that this substitution does not have a material effect on the reported performance.

Respectfully Yours,

François Raab
President

Appendix A: Application Code

This Appendix contains the application source code that implements the transactions and Forms modules.

```
/*
 * Copyright (c)  1995, 1996, 1997, 1998, 1999  by Sun Microsystems,
Inc.
 */

#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/time.h>
#include<sys/procset.h>
#include<sys/param.h>
#include<limits.h>
#include<errno.h>


#include <stdlib.h>
#include <errno.h>

#include “tpcc_client.h”
#include “tpcc_tux.h”



main()
{
        int             menu_selection;
        void             do_transaction(int);

        initialize();
        Send_Menu();

        while ((menu_selection = sel_trans()) != 9) {
                if ((menu_selection < 1) || (menu_selection > 5))
                        continue;
                do_transaction(menu_selection - 1);
                Send_Menu();
        }
        rundown();
}
initialize()
{
        int             menu_selection, start, m, n;
        char list[] =
“0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ”
;
        tty_in = 0;
        tty_out = 1;

        if (Init_Monitor()) {
        fprintf(stderr, “\033[24;1H\033[mUnable to connect to TP Monitor\n\01”);
        exit(1);
        }

        get_wd();

        set_display();


}

rundown()
{
        restore_terminal();
        Rundown_Monitor();
}

get_wd(int num)
{
        num = 5 ;

        setup_wd();
        display_screen(num);
        get_inputs(num);
}
```

```
void
do_transaction(int num)
{
        int             status;
        char c;

        display_screen(num);
        status = get_inputs(num);
        if (status == 3)
                return;
        if ( Snd_Txn_To_Monitor(num) ){
         cleanup("\033[24;1H\033[mTransaction error occured");
        }
        else
                display_output(num);
}
 /*
  * Copyright (c)  1995, 1996, 1997, 1998, 1999  by Sun Microsystems, Inc.
  */

#include <time.h>

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

#define BOOLEAN int
#define LINEMAX 256

#define FALSE 0
#define TRUE 1
#define NEWORDER 0
#define PAYMENT 1
#define ORDSTAT 2
#define DELIVERY 3
#define STOCKLEV 4
#define WD 5
#define MAX_OL 15
#define TPM_ERROR 1

char            date_field[80];
char            tty_name[11];
int             w_id;
int             d_id;
int             xact_type;

struct no_itm_struct {
        int             ol_supply_w_id;
        int             ol_i_id;
        char            i_name[25];
        int             ol_quantity;
        int             s_quantity;
        char            brand[2];
        double          i_price;
        double          ol_amount;
};
struct no_struct {
        int             w_id;
        int             d_id;
        int             c_id;
        int             o_id;
        int             o_ol_cnt;
        double          c_discount;
        double          w_tax;
        double          d_tax;
        char            o_entry_d[20];
        char            c_credit[3];
        char            c_last[17];
        struct          no_itm_struct o_ol[15];
        char    status[25];
        double          total;
};
struct pay_struct {
        int             w_id;
        int             d_id;
        int             c_id;
        int             c_w_id;
        int             c_d_id;
        double          h_amount;
        double          c_credit_lim;
        double          c_balance;
        double          c_discount;
        char            h_date[20];
        char            w_street_1[21];
        char            w_street_2[21];
        char            w_city[21];
        char            w_state[3];
        char            w_zip[11];
        char            d_street_1[21];
        char            d_street_2[21];
        char            d_city[21];
        char            d_state[3];
        char            d_zip[11];
        char            c_first[17];
        char            c_middle[3];
        char    c_last[17];
        char            c_street_1[21];
        char            c_street_2[21];
        char            c_city[21];
        char            c_state[3];
        char            c_zip[11];
        char            c_phone[17];
        char            c_since[11];
        char            c_credit[3];
        char c_data_1[51];
        char c_data_2[51];
        char c_data_3[51];
        char c_data_4[51];


};

struct ord_itm_struct {
        int             ol_supply_w_id;
        int             ol_i_id;
        int             ol_quantity;
        double          ol_amount;
        char            ol_delivery_d[11];
};

struct ord_struct {
        int             ol_cnt;
        int             w_id;
        int             d_id;
        int             c_id;
        int             o_id;
        int             o_carrier_id;
        double          c_balance;
        char            c_first[17];
        char            c_middle[3];
        char            c_last[17];
        char            o_entry_d[20];
        struct ord_itm_struct s_ol[MAX_OL];
};

struct del_struct {
        int             w_id;
        int             o_carrier_id;
        time_t          queue_time;
};

struct stock_struct {
        int             w_id;
        int             d_id;
        int             threshold;
        int             low_stock;
};

struct menu_struct {
        int     w_id;
        int     d_id;
};
typedef union info {
                struct no_struct neworder;
                struct pay_struct payment;
                struct ord_struct ordstat;
                struct del_struct delivery;
                struct stock_struct stocklev;
                struct menu_struct wd;
        }       info_t;
struct io_tpcc {
        int             type;
        info_t          info;
};
/*
 * Copyright (c)  1995, 1996, 1997, 1998, 1999  by Sun Microsystems, Inc.
 */

#include <stdio.h>
#include <sys/termio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include “tpcc_client.h”
#include “tpcc_forms.h”
#include “tpcc_tux.h”

static intscreen_bufindex;
static char     screen_buf[SCRBUF_LEN];
extern void     Clog(char *,...);
extern void     SCREENlog(int, char *);
const charblanks[1802] = “                                  “;

void
setraw()
{
        extern struct tbufsave;
        struct termio   tbuf;
        int             status;

        if (ioctl(tty_in, TCGETA, &tbuf) == -1)
                return;
        tbufsave = tbuf;
        tbuf.c_iflag &= ~(INLCR | ICRNL | IUCLC | ISTRIP | IXON |
BRKINT);
        tbuf.c_oflag &= ~OPOST;
        tbuf.c_lflag &= ~(ICANON | ISIG | ECHO);
        tbuf.c_cc[VMIN] = LEAVE_SCREEN_MIN;
        tbuf.c_cc[VTIME] = LEAVE_SCREEN_TIMEOUT;

        if (ioctl(tty_out, TCSETAF, &tbuf) == -2)
                syserr(“ioctl_ERROR#2 - setting raw mode for STDIN error”);
}
void
restore_terminal()
{
        extern struct tbufsave;

        struct termio   tbuf;
        int             status;

        if (ioctl(tty_out, TCSETAF, &tbufsave) == -1)
                syserr(“ioctl_ERROR#3 - restoring original input terminal settings
error”);

        tbuf = tbufsave;
        if (ioctl(tty_out, TCSETAF, &tbuf) == -1)
                syserr(“ioctl_ERROR#4 - Forcing the original settings back for
STDIN error”);
}

int
sel_trans()
{
        int             c, read_count;
        static char     inbuf[2] = “\0\0”;
        int             i = 0;

        read_count = read(tty_in, inbuf, 1);
        if (read_count == 0)
                syserr(“TTY lost connection”);
        if (inbuf[0] == QUIT)
                return 9;

        switch (inbuf[0]) {
        case ‘n’:
                c = 1; break;
        case ‘p’:
                c = 2; break;
        case ‘o’:
                c = 3; break;
        case ‘d’:
                c = 4; break;
        case ‘s’:
                c = 5; break;
        case ‘e’:
                c = 9; break;
        }
        return c;
}

int         newo_val(int *);
int         paym_val(int *);
int         ords_val(int *);
int         del_val(int *);
int         stock_val(int *);
int         wd_val(int *);
int(*p_check_function[]) () = {
                &newo_val,
                &paym_val,
                &ords_val,
                &del_val,
                &stock_val,
                &wd_val
};

int
get_inputs(int txn_type)
{
        int         done = FALSE;
        int             i, returned_key;
        io_elem         *ioptr;
        int             last_input;
        float           float_h_amount = 0.0;

        memset(tuxibuf, ‘\0’, sizeof(info_t ));
        int_h_amount = 0;

        last_input = Forms[txn_type].num_input_elems - 1;
        i = 0;
        while (done == FALSE) {

                ioptr = &Forms[txn_type].input_elems[i];

                if (txn_type == PAYMENT){
                        if (i == 5)
                                payment_input = TRUE;
                        else
                                payment_input = FALSE;
                }

                returned_key = (ioptr->fptr) (ioptr->x, ioptr->y, ioptr-
>len,ioptr->flags, ioptr->dptr);


                switch (returned_key) {
                case BACKTAB:
                        if (i == 0)
                                i = last_input;
                        else
                                i--;
                        break;
                case TAB:
                        if (i == last_input)
                                i = 0;
                        else
                                i++;
                        break;
                case QUIT:
                        done = TRUE;
                        break;
                case SUBMIT:
                case LF:
                        if (screen_bufindex) {
                                PAINTSCRLEN(screen_buf, screen_bufindex);
                                screen_bufindex = 0;
```

```
                        }
                        payment_input = FALSE;
                        done = (p_check_function[txn_type]) (&i);
                        break;
                }
        }
        return returned_key;
}

int
newo_val(int *pos)
{
        int         done = FALSE;
        struct no_itm_struct *ol_ptr, *ol_ptr2;
        int     blank_line = 0, i, j;
        int     blank_array[MAX_OL];
        char    *bufp;

        iNO->w_id = w_id;

        for (i=0; i<MAX_OL; i++)
                blank_array[i] = 0;

        if (iNO->d_id <= 0) {
                *pos = 0;
                message = TRUE;
                PAINTSCR(MANDATORY_MSG);

        } else if (iNO->c_id <= 0) {
                *pos = 1;
                message = TRUE;
                PAINTSCR(MANDATORY_MSG);

        } else {

                ol_ptr = iNO->o_ol;

                for (i = 0; i < MAX_OL; i++, ol_ptr++) {

                        if (ol_ptr->ol_i_id || ol_ptr->ol_supply_w_id
                            || ol_ptr->ol_quantity)
                        {
                                /* and is that data complete */
                                if (ol_ptr->ol_i_id && ol_ptr->ol_supply_w_id
                                    && ol_ptr->ol_quantity)
                                {
                                        iNO->o_ol_cnt++;
                                        if (blank_line != 0) {
                                                ol_ptr2 = iNO->o_ol;
                                                for (j=0; j < i; j++) {
                                                        if (blank_array[j]) {
                                                                blank_array[j] = 0;
                                                                break;
                                                        }
                                                        ol_ptr2++;
                                                }
                                                ol_ptr2->ol_i_id =
                                                        ol_ptr->ol_i_id;
                                                ol_ptr2->ol_supply_w_id =
                                                        ol_ptr->ol_supply_w_id;
                                                ol_ptr2->ol_quantity =
                                                        ol_ptr->ol_quantity;
                                                ol_ptr->ol_i_id = 0;
                        ol_ptr->ol_supply_w_id = 0;
                        ol_ptr->ol_quantity = 0;
                        blank_array[i] = 1;
                        bufp = output_screen;
                        bufp += DISPLAY_INT(bufp, 4, 3,
j+FIRST_OL_ROW, ol_p
tr2->ol_supply_w_id);
                        bufp += DISPLAY_INT(bufp, 5, 11,
j+FIRST_OL_ROW, ol_
ptr2->ol_i_id);
                        bufp += DISPLAY_INT(bufp, 2, 45,
j+FIRST_OL_ROW, ol_
ptr2->ol_quantity);
                        bufp += DISPLAY(bufp, 3,
i+FIRST_OL_ROW, "    ");
                        bufp += DISPLAY(bufp, 11,
i+FIRST_OL_ROW, "     ");
                        bufp += DISPLAY(bufp, 45,
i+FIRST_OL_ROW, "  ");
                        *bufp++ = ’\0’;
                        PAINTSCRLEN(output_screen, bufp -
output_screen);
                    }
                } else {
                    *pos = 2 + 3 * i;
                    PAINTSCR(INCOMPLINE_MSG);
                    message = TRUE;
                    iNO->o_ol_cnt = 0;
                    return FALSE;
                }

            } else {
                blank_line++;
                blank_array[i] = 1;
            }
        }

        if (!iNO->o_ol_cnt) {
            *pos = 2;
            PAINTSCR(MANDATORY_MSG);
            message = TRUE;
            iNO->o_ol_cnt = 0;
            return FALSE;
        }
        done = TRUE;
    }
    return done;
}


int paym_val(int *pos)
{
    int     done = FALSE;
    iPT->w_id = w_id;
    if (iPT->d_id <= 0) {
        *pos = 0;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    } else if (iPT->c_w_id <= 0) {
        *pos = 2;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    } else if (iPT->c_d_id <= 0) {
        *pos = 3;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    } else if (int_h_amount <= 0) {
        *pos = 5;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    } else if (iPT->c_id <= 0) {
        if (iPT->c_last[0] == ‘\0’) {
            message = TRUE;
            PAINTSCR(ID_OR_LAST_MSG);
            *pos = 1;
        } else {
            done = TRUE;
        }
    } else
        done = TRUE;
    iPT->h_amount = ((float)int_h_amount) / 100.0 ;
    return done;
}

int ords_val(int *pos)
{
    int     done = FALSE;
    iOS->w_id = w_id;
    if (iOS->d_id <= 0) {
        *pos = 0;
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    } else if (iOS->c_id <= 0) {
        if (iOS->c_last[0] == ‘\0’) {
            message = TRUE;
            PAINTSCR(ID_OR_LAST_MSG);
            *pos = 1;
        } else {
            done = TRUE;
        }
    } else
        done = TRUE;
    return done;
}

int del_val(int *pos)
{
    int     done = FALSE;
    iDY->w_id = w_id;
    if (iDY->o_carrier_id <= 0) {
        message = TRUE;
        PAINTSCR(MANDATORY_MSG);
    } else {
        time(&iDY->queue_time);
        done = TRUE;
    }
    return done;
}

int stock_val(int *pos)
{
    int     done = FALSE;
    iSL->w_id = w_id;
    iSL->d_id = d_id;
    if (iSL->threshold <= 0) {
        message = TRUE;
```

```
            PAINTSCR(MANDATORY_MSG);
        } else
            done = TRUE;
        return done;
}

int wd_val(int *pos)
{
        int         done = FALSE;

        if (iWD->w_id == 0 | | iWD->d_id == 0) {
            message = TRUE;
            PAINTSCR(MANDATORY_MSG);
        } else {
            w_id = iWD->w_id ;
            d_id = iWD->d_id;
            done = TRUE;
        }
        return done;
}
void setup_wd()
{
        io_elem         *p;
        char            buf[128];
        void            setup_io_elems();
        setraw();
        setup_screen_buffer(&Forms[5], 5);

        p = Forms[WD].input_elems;
        p++->dptr = &iWD->w_id;
        p++->dptr = &iWD->d_id;

        CLRSCN(buf);
        PAINTSCR(buf);
}

void set_display()
{
        int             i;
        char            buf[128];
        void            setup_io_elems();
        for (i = 0; i < MAX_FORMS; i++)
            setup_screen_buffer(&Forms[i], i);

        setup_io_elems();

        CLRSCN(buf);
        PAINTSCR(buf);
}

void display_screen(int screen_num)
{
        if (PAINTSCRLEN(Forms[screen_num].blank_form,
           Forms[screen_num].blank_formlen) == -1)
            syserr("Can't write out form");
}

void Send_Menu()
{
        if (PAINTSCRLEN(menu_buf, menu_buflen) == -1)
            syserr("Can't send menu");
}
```

```
void setup_io_elems()
{
        io_elem         *p;
        int             i;
        p = Forms[NEWORDER].input_elems;
        p++->dptr = &iNO->d_id;
        p++->dptr = &iNO->c_id;
        for (i = 0; i < 15; i++) {
            p++->dptr = &iNO->o_ol[i].ol_supply_w_id;
            p++->dptr = &iNO->o_ol[i].ol_i_id;
            p++->dptr = &iNO->o_ol[i].ol_quantity;
        }
        p = Forms[PAYMENT].input_elems;
        p++->dptr = &iPT->d_id;
        p++->dptr = &iPT->c_id;
        p++->dptr = &iPT->c_w_id;
        p++->dptr = &iPT->c_d_id;
        p++->dptr = (int *) &iPT->c_last[0];
        p->dptr = &int_h_amount;

        p = Forms[ORDSTAT].input_elems;
        p++->dptr = &iOS->d_id;
        p++->dptr = &iOS->c_id;
        p->dptr = (int *) &iOS->c_last[0];
        p = Forms[DELIVERY].input_elems;
        p->dptr = &iDY->o_carrier_id;
        p = Forms[STOCKLEV].input_elems;
        p->dptr = &iSL->threshold;
}
int
setup_screen_buffer(struct form_info * form_ptr, int txn_type)
{
        FILE            *ifile;
        const text_elem       *tbuf;
        char            *bufp;
        int             ct;
        char            input_display_buf[64];
        io_elem         *io_ptr;

        bufp = screen_buf;
        bufp += CLRSCN(bufp);
        tbuf = form_ptr->tp;
        while (tbuf->text) {
            bufp += DISPLAY(bufp, tbuf->y, tbuf->x, tbuf->text);
            tbuf++;
        }
        bufp += SWITCH_TO_UNDERL(bufp);

        ct = 0;
        for (io_ptr = form_ptr->input_elems; io_ptr->y != 999; io_ptr++) {

            strncpy(input_display_buf, blanks, io_ptr->len);
            input_display_buf[io_ptr->len] = '\0';
            bufp += DISPLAY(bufp, io_ptr->x, io_ptr->y, input_display_buf);
            ct++;

        }

        form_ptr->num_input_elems = ct;
        bufp += SWITCH_TO_NORMAL(bufp);

        if (txn_type == PAYMENT)
            bufp += DISPLAY_INT(bufp, 4, 12, 4, w_id);
```

```
    else if (txn_type != 5)
        bufp += DISPLAY_INT(bufp, 4, 12, 2, w_id);
    if (txn_type == STOCKLEV)
        bufp += DISPLAY_INT(bufp, 2, 29, 2, d_id);
    bufp += SWITCH_TO_UNDERL(bufp);
    *bufp++ = ‘\1’;
    *bufp = ‘\0’;
    form_ptr->blank_formlen = bufp - screen_buf + 1;
    if (!form_ptr->blank_form &&
      ((form_ptr->blank_form = malloc(form_ptr->blank_formlen)) ==
NULL)) {
        Clog(“setup_screen_buffer: malloc failed\n”);
        exit(1);
    }
    memcpy(form_ptr->blank_form, screen_buf, form_ptr-
>blank_formlen);
    memset(screen_buf, ‘\0’, form_ptr->blank_formlen);

}
int
read_integer(col, row, size, flags, data)
    int         col, row, size, flags, *data;
{
    int       exit_read_function = FALSE, previous_data_exists = FALSE;
    int         return_status = TAB, bytes_read = 0, i = 0, j = 0, k = 0,
                size1 = 0, cur_col = col;
    char        *bufp, temp[50];
    float       q;

    char         erase_field[20];

    strncpy(temp, “ “ , 1);
    bufp = screen_buf + screen_bufindex;

    if (curbuf_read == read_count | | curbuf_read == 0) {
        screen_buf[0] = ‘\0’;
        bufp += GOTOXY(bufp, col + size - 1, row);
        PAINTSCRLEN(screen_buf, bufp - screen_buf);
        bufp = screen_buf;
    }
    size1 = size;

    if (*data > 0)
        previous_data_exists = TRUE;

    while (exit_read_function == FALSE) {

        if (curbuf_read == read_count | | curbuf_read == 0) {
            curbuf_read = 0;
            read_count = read(tty_in, curbuf, sizeof(curbuf));
            if (read_count == 0)
                syserr(“TTY lost connection”);
        }

        if (message == TRUE) {
            bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, ERASE_MSG);
            message = FALSE;
        }
        if (previous_data_exists == TRUE) {

            if (curbuf[curbuf_read] == DELETE) {
                previous_data_exists = FALSE;
                strncpy(erase_field, blanks, size);
                erase_field[size] = ‘\0’;
                bufp += DISPLAY(bufp, col, row, erase_field);
                bufp += GOTOXY(bufp, col + size - 1, row);
            } else {
                if (curbuf[curbuf_read] < ‘0’ | | curbuf[curbuf_read] >
‘9’) {
                    exit_read_function = TRUE;
                    previous_data_exists = FALSE;
                    return_status = curbuf[curbuf_read];
                    curbuf[curbuf_read] = ‘\0’;
                } else {
                    previous_data_exists = FALSE;
                    strncpy(erase_field, blanks, size);
                    erase_field[size] = ‘\0’;
                    bufp += DISPLAY(bufp, col, row, erase_field);
                }
            }
        }
        while ((curbuf_read < read_count) && (exit_read_function ==
FALSE)) {
            if (payment_input == TRUE)
                size1 = size - 1;
            if ((curbuf[curbuf_read] >= ‘0’ && curbuf[curbuf_read] <=
‘9’) | | (curbuf[curbuf_read] == ‘.’)) {
                for (; curbuf_read < read_count &&
                  ((curbuf[curbuf_read] >= ‘0’
                  && curbuf[curbuf_read] <= ‘9’) | |
curbuf[curbuf_read] == ‘.’); curbuf_read++) {
                    if (curbuf_consumed < size1) {
                        temp[curbuf_consumed] =
curbuf[curbuf_read];
                        curbuf_consumed++;
                    }
                    else
                        OVERFLOW = TRUE;
                    curbuf[curbuf_read] = ‘\0’;
                }
                temp[curbuf_consumed] = ‘\0’;
                if (payment_input == TRUE) {
                    q = (atof(temp)) ;
                        bufp += DISPLAY_FLOAT(bufp, 2, (col +
size - 4), row, q);
                } else {
                    if (curbuf_consumed < size + 1)
                        bufp += DISPLAY(bufp, (col + size -
                           curbuf_consumed), row,
                                temp);
                    return_status = curbuf[curbuf_read];
                    cur_col++;
                }
            }
            else if (curbuf[curbuf_read] == TAB
                 | | curbuf[curbuf_read] == LF
                 | | curbuf[curbuf_read] == BACKTAB
                 | | curbuf[curbuf_read] == SUBMIT) {
                if (message == TRUE) {
                    bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, ERASE_MSG);
                    message = FALSE;
                }
                temp[curbuf_consumed] = ‘\0’;
                if (payment_input == TRUE) {
                    q = atof(temp);
                    *data = q*100;
                }
                else {
                    *data = atoi(temp);
                }
                exit_read_function = TRUE;
                return_status = curbuf[curbuf_read];
                curbuf[curbuf_read] = ‘\0’;
                curbuf_read++;
                curbuf_consumed = 0;
            }
            else if (curbuf[curbuf_read] == DELETE) {
                if (payment_input == TRUE) {
                    if (curbuf_consumed != 0)
                        curbuf_consumed--;
                    if (message == TRUE) {
                        bufp += DISPLAY(bufp,
MESSAGE_COL,
                                MESSAGE_ROW,
                                ERASE_MSG);
                        message = FALSE;
                    }
                    OVERFLOW = FALSE;
                    PAINTSCR(screen_buf);
                    temp[curbuf_consumed] = ‘\0’;
                    q = atof(temp);
                    curbuf[curbuf_read] = ‘\0’;
                    strncpy(erase_field, blanks, size);
                    erase_field[size] = ‘\0’;
                    bufp = screen_buf;
                    screen_bufindex = 0;
                    bufp += DISPLAY(bufp, col, row,
erase_field);
                    if (curbuf_consumed < 3)
                        bufp += DISPLAY_FLOAT(bufp, 2,
                         (col + size - 4), row, q);
                    else
                        bufp += DISPLAY_FLOAT(bufp, 2,
                                   (col + size -
curbuf_consumed - 1), row, q);
                    if (cur_col != 0)
                        cur_col--;
                    if (curbuf_read < 40)
                        curbuf_read++;
                    bufp += GOTOXY(bufp, col + size, row);
                } else {
                    if (curbuf_consumed != 0)
                        curbuf_consumed--;
                    curbuf[curbuf_read] = ‘\0’;
                    curbuf_read++;
                    if (message == TRUE) {
                        bufp += DISPLAY(bufp,
MESSAGE_COL,
                                MESSAGE_ROW,
                                ERASE_MSG);
                        message = FALSE;
                    }
                    OVERFLOW = FALSE;
                    PAINTSCR(screen_buf);
                    temp[curbuf_consumed] = ‘\0’;
                    strncpy(erase_field, blanks, size);
                    erase_field[size] = ‘\0’;
                    bufp = screen_buf;
                    screen_bufindex = 0;
                    bufp += DISPLAY(bufp, col, row, erase_field);
                    bufp += DISPLAY(bufp, (col + size -
                       curbuf_consumed), row, temp);
                    if (cur_col != 0)
                        cur_col--;
                    bufp += GOTOXY(bufp, col + size, row);
                }
            }
            else if (curbuf[curbuf_read] == QUIT) {
                temp[0] = ‘\0’;
                return_status = QUIT;
                curbuf[curbuf_read] = ‘\0’;
                exit_read_function = TRUE;
            } else {
                if (message == FALSE) {
                    bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, INVALID_MSG);
                    bufp += GOTOXY(bufp, col + size, row);
                    PAINTSCR(screen_buf);
                    bufp = screen_buf;
                    screen_bufindex = 0;
                    message = TRUE;
                }
                curbuf_read++;
            }
        }
        if (OVERFLOW == TRUE && exit_read_function == FALSE) {
            if (message == FALSE) {
                bufp += DISPLAY(bufp, MESSAGE_COL,
                       MESSAGE_ROW, EXC_FLD_LIM_MSG);
                PAINTSCR(screen_buf);
                bufp = screen_buf;
                screen_bufindex = 0;
                message = TRUE;
            }
            *data = atoi(temp);
            return_status = curbuf[curbuf_read];
            curbuf[curbuf_read] = ‘\0’;
            curbuf_read = 0;
            OVERFLOW = FALSE;
        } else {
            screen_bufindex = bufp - screen_buf;
            if ((curbuf_read == read_count) | | (curbuf_read == 0)
              | | (screen_bufindex > SCRBUF_LEN - CURBUFLEN)) {
                PAINTSCRLEN(screen_buf, screen_bufindex);
                screen_bufindex = 0;
                bufp = screen_buf;
            }
        }
    }
    if (message == TRUE) {
        bufp += DISPLAY(bufp, MESSAGE_COL,
                MESSAGE_ROW, ERASE_MSG);
        message = FALSE;
        PAINTSCR(screen_buf);
        bufp = screen_buf;
        screen_bufindex = 0;
    }
    return (return_status);
}
int
```

```
read_string(col, row, size, flags, data)
    int         col, row, size, flags;
    char        *data;
{
    int      exit_read_function = FALSE, previous_data_exists = FALSE,
data_full = FALSE;
    int         return_status = TAB, bytes_read = 0, i = 0, j = 0,
                size_tot = 0;
    char        *bufp, temp[80];
    char         erase_field[20];

    strncpy(temp, “\0”, 1);
    curbuf_consumed = 0;
    bufp = screen_buf + screen_bufindex;

    if (curbuf_read == read_count | | curbuf_read == 0) {
        screen_buf[0] = ‘\0’;
        bufp += GOTOXY(bufp, col, row);
        PAINTSCRLEN(screen_buf, bufp - screen_buf);
        bufp = screen_buf;
    }
    if ((*(char *) data) != ‘\0’)
        previous_data_exists = TRUE;
    while (exit_read_function == FALSE) {
        if (curbuf_read == read_count) {
            curbuf_read = 0;
            read_count = read(tty_in, curbuf, size - size_tot);
            if (read_count == 0)
                syserr(“TTY lost connection”);
        }
        if (message == TRUE) {
            bufp += DISPLAY(bufp, MESSAGE_COL,
                    MESSAGE_ROW, ERASE_MSG);
            message = FALSE;
        }
        if (previous_data_exists == TRUE) {
            if (curbuf[curbuf_read] == DELETE) {
                previous_data_exists = FALSE;
                strncpy(erase_field, blanks, size);
                erase_field[size] = ‘\0’;
                bufp += DISPLAY(bufp, col, row, erase_field);
                bufp += GOTOXY(bufp, col, row);
            } else {
                if (curbuf[curbuf_read] < ‘ ‘ | | curbuf[curbuf_read]
> ‘~’) {

                    exit_read_function = TRUE;
                    previous_data_exists = FALSE;
                    return_status = curbuf[curbuf_read];
                    curbuf[curbuf_read] = ‘\0’;

                } else {
                    previous_data_exists = FALSE;
                    strncpy(erase_field, blanks, size);
                    erase_field[size] = ‘\0’;
                    bufp += DISPLAY(bufp, col, row, erase_field);
                    bufp += GOTOXY(bufp, col, row);
                }
            }
        }
        while ((curbuf_read < read_count) && (exit_read_function ==
FALSE)) {
            if (curbuf[curbuf_read] >= ‘ ‘ && curbuf[curbuf_read] <=
‘~’) {
```

```
                for (; curbuf[curbuf_read] >= ‘ ‘
                  && curbuf[curbuf_read] <= ‘~’; curbuf_read++) {
                    if (curbuf_consumed < size) {
                        temp[curbuf_consumed] =
curbuf[curbuf_read];

                        curbuf_consumed++;
                    }
                    else
                        OVERFLOW = TRUE;
                    curbuf[curbuf_read] = ‘\0’;
                }
                temp[curbuf_consumed] = ‘\0’;
                bufp += DISPLAY(bufp, col, row, temp);
                return_status = curbuf[curbuf_read];
            } else if (curbuf[curbuf_read] == TAB
                   | | curbuf[curbuf_read] == LF
                   | | curbuf[curbuf_read] == BACKTAB
                   | | curbuf[curbuf_read] == SUBMIT) {
                if (curbuf_consumed > 0) {
                    if (message == TRUE) {
                        bufp += DISPLAY(bufp, MESSAGE_COL,
                                MESSAGE_ROW,
                                ERASE_MSG);
                        message = FALSE;
                    }
                    temp[curbuf_consumed] = ‘\0’;
                    strcpy(data, temp);
                    exit_read_function = TRUE;
                    return_status = curbuf[curbuf_read];
                    curbuf[curbuf_read] = ‘\0’;
                    curbuf_read++;
                    curbuf_consumed = 0;
                } else {
                    if (message == TRUE) {
                        bufp += DISPLAY(bufp, MESSAGE_COL,
                                MESSAGE_ROW,
                                ERASE_MSG);
                        message = FALSE;
                    }
                    temp[curbuf_consumed] = ‘\0’;
                    strcpy(data, temp);
                    exit_read_function = TRUE;
                    return_status = curbuf[curbuf_read];
                    curbuf[curbuf_read] = ‘\0’;
                    curbuf_read++;
                }
            } else if (curbuf[curbuf_read] == DELETE) {
                for (curbuf_read = curbuf_read; curbuf[curbuf_read]
==

                  DELETE
                  ; curbuf_read++) {
                    curbuf[curbuf_read] = ‘\0’;
                    temp[curbuf_consumed - 1] = ‘\0’;

                    if (curbuf_consumed != 0)
                        curbuf_consumed--;
                }
                if (curbuf_consumed >= 0) {
                    bufp += BLANK_UNDERLINE(bufp, col, row, “
“);

                    bufp += DISPLAY(bufp, col, row, temp);
                    PAINTSCR(screen_buf);
                    bufp = screen_buf;
```

```
                                        screen_bufindex = 0;
                                } else {
                                        if (message == FALSE) {
                                                bufp += DISPLAY(bufp,
MESSAGE_COL,
                                                                MESSAGE_ROW,
                                                            EXC_FLD_LIM_MSG);
                                                bufp += BEEP(bufp);
                                                PAINTSCR(screen_buf);
                                                bufp = screen_buf;
                                                screen_bufindex = 0;
                                                message = TRUE;
                                        }
                                        curbuf[curbuf_read] = ‘\0’;
                                        curbuf_read = 0;
                                }
                        } else if (curbuf[curbuf_read] == QUIT) {
                                temp[0] = ‘\0’;
                                return_status = QUIT;
                                curbuf[curbuf_read] = ‘\0’;
                                exit_read_function = TRUE;
                        } else {
                                if (message == FALSE) {
                                        bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, INVALID_MSG);
                                        bufp += GOTOXY(bufp, col, row);
                                        message = TRUE;
                                }
                                curbuf_read++;
                        }
                }
                if (OVERFLOW == TRUE && exit_read_function == FALSE)
                {
                        if (message == FALSE) {
                                bufp += DISPLAY(bufp, MESSAGE_COL,
                                           MESSAGE_ROW,
EXC_FLD_LIM_MSG);
                                PAINTSCR(screen_buf);
                                bufp = screen_buf;
                                screen_bufindex = 0;
                                message = TRUE;
                        }
                        OVERFLOW = FALSE;
                        temp[curbuf_consumed] = ‘\0’;
                        strcpy(data, temp);
                        curbuf_consumed--;
                        return_status = curbuf[curbuf_read];
                } else {
                        screen_bufindex = bufp - screen_buf;
                        if ((curbuf_read == read_count) | | (curbuf_read == 0)
                            | | (screen_bufindex > SCRBUF_LEN -
CURBUFLEN)) {
                                PAINTSCRLEN(screen_buf, screen_bufindex);
                                screen_bufindex = 0;
                                bufp = screen_buf;
                        }
                }
        }
        if (message == TRUE) {
                bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW,
                                ERASE_MSG);
                message = FALSE;
                PAINTSCR(screen_buf);
```

```
                screen_bufindex = 0;
        }
        return (return_status);
}
void display_newo();
void display_paym();
void display_ords();
void display_del();
void display_stock();
void (*p_print_function[]) () = {
                &display_newo,
                &display_paym,
                &display_ords,
                &display_del,
                &display_stock
};
display_output(int txn_type)
{
        char            c;

        (p_print_function[txn_type]) ();
        read(tty_in, &c, 1);
}
void
display_newo()
{
        struct no_itm_struct *ol_ptr, *ool;

        char            *bufp;
        int             i, r;

        bufp = output_screen;

        if (oNO->status == ‘\0’) {

                PAINTSCR(EXECUTION_STATUS_MSG);
                return;

        } else {

                bufp += SWITCH_TO_NORMAL(bufp);
                bufp += DISPLAY(bufp, 61, 2, oNO->o_entry_d);
                bufp += DISPLAY(bufp, 25, 3, oNO->c_last);
                bufp += DISPLAY(bufp, 52, 3, oNO->c_credit);
                bufp += DISPLAY_FLOAT(bufp, 5, 64, 3, oNO->c_discount);
                bufp += DISPLAY_INT(bufp, 8, 15, 4, oNO->o_id);
                bufp += DISPLAY_INT(bufp, 2, 42, 4, oNO->o_ol_cnt);
                bufp += DISPLAY_FLOAT(bufp, 5, 59, 4, oNO->w_tax);
                bufp += DISPLAY_FLOAT(bufp, 5, 74, 4, oNO->d_tax);
                ol_ptr = iNO->o_ol;
                ool = oNO->o_ol;

                for (i = 0, r = FIRST_OL_ROW; i < iNO->o_ol_cnt;
                    r++, i++, ol_ptr++, ool++) {
                        bufp += DISPLAY(bufp, 19, r, ool->i_name);
                        bufp += DISPLAY_INT(bufp, 3, 51, r, ool->s_quantity);
                        bufp += DISPLAY(bufp, 58, r, ool->brand);
                        bufp += DISPLAY_MONEY(bufp, 6, 62, r, ool->i_price);
                        bufp += DISPLAY_MONEY(bufp, 7, 71, r, ool->ol_amount);
                }

                bufp += DISPLAY_MONEY(bufp, 8, 70, 22, oNO->total);
                bufp += DISPLAY(bufp, 19, 22, oNO->status);
        bufp += DISPLAY(bufp, 23, 75, “**((“);
        *bufp++ = ‘\0’;
        PAINTSCRLEN(output_screen, bufp - output_screen);
    }
}
void
display_paym()
{
    char        *bufp, temp[51], tempbuf2[201];
    char        *make_phone(char *), *make_zip(char *);
    bufp = output_screen;
    bufp += SWITCH_TO_NORMAL(bufp);
    bufp += DISPLAY(bufp, 7, 2, oPT->h_date);
    bufp += DISPLAY(bufp, 1, 5, oPT->w_street_1);
    bufp += DISPLAY(bufp, 1, 6, oPT->w_street_2);
    bufp += DISPLAY(bufp, 1, 7, oPT->w_city);
    bufp += DISPLAY(bufp, 22, 7, oPT->w_state);
    bufp += DISPLAY(bufp, 25, 7, make_zip(oPT->w_zip));
    bufp += DISPLAY(bufp, 42, 5, oPT->d_street_1);
    bufp += DISPLAY(bufp, 42, 6, oPT->d_street_2);
    bufp += DISPLAY(bufp, 42, 7, oPT->d_city);
    bufp += DISPLAY(bufp, 63, 7, oPT->d_state);
    bufp += DISPLAY(bufp, 66, 7, make_zip(oPT->d_zip));
    bufp += DISPLAY_INT(bufp, 4, 11, 9, oPT->c_id);
    bufp += DISPLAY(bufp, 29, 10, oPT->c_last);
    bufp += DISPLAY(bufp, 9, 10, oPT->c_first);
    bufp += DISPLAY(bufp, 26, 10, oPT->c_middle);
    bufp += DISPLAY(bufp, 9, 11, oPT->c_street_1);
    bufp += DISPLAY(bufp, 9, 12, oPT->c_street_2);
    bufp += DISPLAY(bufp, 9, 13, oPT->c_city);
    bufp += DISPLAY(bufp, 30, 13, oPT->c_state);
    bufp += DISPLAY(bufp, 33, 13, make_zip(oPT->c_zip));
    bufp += DISPLAY(bufp, 58, 10, oPT->c_since);
    bufp += DISPLAY(bufp, 58, 11, oPT->c_credit);
    bufp += DISPLAY_FLOAT(bufp, 5, 58, 12, oPT->c_discount);
    bufp += DISPLAY(bufp, 58, 13, make_phone(oPT->c_phone));
    bufp += DISPLAY_MONEY(bufp, 14, 55, 15, oPT->c_balance);
    bufp += DISPLAY_MONEY(bufp, 13, 17, 16, oPT->c_credit_lim);

    if (oPT->c_data_1[0] != ‘ ‘ | | oPT->c_data_1[0] != ‘\0’) {
        bufp += DISPLAY50(bufp, 12, 18, oPT->c_data_1);
        bufp += DISPLAY50(bufp, 12, 19, oPT->c_data_2);
        bufp += DISPLAY50(bufp, 12, 20, oPT->c_data_3);
        bufp += DISPLAY50(bufp, 12, 21, oPT->c_data_4);
    }
    if (!oPT->h_date)
        bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW - 2,
BAD_INPUTS);
    bufp += DISPLAY(bufp, 23, 75, “**((“);
    *bufp++ = ‘\0’;
    PAINTSCRLEN(output_screen, bufp - output_screen);
}
void
display_ords()
{
    struct ord_itm_struct *sol;
    char        *bufp;
    int         i = 0, r = 8;

    bufp = output_screen;
    bufp += SWITCH_TO_NORMAL(bufp);
    bufp += DISPLAY_INT(bufp, 4, 11, 3, oOS->c_id);
    bufp += DISPLAY(bufp, 44, 3, oOS->c_last);
```

```
    bufp += DISPLAY(bufp, 24, 3, oOS->c_first);
    bufp += DISPLAY(bufp, 41, 3, oOS->c_middle);
    bufp += DISPLAY_MONEY(bufp, 9, 15, 4, oOS->c_balance);
    bufp += DISPLAY_INT(bufp, 8, 15, 6, oOS->o_id);
    bufp += DISPLAY(bufp, 38, 6, oOS->o_entry_d);
    bufp += DISPLAY_INT(bufp, 2, 76, 6, oOS->o_carrier_id);


    for (i = 0; i < oOS->ol_cnt; i++) {

        sol = &oOS->s_ol[i];

        if (sol->ol_supply_w_id > 0) {
            bufp += DISPLAY_INT(bufp, 4, 3, r, sol->ol_supply_w_id);
            bufp += DISPLAY_INT(bufp, 6, 14, r, sol->ol_i_id);
            bufp += DISPLAY_INT(bufp, 2, 25, r, sol->ol_quantity);
            bufp += DISPLAY_MONEY(bufp, 8, 32, r, sol->ol_amount);
            bufp += DISPLAY(bufp, 47, r, sol->ol_delivery_d);
            r++;
        }
    }
    if (!oOS->ol_cnt)
        bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW - 2,
BAD_INPUTS);

    bufp += DISPLAY(bufp, 23, 75, “**((“);
    *bufp++ = ‘\0’;
    PAINTSCRLEN(output_screen, bufp - output_screen);
}
void
display_del()
{
    char        *bufp;

    bufp = output_screen;
    bufp += sprintf(bufp,”%s”,DELIVERY_QUEUED_MSG);
    bufp += DISPLAY(bufp, 23, 75, “**((“);
    *bufp++ = ‘\0’;
    PAINTSCRLEN(output_screen, bufp - output_screen);
    Clog(“DBG: Screen output chars = %d\n”, (bufp -
}
void
display_stock()
{
    char        *bufp;
    bufp = output_screen;
    bufp += SWITCH_TO_NORMAL(bufp);

    bufp += DISPLAY_INT(bufp, 3, 12, 6, oSL->low_stock);
    bufp += DISPLAY(bufp, 23, 75, “**((“);
    *bufp++ = ‘\0’;
    PAINTSCRLEN(output_screen, bufp - output_screen);
}
char        *
make_phone(char *data)
{
    static char     tempphone[20];
    strncpy(tempphone, data, 6);
    tempphone[6] = ‘-’;
    strncpy(&tempphone[7], &data[6], 3);
    tempphone[10] = ‘-’;
    strncpy(&tempphone[11], &data[9], 3);
    tempphone[14] = ‘-’;
```

```
        strncpy(&tempphone[15], &data[12], 4);
        tempphone[19] = ‘\0’;
        return tempphone;
}
char            *
make_zip(char *data)
{
        static char     temp[10];
        strncpy(temp, data, 5);
        temp[5] = ‘-’;
        strncpy(&temp[6], &data[5], 4);
        temp[10] = ‘\0’;
        return temp;
}

 /*
  * Copyright (c) 1995, 1996, 1997, 1998, 1999 by Sun Microsystems, Inc.
  */

#include <sys/termio.h>
extern int      tty_in;
extern int      tty_out;
#define MAX_FORMS 6
#define MESSAGE_ROW 24
#define MESSAGE_COL 1
#define RTE_SYNCH_CHARACTER ‘\1’
#define SCRBUF_LEN 1536
#define FIRST_OL_ROW 7
#define CLRSCN(buf) sprintf(buf,”\033[H\033[2J”)
#define DISPLAY_INT(buf,wid,x,y,ip)
sprintf(buf,”\033[%d;%dH%*.1d”, y, x,wid,ip)
#define DISPLAY_MONEY(buf,wid,x,y,fp)
sprintf(buf,”\033[%d;%dH$%#*.2f”,y,x,wid,fp)
#define DISPLAY_FLOAT(buf,wid,x,y,fp)
sprintf(buf,”\033[%d;%dH%#*.2f”, y, x,wid,fp)
#define DISPLAY(buf,x,y,txt) sprintf(buf,”\033[%d;%dH%s”, y, x,txt)
#define DISPLAY50(buf,x,y,txt) sprintf(buf,”\033[%d;%dH%50.50s”, y,
x,txt)
#define PAINTSCR(buf) write(tty_out,buf,strlen(buf))
#define PAINTSCRLEN(buf, len) write(tty_out,buf,len)
#define SWITCH_TO_NORMAL(buf) sprintf(buf,”\033[m”)
#define SWITCH_TO_UNDERL(buf) sprintf(buf,”\033[4m”)
#define GOTOXY(buf,x,y) sprintf(buf,”\033[%d;%dH”, y, x)
#define BEEP(buf) sprintf(buf,”\007”)
#define BLANK_UNDERLINE(buf,x,y,txt)
sprintf(buf,”\033[4m;\033[%d;%dH%s”,y,x,txt);

#define CLRSCN_STR “‘\033[H\033[2J’
#define DISPLAY_STR(x,y,txt) ‘\033[/**/y;/**/xH/**/txt’


#define CANCELLED 3
#define PREVIOUS_FIELD 4

#define BACKTAB 2
#define DELETE 8
#define ESCAPE 27
#define LF 10
#define QUIT 3
#define SPACE 32
#define SUBMIT 13

#define TAB 9
```

```
#define UNDERLINE 95
#define LEAVE_SCREEN_MIN 300
#define LEAVE_SCREEN_TIMEOUT 2
static int      curbuf_consumed = 0;
static int      curbuf_read = 0;
static int      read_count = 0;
#defineCURBUFLEN300
static char     curbuf[CURBUFLEN];
static BOOLEAN  OVERFLOW = FALSE;
static BOOLEAN  message;
BOOLEAN         payment_input = FALSE;
static struct termio tbufsave;
extern void     syserr();
void            Init_Screen();
void            display_screen_array(int);
void            Send_Menu();
int             Get_Menu_Input();

typedef struct {
        int             y;
        int             x;
        int             len;
        int             flags;
        int            *dptr;
        int             (*fptr) ();
}               io_elem;

int             int_h_amount;
const static char     MANDATORY_MSG[] =
“\033[24;1H\033[mMandatory data field! Please enter data.”;
const static char     INVALID_MSG[] =
“\007\033[24;1HAn invalid character was entered. Please enter again.”;
const static char     ERASE_MSG[] = “\033[24;1H\033[K\033[4m”;
const static char     MIN1DIGIT_MSG[] = “\033[24;1H\033[mYou must enter
atleast 1 digit. Please reenter.\033[4m\1”;
const static char     BAD_INPUTS[] = “ #### Bad input data was entered -- Select
again #### \1”;
const static char     INCOMPLINE_MSG[] = “\033[24;1H\033[mOrder line is
incomplete. Please complete the whole line.\033[4m\1”;
const static char     ID_OR_LAST_MSG[] = “\033[24;1H\033[mYou must enter
either the Last Name or the Customer Number.\033[4m\1”;
const static char     EXC_MAX_LFT_DEC_DGT_MSG[] =
“\033[24;1H\033[mMaximum digits left of decimal point already entered. ‘.’
expected\033[4m\1”;
const static char     EXC_FLD_LIM_MSG[] = “\007\033[24;1H\033[mMaximum
digits already entered. Tab or <CR> expected\033[4m\1”;
const static char     EXECUTION_STATUS_MSG[] = “\033[m\033[22;18HItem
number is not valid”;
const static char     DELIVERY_QUEUED_MSG[] = “\033[m\033[6;19HDelivery
has been queued”;
int             read_integer(int, int, int, int, int *);
int             read_money(int, int, int, int, float *);
int             read_string(int, int, int, int, char *);
char             menu_buf[] = “\033[H\033[J\033[mNew-Order(n)  Payment(p)
Order-Status(o)  Delivery(d)  Stock-Level(s)  Exit(e)”;

int            menu_buflen = sizeof (menu_buf);
io_elem          neworder_inputs[] = {
        2, 29, 2, 0, 0, &read_integer,
        3, 12, 4, 0, 0, &read_integer,
        7, 3, 4, 0, 0, &read_integer,
        7, 10, 6, 0, 0, &read_integer,
        7, 45, 2, 0, 0, &read_integer,
        8, 3, 4, 0, 0, &read_integer,
        8, 10, 6, 0, 0, &read_integer,
        8, 45, 2, 0, 0, &read_integer,
        9, 3, 4, 0, 0, &read_integer,
        9, 10, 6, 0, 0, &read_integer,
        9, 45, 2, 0, 0, &read_integer,
        10, 3, 4, 0, 0, &read_integer,
        10, 10, 6, 0, 0, &read_integer,
        10, 45, 2, 0, 0, &read_integer,
        11, 3, 4, 0, 0, &read_integer,
        11, 10, 6, 0, 0, &read_integer,
        11, 45, 2, 0, 0, &read_integer,
        12, 3, 4, 0, 0, &read_integer,
        12, 10, 6, 0, 0, &read_integer,
        12, 45, 2, 0, 0, &read_integer,
        13, 3, 4, 0, 0, &read_integer,
        13, 10, 6, 0, 0, &read_integer,
        13, 45, 2, 0, 0, &read_integer,
        14, 3, 4, 0, 0, &read_integer,
        14, 10, 6, 0, 0, &read_integer,
        14, 45, 2, 0, 0, &read_integer,
        15, 3, 4, 0, 0, &read_integer,
        15, 10, 6, 0, 0, &read_integer,
        15, 45, 2, 0, 0, &read_integer,
        16, 3, 4, 0, 0, &read_integer,
        16, 10, 6, 0, 0, &read_integer,
        16, 45, 2, 0, 0, &read_integer,
        17, 3, 4, 0, 0, &read_integer,
        17, 10, 6, 0, 0, &read_integer,
        17, 45, 2, 0, 0, &read_integer,
        18, 3, 4, 0, 0, &read_integer,
        18, 10, 6, 0, 0, &read_integer,
        18, 45, 2, 0, 0, &read_integer,
        19, 3, 4, 0, 0, &read_integer,
        19, 10, 6, 0, 0, &read_integer,
        19, 45, 2, 0, 0, &read_integer,
        20, 3, 4, 0, 0, &read_integer,
        20, 10, 6, 0, 0, &read_integer,
        20, 45, 2, 0, 0, &read_integer,
        21, 3, 4, 0, 0, &read_integer,
        21, 10, 6, 0, 0, &read_integer,
        21, 45, 2, 0, 0, &read_integer,
        999
};
io_elem         payment_inputs[] = {
        4, 52, 2, 0, 0, &read_integer,
        9, 11, 4, 0, 0, &read_integer,
        9, 33, 4, 0, 0, &read_integer,
        9, 54, 2, 0, 0, &read_integer,
        10, 29, 16, 0, 0, &read_string,
        15, 24, 7, 0, 0, &read_integer,
        999
};
io_elem         ordstat_inputs[] = {
        2, 29, 2, 0, 0, &read_integer,
        3, 11, 4, 0, 0, &read_integer,
        3, 44, 16, 0, 0, &read_string,
        999
};
io_elem         delivery_inputs[] = {
        4, 17, 2, 0, 0, &read_integer,
        999
};
io_elem         stocklev_inputs[] = {
        4, 24, 2, 0, 0, &read_integer,
        999
};
io_elem         wd_inputs[] = {
        2, 16, 4, 0, 0, &read_integer,
        2, 43, 4, 0, 0, &read_integer,
        999
};

typedef struct {
        int             x;
        int             y;
        char            *text;
}               text_elem;

const text_elem         NO_text_elem[] = {
        1, 36, “New Order”,
        2, 1, “Warehouse:”,
        2, 19, “District:”,
        2, 55, “Date:”,
        3, 1, “Customer:”,
        3, 19, “Name:”,
        3, 44, “Credit:”,
        3, 57, “%Disc:”,
        4, 1, “Order Number:”,
        4, 25, “Number of Lines:”,
        4, 52, “W_tax:”,
        4, 67, “D_tax:”,
        6, 2, “Supp_W Item_Id Item Name”,
        6, 45, “Qty Stock B/G Price Amount”,
        22, 1, “Execution Status:”,
        22, 62, “Total:”,
        0
};
const text_elem         PT_text_elem[] = {
        1, 38, “Payment”,
        2, 1, “Date:”,
        4, 1, “Warehouse:”,
        4, 42, “District:”,
        9, 1, “Customer:”,
        9, 17, “Cust-Warehouse:”,
        9, 39, “Cust-District:”,
        10, 1, “Name:”,
        10, 50, “Since:”,
        11, 50, “Credit:”,
        12, 50, “%Disc:”,
        13, 50, “Phone:”,
        15, 1, “Amount Paid:”,
        15, 23, “$”,
        15, 37, “New Cust-Balance:”,
        16, 1, “Credit Limit:”,
        18, 1, “Cust-Data:”,
        0
};
const text_elem         OS_text_elem[] = {
        1, 35, “Order-Status”,
        2, 1, “Warehouse:”,
        2, 19, “District:”,
        3, 1, “Customer:”,
        3, 18, “Name:”,
        4, 1, “Cust-Balance:”,
        6, 1, “Order-Number:”,
        6, 26, “Entry-Date:”,
        6, 60, “Carrier_Number:”,
        7, 1, “Supply-W”,
        7, 14, “Item-Id”,
        7, 25, “Qty”,
        7, 33, “Amount”,
        7, 45, “Delivery-Date”,
        0
};
const text_elem     DY_text_elem[] = {
        1, 38, “Delivery”,
        2, 1, “Warehouse:”,
        4, 1, “Carrier Number:”,
        6, 1, “Execution Status:”,
        0
};
const text_elem     SL_text_elem[] = {
        1, 38, “Stock-Level”,
        2, 1, “Warehouse:”,
        2, 19, “District:”,
        4, 1, “Stock Level Threshold:”,
        6, 1, “low stock:”,
        0
};
const text_elem     WD_text_elem[] = {
        2, 1, “Warehouse:”,
        2, 26, “District:”,
        0
};
struct form_info {
        const text_elem     *tp;
        char            *blank_form;
        int             blank_formlen;
        io_elem         *input_elems;
        int             num_input_elems;
};

char            output_screen[SCRBUF_LEN];

struct form_info Forms[MAX_FORMS] = {
        {NO_text_elem, 0, 0, neworder_inputs, 0},
        {PT_text_elem, 0, 0, payment_inputs, 0},
        {OS_text_elem, 0, 0, ordstat_inputs, 0},
        {DY_text_elem, 0, 0, delivery_inputs, 0},
        {SL_text_elem, 0, 0, stocklev_inputs, 0},
        {WD_text_elem, 0, 0, wd_inputs, 0}
};
/*
 * Copyright (c) 1995, 1996, 1997, 1998, 1999 by Sun Microsystems, Inc.
 */

#include <stdio.h>
#include <stdarg.h>
#define BACKTAB 2
#define DELETE 127
#define ESCAPE 27
#define LF 10
#define QUIT 3
#define SPACE 32
#define SUBMIT 13
#define TAB 9
#define RTE_SYNCH_CHARACTER ‘\1’
```

```
static FILE    *clientlog;
static int     Clog_open = 0;
void
Clog(char *fmt,...)
{
}
void
SCREENlog(int *flag, char *screen)
{
        char            fname[100];
        int             i, char_ct;
        if (!Clog_open) {
                sprintf(fname, “%s/%s.%d”, getenv(“TMPDIR”), “CLIENTLOG”,
                        getpid());
                clientlog = fopen(fname, “w”);
                Clog_open = 1;
        }
        fprintf(clientlog, “** %d **\n”, flag);
        char_ct = 0;
        fprintf(clientlog, “SCR: “);
        for (i = 0; screen[i] != 0; char_ct++, i++) {
                switch (screen[i]) {
                case BACKTAB:
                        fprintf(clientlog, “<BACKTAB>”);
                        break;
                case DELETE:
                        fprintf(clientlog, “<DEL>”);
                        break;
                case ESCAPE:
                        fprintf(clientlog, “<ESC>”);
                        break;
                case LF:
                        fprintf(clientlog, “<LF>”);
                        break;
                case QUIT:
                        fprintf(clientlog, “<^C>”);
                        break;
                case SUBMIT:
                        fprintf(clientlog, “<CR>”);
                        break;
                case TAB:
                        fprintf(clientlog, “<TAB>”);
                        break;
                case RTE_SYNCH_CHARACTER:
                        fprintf(clientlog, “<^A>”);
                        break;
                default:
                        fprintf(clientlog, “%c”, screen[i]);
                }
                if (char_ct > 192) {
                        char_ct = 0;
                }
        }
        fprintf(clientlog, “\n”);
        fflush(clientlog);
}

void
syserr(msg)
        char            *msg;
{
        extern int      errno, sys_nerr;
        extern char     *sys_errlist[];
        extern char     tty_name[];
        fprintf(stderr, “\007ERROR: (%s) %s (%d”, tty_name, msg, errno);
        if (errno > 0 && errno < sys_nerr)
                fprintf(stderr, “;%s)\n”, sys_errlist[errno]);
        else
                fprintf(stderr, “)\n”);
        exit(1);
}


void
cleanup(msg)
        char            *msg;
{
extern int      tty_out;
extern int      tty_in;
char c;
                write(tty_out,msg,strlen(msg));
        read(tty_in, &c, 1);
}


/*
 * Copyright (c) 1995, 1996, 1997, 1998, 1999 by Sun Microsystems, Inc.
 */

#include <stdio.h>
#include <stdarg.h>
#include “tpcc_client.h”
#include <atmi.h>
#include “tpcc_tux.h”

const char              *svc_names[] = {“NEWO”, “PAYM”, “ORDS”, “DEL”,
“STOCK”};
int
Snd_Txn_To_Monitor(int txn_type)
{
        int             status;

        if (txn_type == DELIVERY) {
                if ( tpacall((char *)svc_names[txn_type], tuxibuf, ilen,
TPNOREPLY) == -1){
                        return (TPM_ERROR);
                }
                return(0);
        } else {
                if ( tpcall((char *)svc_names[txn_type], (char *)tuxibuf, ilen,
&tuxobuf, &olen, 0) == -1){
                        return (TPM_ERROR);
                }
                return (0);
        }
}

int Init_Monitor()
{
        char            *text;
        ilen = sizeof(struct io_tpcc);
        olen = sizeof(struct io_tpcc);
        if (tpinit(NULL) == -1) {
                tpmerror(“tpinit”, tperrno);
                return -1;
        }
        if ((tuxibuf = tpalloc(“CARRAY”, NULL, ilen)) == NULL) {
                tpmerror(“tpalloc”, tperrno);
                return (-1);
        }
        if ((tuxobuf = tpalloc(“CARRAY”, NULL, ilen)) == NULL) {
                tpmerror(“tpalloc”, tperrno);
                return (-1);
        }
        return (NULL);
}
Rundown_Monitor()
{
        int             status;

        tpfree(tuxibuf);
        status = tpterm();

}
tpmerror(char *service_called, int errnum)
{
        char            errmsg[256];
        fprintf(stderr, “\033[24;1H\033[mTUXEDO: Failed %s with error:
%s\n”,
                service_called, tpstrerror(errnum));
        fprintf(stderr, “\n”);
}
/*****************************************************************************
***************************** monitor.h *************************************
*****************************************************************************/
/* ** monitor.h -- All Tuxedo definitions and storage ** */
long            ilen;
long            olen;
int tty_in;
int tty_out;

char            *tuxibuf;
char            *tuxobuf;
extern void     Clog(char *,...);
#define oNO (&((info_t *) tuxobuf)->neworder)
#define oPT (&((info_t *) tuxobuf)->payment)
#define oOS (&((info_t *) tuxobuf)->ordstat)
#define oDY (&((info_t *) tuxobuf)->delivery)
#define oSL (&((info_t *) tuxobuf)->stocklev)
#define iNO (&((info_t *) tuxibuf)->neworder)
#define iPT (&((info_t *) tuxibuf)->payment)
#define iOS (&((info_t *) tuxibuf)->ordstat)
#define iDY (&((info_t *) tuxibuf)->delivery)
#define iSL (&((info_t *) tuxibuf)->stocklev)
#define iWD (&((info_t *) tuxibuf)->wd)

######################################################################
############
#
# tpcc_proc_case.sh
#
######################################################################
############
#
# This is the version of procs which was used in the Compaq-Sybase 11.G
# TPC-C benchmark (with the last-minute fixes) - March 26 1997
#
# This case script has the following changes from tpcc_proc_spec.sh
#   In new_order (both local and remote), the stock-item cursor, c_no_is
#   has been removed and replaced with an update-set-local variable
stmt.
# Also CASE statements replace the nested if's.
#
# Also modified delivery proc where the ol and order table cursors
have
#   been replaced by update_set_local_variable statements.
#
# In Payment procs, the cursor on customer, c_pay_c has been
removed. Instead.
# added two update statements (with set local variables).
#
# Reinstated c_find cursor to find cust_id given c_last;
# Stock_level is back o its "single query" state!
#
#
################################################################
##################
#
#!/bin/sh -f

# Stored procedure for TPC-C 3.2 on SQL Server 11.1 and later
# Copyright Sybase 1997, 1998, 1999
#
isql -Usa -P -e -n <<EOF
use tpcc
go
if exists ( SELECT name FROM sysobjects WHERE name =
'neworder_local')
        DROP PROC neworder_local
go

CREATE PROC neworder_local (
        @w_id        smallint,
        @d_id        tinyint,
        @c_id        int,
        @o_ol_cnttinyint,

        @i_idint = 0, @ol_qtytinyint = 0,
        @i_id2int = 0, @ol_qty2tinyint = 0,
        @i_id3int = 0, @ol_qty3tinyint = 0,
        @i_id4int = 0, @ol_qty4tinyint = 0,
        @i_id5int = 0, @ol_qty5tinyint = 0,
        @i_id6int = 0, @ol_qty6tinyint = 0,
        @i_id7int = 0, @ol_qty7tinyint = 0,
        @i_id8int = 0, @ol_qty8tinyint = 0,
        @i_id9int = 0, @ol_qty9tinyint = 0,
        @i_id10int = 0, @ol_qty10tinyint = 0,
        @i_id11int = 0, @ol_qty11tinyint = 0,
        @i_id12int = 0, @ol_qty12tinyint = 0,
        @i_id13int = 0, @ol_qty13tinyint = 0,
        @i_id14int = 0, @ol_qty14tinyint = 0,
        @i_id15int = 0, @ol_qty15tinyint = 0
)
as

declare
        @w_tax       real,          @d_tax       real,
        @c_last      char(16),@c_creditchar(2),
        @c_discountreal, @commit_flagtinyint,
        @c_ins_id int,

        @i_pricereal,
        @i_name      char(24),@i_data char(50),

        @s_quantitysmallint,
        @s_ytd       int,           @s_order_cntint,
        @s_dist      char(24),@s_datachar(50),

        @ol_numbertinyint,@o_idint,
        @o_entry_ddatetime,@b_gchar(1),
        @ol_amount real



begin

        begin transaction NO

        -- @#@# UPDATE district FROM district, warehouse, customer
        --

        UPDATE district
        SET     d_next_o_id = d_next_o_id + 1
                , @o_id       = d_next_o_id
                , @d_tax = d_tax
                , @commit_flag= 1
                , @ol_number= 0
                , @o_entry_d= getdate()
        WHERE     d_w_id= @w_id
                AND d_id  = @d_id

        while (@ol_number < @o_ol_cnt) begin
                SELECT @ol_number = @ol_number + 1
                ,@i_id = case @ol_number
         when 1 then @i_id2
                 when 2 then @i_id3
                 when 3 then @i_id4
                 when 4 then @i_id5
                 when 5 then @i_id6
                 when 6 then @i_id7
                 when 7 then @i_id8
                 when 8 then @i_id9
                 when 9 then @i_id10
                 when 10 then @i_id11
                 when 11 then @i_id12
                 when 12 then @i_id13
                 when 13 then @i_id14
                 when 14 then @i_id15
                 else @i_id
                 end
                , @ol_qty = case @ol_number
                 when 1 then @ol_qty2
                 when 2 then @ol_qty3
                 when 3 then @ol_qty4
                 when 4 then @ol_qty5
                 when 5 then @ol_qty6
                 when 6 then @ol_qty7
                 when 7 then @ol_qty8
                 when 8 then @ol_qty9
                 when 9 then @ol_qty10
                 when 10 then @ol_qty11
                 when 11 then @ol_qty12
                 when 12 then @ol_qty13
                 when 13 then @ol_qty14
                 when 14 then @ol_qty15
```

```
            else @ol_qty
            end

        select @i_price = i_price,
               @i_name = i_name,
               @i_data = i_data
            from item HOLDLOCK
            where i_id = @i_id

        if (@@rowcount = 0)
            begin
              select @commit_flag = 0
              select NULL, NULL, NULL, NULL, NULL
              continue
            end
            update stock
                set s_ytd = s_ytd + @ol_qty,
                    @ol_amount = @ol_qty * @i_price,
                    @s_quantity = s_quantity - @ol_qty +
                          case when (s_quantity - @ol_qty < 10)
                          then 91 else 0 end,
                     s_quantity = s_quantity - @ol_qty +
                          case when (s_quantity - @ol_qty < 10)
                          then 91 else 0 end,
                     s_order_cnt = s_order_cnt + 1,
                     @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_w_id = @w_id and
                           s_i_id = @i_id


    INSERT INTO order_line (
         ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
         ol_supply_w_id, ol_delivery_d, ol_quantity,
         ol_amount, ol_dist_info)
    VALUES (
         @o_id, @d_id, @w_id, @ol_number, @i_id,
         @w_id, “19000101”, @ol_qty,
         @ol_amount, @s_dist)


    select
       @i_name,
       @i_price,
       @s_quantity,
       @ol_amount,
       b_g= case when((patindex(“%ORIGINAL%”, @i_data) > 0) and
                  (patindex(“%ORIGINAL%”, @s_data) > 0))
             then “B” else “G” end

  end
```

```
        SELECT @c_last = c_last,
                 @c_discount = c_discount,
                 @c_credit   = c_credit,
                 @c_ins_id= c_id
        FROM customer (index c_clu prefetch 2 lru)  HOLDLOCK
        WHEREc_w_id= @w_id
          ANDc_d_id= @d_id
          ANDc_id = @c_id

        INSERT INTO orders (
                o_id, o_c_id, o_d_id, o_w_id,
                o_entry_d, o_carrier_id, o_ol_cnt, o_all_local)
        VALUES (
                @o_id, @c_ins_id, @d_id, @w_id,
                @o_entry_d, -1, @o_ol_cnt, 1)
        INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
        VALUES (@o_id, @d_id, @w_id)


        SELECT  @w_tax = w_tax
        FROM warehouse HOLDLOCK
        WHERE w_id = @w_id



        if (@commit_flag = 1)
                commit transaction NO
        else
                rollback transaction NO

        select
                @w_tax, @d_tax, @o_id, @c_last,
                @c_discount, @c_credit, @o_entry_d
end
go

if exists ( SELECT name FROM sysobjects WHERE name =
‘neworder_remote’)
        DROP PROC neworder_remote
go
CREATE PROC neworder_remote (
        @w_id        smallint,
        @d_id        tinyint,
        @c_id        int,
        @o_ol_cnttinyint,

        @i_id int = 0, @s_w_idsmallint = 0, @ol_qtytinyint = 0,
        @i_id2int = 0, @s_w_id2smallint = 0, @ol_qty2tinyint = 0,
        @i_id3int = 0, @s_w_id3smallint = 0, @ol_qty3tinyint = 0,
        @i_id4int = 0, @s_w_id4smallint = 0, @ol_qty4tinyint = 0,
        @i_id5int = 0, @s_w_id5smallint = 0, @ol_qty5tinyint = 0,
        @i_id6int = 0, @s_w_id6smallint = 0, @ol_qty6tinyint = 0,
        @i_id7int = 0, @s_w_id7smallint = 0, @ol_qty7tinyint = 0,
        @i_id8int = 0, @s_w_id8smallint = 0, @ol_qty8tinyint = 0,
        @i_id9int = 0, @s_w_id9smallint = 0, @ol_qty9tinyint = 0,
        @i_id10int = 0, @s_w_id10smallint = 0, @ol_qty10tinyint = 0,
        @i_id11int = 0, @s_w_id11smallint = 0, @ol_qty11tinyint = 0,
        @i_id12int = 0, @s_w_id12smallint = 0, @ol_qty12tinyint = 0,
        @i_id13int = 0, @s_w_id13smallint = 0, @ol_qty13tinyint = 0,
        @i_id14int = 0, @s_w_id14smallint = 0, @ol_qty14tinyint = 0,
        @i_id15int = 0, @s_w_id15smallint = 0, @ol_qty15tinyint = 0
)
as
```

```
declare
        @w_tax      real,           @d_tax      real,
        @c_last     char(16),@c_creditchar(2),
        @c_discountreal, @commit_flagtinyint,
        @c_ins_id int,

        @i_pricereal,
        @i_name     char(24),@i_data char(50),

        @s_quantitysmallint,
        @s_ytd      int,            @s_order_cntint,
        @s_dist     char(24),@s_datachar(50),
        @s_remote_cntint,@remote    int,

        @ol_numbertinyint,@o_idint,
        @o_entry_ddatetime,@b_gchar(1),
        @ol_amount real


begin

        begin transaction NO

        -- @#@# UPDATE district FROM district, warehouse, customer
        --

        UPDATE district
        SET     d_next_o_id = d_next_o_id + 1
                , @o_id     = d_next_o_id
                , @d_tax = d_tax
                , @commit_flag= 1
                , @ol_number= 0
                , @o_entry_d= getdate()
        WHERE   d_w_id= @w_id
                AND d_id  = @d_id

        while (@ol_number < @o_ol_cnt) begin
                SELECT @ol_number = @ol_number + 1
                ,@i_id = case @ol_number
         when 1 then @i_id2
                 when 2 then @i_id3
                 when 3 then @i_id4
                 when 4 then @i_id5
                 when 5 then @i_id6
                 when 6 then @i_id7
                 when 7 then @i_id8
                 when 8 then @i_id9
                 when 9 then @i_id10
                 when 10 then @i_id11
                 when 11 then @i_id12
                 when 12 then @i_id13
                 when 13 then @i_id14
                 when 14 then @i_id15
                 else @i_id
                 end
                , @ol_qty = case @ol_number
                 when 1 then @ol_qty2
                 when 2 then @ol_qty3
                 when 3 then @ol_qty4
                 when 4 then @ol_qty5
                 when 5 then @ol_qty6
                 when 6 then @ol_qty7
```

```
        when 7 then @ol_qty8
        when 8 then @ol_qty9
        when 9 then @ol_qty10
        when 10 then @ol_qty11
        when 11 then @ol_qty12
        when 12 then @ol_qty13
        when 13 then @ol_qty14
        when 14 then @ol_qty15
        else @ol_qty
        end
       ,@s_w_id = case @ol_number
when 1 then @s_w_id2
        when 2 then @s_w_id3
        when 3 then @s_w_id4
        when 4 then @s_w_id5
        when 5 then @s_w_id6
        when 6 then @s_w_id7
        when 7 then @s_w_id8
        when 8 then @s_w_id9
        when 9 then @s_w_id10
        when 10 then @s_w_id11
        when 11 then @s_w_id12
        when 12 then @s_w_id13
        when 13 then @s_w_id14
        when 14 then @s_w_id15
        else @s_w_id
        end

        select @i_price = i_price,
            @i_name = i_name ,
            @i_data = i_data
          from item HOLDLOCK
          where i_id = @i_id

       if (@@rowcount = 0)
         begin
             select @commit_flag = 0
             select NULL, NULL, NULL, NULL, NULL
              continue
         end
    update stock
     set s_ytd = s_ytd + @ol_qty,
         @ol_amount = @ol_qty * @i_price,
         @s_quantity = s_quantity - @ol_qty +
                case when (s_quantity - @ol_qty < 10)
                then 91 else 0 end,
         s_quantity = s_quantity - @ol_qty +
         case when (s_quantity - @ol_qty < 10)
           then 91 else 0 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,
         s_order_cnt = s_order_cnt + 1,
                s_remote_cnt = s_remote_cnt +
                case when (@s_w_id = @w_id)
                    then 0 else 1 end
            where s_w_id = @w_id and
                s_i_id = @i_id


        INSERT INTO order_line (
            ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
            ol_supply_w_id, ol_delivery_d, ol_quantity,
            ol_amount, ol_dist_info)
        VALUES (
            @o_id, @d_id, @w_id, @ol_number, @i_id,
            @w_id, "19000101", @ol_qty,
            @ol_amount, @s_dist)

            select
            @i_name,
            @i_price,
            @s_quantity,
            @ol_amount,
            b_g = case when ((patindex("%ORIGINAL%", @i_data) > 0) and
                        (patindex("%ORIGINAL%", @s_data) > 0))
                  then "B" else "G" end
        end


        SELECT @c_last = c_last,
              @c_discount = c_discount,
              @c_credit = c_credit,
              @c_ins_id= c_id
        FROM customer (index c_clu prefetch 2 lru)  HOLDLOCK
        WHEREc_w_id= @w_id
          ANDc_d_id= @d_id
          ANDc_id = @c_id

        INSERT INTO orders (
                o_id, o_c_id, o_d_id, o_w_id,
                o_entry_d, o_carrier_id, o_ol_cnt, o_all_local)
        VALUES (
                @o_id, @c_ins_id, @d_id, @w_id,
                @o_entry_d, -1, @o_ol_cnt, 0)
        INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
        VALUES (@o_id, @d_id, @w_id)

        SELECT  @w_tax = w_tax
        FROM warehouse HOLDLOCK
        WHERE w_id = @w_id

        if (@commit_flag = 1)
                commit transaction NO
        else
                rollback transaction NO

        select
                @w_tax, @d_tax, @o_id, @c_last,
                @c_discount, @c_credit, @o_entry_d
end
go
if exists (select * from sysobjects where name = 'payment_byid')
        DROP PROC payment_byid
go
CREATE PROC payment_byid
        @w_id       smallint,@c_w_idsmallint,
        @h_amount float,
        @d_id       tinyint,@c_d_id  tinyint,
        @c_idint
as
declare@c_last    char(16)

declare@w_street_1char(20),@w_street_2char(20),
        @w_city     char(20),@w_statechar(2),
        @w_zip      char(9),@w_namechar(10),
        @w_ytd      float

declare@d_street_1char(20),@d_street_2char(20),
        @d_city     char(20),@d_statechar(2),
        @d_zip      char(9),@d_namechar(10),
        @d_ytd      float

declare@c_firstchar(16),@c_middlechar(2),
        @c_street_1char(20),@c_street_2char(20),
        @c_city     char(20),@c_statechar(2),
        @c_zip      char(9),@c_phonechar(16),
        @c_sincedatetime,@c_creditchar(2),
        @c_credit_limnumeric(12,0),@c_balancefloat,
        @c_discountreal,
        @data1      char(250),@data2char(250),
        @c_data_1char(250),@c_data_2char(250)

declare @screen_datachar(200),@today datetime


BEGIN TRANSACTION PID

        UPDATE district
                SET d_ytd = d_ytd + @h_amount
                   ,@d_ytd = d_ytd
                   ,@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
                WHERE d_w_id= @w_id
                AND d_id  = @d_id

        UPDATE warehouse
                SET w_ytd = w_ytd + @h_amount
                   ,@w_ytd = w_ytd
                   ,@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
                WHERE w_id = @w_id


        SELECT  @screen_data = NULL
        UPDATE customer SET
                  @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_balance = c_balance - @h_amount
            , @c_balance = c_balance - @h_amount
            , c_ytd_payment = c_ytd_payment + @h_amount
            , c_payment_cnt = c_payment_cnt + 1
            , @c_since = c_since
            , @data1 = c_data1
            , @data2 = c_data2
            , @today = getdate()
        where
            c_id = @c_id
            and c_w_id = @c_w_id
            and c_d_id = @c_d_id

        if (@c_credit = "BC")
        begin
            SELECT  @c_data_2 =
                substring(@data1, 209, 42) +
                substring(@data2, 1, 208)
                ,@c_data_1 =
                convert(char(5), @c_id) +
                convert(char(4), @c_d_id) +
                convert(char(5), @c_w_id) +
                convert(char(4), @d_id) +
                convert(char(5), @w_id) +
                convert(char(19), @h_amount/100) +
    substring(@data1, 1, 208)

            UPDATE customer SET
                  c_data1 = @c_data_1
                , c_data2 = @c_data_2
                , @screen_data = substring(@c_data_1, 1, 200)
            WHERE
                c_id = @c_id
                AND c_w_id = @c_w_id
                AND c_d_id = @c_d_id
        end

        INSERT INTO history (
            h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
            h_date, h_amount, h_data)
        VALUES (
            @c_id, @c_d_id, @c_w_id, @d_id, @w_id,
            @today, @h_amount, (@w_name + "    " + @d_name))

    COMMIT TRANSACTION PID

        select
            @c_id,
            @c_last,
            @today,
            @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
if exists (select * from sysobjects where name = 'payment_byname')
    DROP PROC payment_byname
go
CREATE PROC payment_byname
    @w_id       smallint,@c_w_idsmallint,
    @h_amount float,
    @d_id       tinyint,@c_d_id  tinyint,
    @c_lastchar(16)
as
declare@n int,  @c_idint

declare@w_street_1char(20),@w_street_2char(20),
    @w_city     char(20),@w_statechar(2),
    @w_zip      char(9),@w_namechar(10),
    @w_ytd      float

declare@d_street_1char(20),@d_street_2char(20),
    @d_city     char(20),@d_statechar(2),
    @d_zip      char(9),@d_namechar(10),
    @d_ytd      float

declare@c_firstchar(16),@c_middlechar(2),
    @c_street_1char(20),@c_street_2char(20),
    @c_city     char(20),@c_statechar(2),
    @c_zip      char(9),@c_phonechar(16),
    @c_sincedatetime,@c_creditchar(2),
    @c_credit_limnumeric(12,0),@c_balancefloat,
    @c_discountreal,
    @data1      char(250),@data2char(250),
    @c_data_1char(250),@c_data_2char(250)

declare @screen_datachar(200),@today datetime



BEGIN TRANSACTION PNM
    SELECT @n = (count(*)+1)/2
    FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
    WHEREc_w_id = @c_w_id and
        c_d_id = @c_d_id and
        c_last = @c_last
```

```
set rowcount @n

-- @#@ SELECT FROM customer HOLDLOCK
SELECT @c_id = c_id
FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
WHEREc_w_id = @c_w_id and
        c_d_id = @c_d_id and
        c_last = @c_last

-- Reset, so as to do full retrievals hereafter.
set rowcount 0

UPDATE district
        SET d_ytd = d_ytd + @h_amount
           ,@d_ytd = d_ytd
           ,@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
        WHERE d_w_id= @w_id
        AND d_id  = @d_id

UPDATE warehouse
        SET w_ytd = w_ytd + @h_amount
           ,@w_ytd = w_ytd
           ,@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
        WHERE w_id = @w_id

UPDATE customer SET
          @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_balance = c_balance - @h_amount
        , @c_balance = c_balance - @h_amount
        , c_ytd_payment = c_ytd_payment + @h_amount
        , c_payment_cnt = c_payment_cnt + 1
        , @c_since = c_since
        , @data1 = c_data1
        , @data2 = c_data2
        , @today = getdate()
where
        c_id = @c_id
        and c_w_id = @c_w_id
        and c_d_id = @c_d_id

SELECT@screen_data = NULL
if (@c_credit = “BC”)
```

```
        begin
                SELECT  @c_data_2 =
                        substring(@data1, 209, 42) +
                        substring(@data2, 1, 208)
                        ,@c_data_1 =
                        convert(char(5), @c_id) +
                        convert(char(4), @c_d_id) +
                        convert(char(5), @c_w_id) +
                        convert(char(4), @d_id) +
                        convert(char(5), @w_id) +
                        convert(char(19), @h_amount/100) + substring(@data1, 1,
208)

                UPDATE customer SET
                          c_data1 = @c_data_1
                        , c_data2 = @c_data_2
                        , @screen_data = substring(@c_data_1, 1, 200)
                WHERE
                        c_id = @c_id
                        AND c_w_id = @c_w_id
                        AND c_d_id = @c_d_id
        end

        INSERT INTO history (
                h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                h_date, h_amount, h_data)
        VALUES (
                @c_id, @c_d_id, @c_w_id, @d_id, @w_id,
                @today, @h_amount, (@w_name + “     “ + @d_name))

COMMIT TRANSACTION PNM

        select
                @c_id,
                @c_last,
                @today,
                @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
```

```
if exists (select * from sysobjects where name = ‘order_status_byid’)
        DROP PROC order_status_byid
go
CREATE PROC order_status_byid
        @w_id        smallint,
        @d_id        tinyint,
        @c_id        int
as

DECLARE@o_id int,
        @o_entry_ddatetime,
        @o_carrier_idsmallint

BEGIN TRANSACTION OSID

        set rowcount 1
        SELECT@o_id = o_id, @o_carrier_id = o_carrier_id,
                @o_entry_d = o_entry_d
        FROMorders (index o_clu prefetch 16 mru) HOLDLOCK
        WHEREo_w_id= @w_id
        AND o_d_id= @d_id
        AND o_c_id= @c_id
        ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
        set rowcount 0

        select
                ol_supply_w_id,
                ol_i_id,
                ol_quantity,
                ol_amount,
                ol_delivery_d
        FROMorder_line HOLDLOCK
        WHEREol_o_id = @o_id
        AND ol_d_id = @d_id
        AND ol_w_id = @w_id

        select
                @c_id, c_last, c_first, c_middle, c_balance,
                @o_id,
                @o_entry_d,
                @o_carrier_id
        FROMcustomer (index c_clu prefetch 2 lru) HOLDLOCK
        WHERE   c_id    = @c_id
        AND c_d_id  = @d_id
        AND c_w_id  = @w_id

COMMIT TRANSACTION OSID
go
if exists (select * from sysobjects where name = ‘order_status_byname’)
        DROP PROC order_status_byname
go
CREATE PROC order_status_byname
        @w_id        smallint,
        @d_id        tinyint,
        @c_lastchar(16)
as

DECLARE@o_id int,
        @o_entry_ddatetime,
        @o_carrier_idsmallint

declare@n int,    @c_idint
```

```
BEGIN TRANSACTION OSNM
        SELECT @n = (count(*)+1)/2
        FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
        WHEREc_w_id = @w_id and
                c_d_id = @d_id and
                c_last = @c_last


        -- Retrieve upto mid-point number of rows.
        set rowcount @n

        -- @#@# SELECT FROM customer HOLDLOCK
        SELECT @c_id = c_id
        FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
        WHEREc_w_id = @w_id and
                c_d_id = @d_id and
                c_last = @c_last


        set rowcount 1
        SELECT@o_id = o_id, @o_carrier_id = o_carrier_id,
                @o_entry_d = o_entry_d
        FROMorders (index o_clu prefetch 16 mru) HOLDLOCK
        WHEREo_w_id= @w_id
        AND o_d_id= @d_id
        AND o_c_id= @c_id
        ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
        set rowcount 0

        select
                ol_supply_w_id,
                ol_i_id,
                ol_quantity,
                ol_amount,
                ol_delivery_d
        FROMorder_line HOLDLOCK
        WHEREol_o_id = @o_id
        AND ol_d_id = @d_id
        AND ol_w_id = @w_id

        select
                @c_id, c_last, c_first, c_middle, c_balance,
                @o_id,
                @o_entry_d,
                @o_carrier_id
        FROMcustomer (index c_clu prefetch 2 lru) HOLDLOCK
        WHERE   c_id    = @c_id
        AND c_d_id  = @d_id
        AND c_w_id  = @w_id

COMMIT TRANSACTION OSNM
go

if exists (select * from sysobjects where name = ‘delivery’)
        drop proc delivery
go

CREATE PROC delivery
        @w_id            smallint,
        @o_carrier_id    smallint,
        @d_id            tinyint = 1
as
```

```
declare @no_o_id        int,            @o_c_id         smallint,
        @ol_total       float,          @ol_amount      float,
        @junk_id        smallint,
        @today          datetime

declare c_del_no CURSOR FOR
        SELECT   no_o_id
        FROM    new_order (index no_clu) HOLDLOCK
        WHERE   no_d_id = @d_id
        AND      no_w_id = @w_id
        FOR UPDATE

begin

    while (@d_id <= 10) begin
    BEGIN TRANSACTION DEL

        OPEN c_del_no

        FETCH c_del_no INTO @no_o_id

        if (@@sqlstatus != 0)
        begin
                COMMIT TRANSACTION DEL
                select NULL
                CLOSE c_del_no
        end
        else
        begin
                DELETE  FROM new_order
                WHERE   CURRENT OF c_del_no
                CLOSE   c_del_no

                SELECT  @ol_total = 0.0, @today = getdate()

                -- @#@# UPDATE order_line
                UPDATE order_line
                SET       ol_delivery_d = @today
                        , @ol_total = @ol_total + ol_amount
                WHERE   ol_o_id = @no_o_id
                AND      ol_d_id = @d_id
                AND ol_w_id = @w_id

                -- @#@# UPDATE orders
                UPDATE  orders
                SET       o_carrier_id = @o_carrier_id
                        , @o_c_id = o_c_id
                WHERE   o_id    = @no_o_id
                AND      o_d_id  = @d_id
                AND      o_w_id  = @w_id

                UPDATE  customer
                SET     c_balance       = c_balance + @ol_total,
                        c_delivery_cnt  = c_delivery_cnt + 1
                WHERE   c_id    = @o_c_id
                AND      c_d_id  = @d_id
                AND      c_w_id  = @w_id

                COMMIT TRANSACTION DEL

                select
                        @no_o_id
        end
```

```
        select @d_id = @d_id + 1
    end
end
go


if exists ( SELECT name FROM sysobjects WHERE name = ‘stock_level’)
    DROP PROC stock_level
go

CREATE PROC stock_level
        @w_idsmallint,
        @d_idtinyint,
        @threshold smallint
as
        select  s_i_id
        FROMdistrict,
                order_line (index ol_clu prefetch 2 lru),
                stock (index s_clu prefetch 2 lru)
        WHEREd_w_id=@w_id
        AND d_id   =       @d_id
        AND ol_w_id=    @w_id
        AND ol_d_id=    @d_id
        AND ol_o_idbetween (d_next_o_id - 20) and (d_next_o_id - 1)
        AND s_w_id=     ol_w_id
        AND s_i_id=     ol_i_id
        AND      s_quantity < @threshold
go
EOF

/*
 * Copyright (c) 1995, 1996, 1997, 1998, 1999 by Sun Microsystems, Inc.
 */

#ifndef SYB_TPCC_H
#define SYB_TPCC_H

#defineMAXDIST 10
#define MaxTries 5
#define smaller(x,y) (x<y ? x : y)
#define XCTION_COUNT 8

int     err_handler();
int     msg_handler();

#define XACT_NEWO0
#define XACT_PAYM_ID1
#define XACT_PAYM_NAME2
#define XACT_ORDS_ID3
#define XACT_ORDS_NAME4
#define XACT_DEL5
#define XACT_STOCK6
#define XACT_BKEND7

typedef struct Order_Line {
        int     i_id;
        DBSMALLINT      supply_w_id;
        DBSMALLINT  quantity;
        DBSMALLINT  s_quantity;
        DBFLT8  i_price;
        DBFLT8  ol_amount;
        char    i_name[26];
        int             increment;
} ORDER_LINE;

void gen_new_order();voidnew_order_rpc();
void gen_payment_byid();voidpayment_byid_rpc();
void gen_payment_byname();voidpayment_byname_rpc();
void gen_order_status_byid(); voidorder_status_byid_rpc();
void gen_order_status_byname();voidorder_status_byname_rpc();
void gen_delivery();   void delivery_qu_add();
void gen_stock_level();voidstock_level_rpc();
void delivery_qu_del();voiddelivery_rpc();
void delivery_qu_connect();
void display_xction();
void sleep_before_retry ();
void display_xction();
void pick_xact_type();

typedef struct Xction {
        char  name[30];
} XCTION;

extern XCTION func_array[XCTION_COUNT+1];

extern int  rollback_pct;
extern int  lines_per_call;
extern charb_g[2];
extern DBFLT8total_amount;
extern DBTINYINTcommit_flag;
extern int  xact_type, prev_xact_type;
extern int  deadlock;
extern int  bad_items;
extern int  max_ware;
extern RETCODEcode;


extern DBSMALLINTglobal_w_id;
extern DBTINYINTglobal_d_id;
extern ORDER_LINEol[15];

extern DBINTc_id;
extern DBTINYINTc_d_id;
extern DBSMALLINTc_w_id;
extern charc_first[17];
extern charc_middle[3];
extern charc_last[17];
extern charc_street_1[21];
extern charc_street_2[21];
extern charc_city[21];
extern charc_state[3];
extern charc_zip[10];
extern charc_phone[17];
extern charc_since[31];
extern charc_credit[3];
extern DBFLT8c_credit_lim;
extern DBREALc_discount;
extern DBFLT8c_balance;
extern charc_data[201];

extern charw_name[11];
extern charw_street_1[21];
extern charw_street_2[21];
extern charw_city[21];
extern charw_state[3];
extern charw_zip[10];
```

```
extern DBREALw_tax;

extern DBTINYINTglobal_d_id;
extern DBSMALLINTd_w_id;
extern char    d_name[11];
extern char    d_street_1[21];
extern char    d_street_2[21];
extern char    d_city[21];
extern char    d_state[3];
extern char    d_zip[10];
extern DBREAL  d_tax;

extern int  i_id;
extern DBFLT8i_price;
extern chari_name[25];

extern DBSMALLINTs_quantity;
extern DBSMALLINTthreshold;
extern DBINTlow_count;
extern chars_dist[25];

extern int  o_id;
extern DBTINYINTo_d_id;
extern DBSMALLINTo_w_id;
extern DBSMALLINTo_c_id;
extern charo_entry_d[31];
extern DBSMALLINTo_carrier_id;
extern DBSMALLINTo_ol_cnt, o_ol_now, o_ol_done;
extern DBTINYINTo_all_local;

extern int  ol_o_id;
extern DBTINYINTol_d_id;
extern DBSMALLINTol_w_id;
extern DBSMALLINTol_number;
extern DBINTol_i_id;
extern DBSMALLINTol_supply_w_id;
extern charol_delivery_d[31];
extern DBSMALLINTol_quantity;
extern DBFLT8ol_amount;

extern int  no_o_id;
extern DBTINYINTno_d_id;
extern DBSMALLINTno_w_id;

extern DBFLT8h_amount;
extern charh_date[20];

#endif SYB_TPCC_H
/*
 * Copyright (c) 1995, 1996, 1997, 1998, 1999 by Sun Microsystems, Inc.
 */

#ifndef SYB_DRIVER_H
#define SYB_DRIVER_H

#define SERVER  NULL
#define DATABASE"tpcc"
#define USER"sa"
#define MAX_ERROR       1

#defineDBNAME1
#define FUNC_NAME2
#defineNUSERS 3
```

```
#defineRAMP_UP4
#defineSTDYSTATE5
#defineRAMP_DOWN6
#define MAX_WAREHOUSE7
#defineROLLBACK_PCT8
#defineDELTA    9

#defineINTERVAL25
#defineUNIT      .5
#defineHIST_MAX50
#define BUCKET500
#defineMATCH  0
#define MILLI1000

typedef struct  CNTRL
{
        int     tran_count;
        int     deadlock_cnt;
        int     res_time;
        doubletot_time;
        int     min_res;
        int     max_res;
        int     not_done;
        double  tran_sqr;
        int     tran_2sec;
} CONTROL;

extern LOGINREC*login;
extern DBPROCESS*dbproc;

extern intscale, nusers;
extern DBINTrun_id;
extern char* db_name;
extern intrampup, stdystate, rampdown;
extern intend_rampup, end_stdystate, end_rampdown;
extern charfunc_name[32];
extern CONTROLstatus[11];
extern unsignedlong avg_delay;
extern unsignedlong last_resp;
extern intdelta;

doubledrand();
void sel1();
void     init_time();
unsigned long delay();

#endif SYB_DRIVER_H
/*
** Confidential property of Sybase, Inc.
** (c) Copyright Sybase, Inc. 1999
** All rights reserved
*/
#include <stdio.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include "SYB_tpcc.h"

#defineCONTEXT_SET5701
#defineLANGUAGE_SET5703
#defineCHARACTER_SET5704
#defineABORT_ERROR6104
```

```
#include "atmi.h"
#include "userlog.h"

int
err_handler(dbproc, severity, errno, oserr)
        DBPROCESS *dbproc;
        int severity;
        int errno;
        int oserr;
{
        userlog("DB-LIBRARY Error %d:", errno);
        display_xction(dberrstr(errno));

        if (oserr != DBNOERR)
                {
                userlog("O/S Error: ");
                display_xction(dboserrstr(oserr));
                }

        exit(-100);
}



int
msg_handler(dbproc,msgno,msgstate,severity,msgtext,servername,procname,
line)
        DBPROCESS*dbproc;
        int             msgno;
        int             msgstate;
        int             severity;
        char            *msgtext;
        char             *servername;
        char             *procname;
        int             line;
{
        if (msgno == CONTEXT_SET ||
            msgno == LANGUAGE_SET ||
            msgno == CHARACTER_SET)
                return(SUCCEED);

        if (msgno == ABORT_ERROR)
                return(SUCCEED);

        if (msgno == 1205)
        {
                display_xction(msgtext);
                deadlock = 1;
                return(SUCCEED);
        }
        else {
                userlog("msg no %d - %s\n", msgno, msgtext);
                userlog("xact_type: %d deadlock= %d\n", xact_type, deadlock);
                if (msgno == 0)
                        return(SUCCEED);
                else
                        return(FAIL);
        }
}
/*
 * Copyright (c)  1995, 1996, 1997, 1998, 1999  by Sun Microsystems, Inc.
 */
```

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <sybfront.h>
#include <sybdb.h>

#include “SYB_tpcc.h”
#include “SYB_driver.h”
#include “SYB_rpc_var.c”

#include “atmi.h”
#include “userlog.h”
#include “fml.h”
#include “mods.h”
#include “Usysflds.h”

#include “tpcc_tux_forms.h”
#include “tpcc_tux_forms_var.c”

DBDATETIME      syb_datetime;
DBDATETIME      syb_date;

int invalid_xact;

void sybdate2datetime(DBDATETIME * sybdate, char * datetime)
{
    DBDATEREC    daterec;

    dbdatecrack(NULL, &daterec, sybdate);

    sprintf(datetime, “%02d-%02d-%04d %02d:%02d:%02d”,
        daterec.datedmonth,
        daterec.datemonth+1,
        daterec.dateyear,
        daterec.datehour,
        daterec.dateminute,
        daterec.datesecond);
}

void sybdate2date(DBDATETIME * sybdate, char * date)
{
    DBDATEREC    daterec;

    dbdatecrack(NULL, &daterec, sybdate);

    sprintf(date, “%02d-%02d-%04d”,
        daterec.datedmonth,
        daterec.datemonth+1,
        daterec.dateyear);
}

void
new_order_rpc()
{
    int         try;

    for (try=0; try<MaxTries; try++)
    {
        if (try > 0) display_xction(“Repeating NO”);

        deadlock = 0;
        if (new_order_body() != TRUE) break;;
        dbcancel(dbproc);
        sleep_before_retry();
    }
    if (try >= MaxTries) display_xction(“Failed”);
}


int
new_order_body()
{
    int i,j;
    DBINT retcode;
    struct items_inf *cur_ip;

    deadlock = 0;
    if (o_all_local)
        dbrpcinit(dbproc, “neworder_local”, 0);
    else
        dbrpcinit(dbproc, “neworder_remote”, 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, &global_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, &c_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &o_ol_cnt);

    for(i = o_ol_done; i < (int)o_ol_cnt; i++)
    {
        dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, &ol[i].i_id);
        if (!o_all_local)
                dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &ol[i].supply_w_id);
                dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &ol[i].quantity);
    }

    if (dbrpcsend(dbproc) != SUCCEED) return TRUE;
    if (dbsqlok(dbproc)   != SUCCEED) return TRUE;

    for (i = 0; i < o_ol_cnt; i++)
    {
        if (dbresults(dbproc) != SUCCEED || deadlock)
            return TRUE;
        else
        {
            dbbind(dbproc, 1, NTBSTRINGBIND,  sizeof(i_name), i_name);
            dbbind(dbproc, 2, FLT8BIND,       0, &i_price);
            dbbind(dbproc, 3, SMALLBIND,      0, &s_quantity);
            dbbind(dbproc, 4, FLT8BIND,       0, &ol_amount);
            dbbind(dbproc, 5, NTBSTRINGBIND,  sizeof(b_g), b_g);
            if (dbnextrow(dbproc) != REG_ROW) return TRUE;


            if(*i_name == ‘\0’)
            {

                strcpy(neworder->status, “Item number is not valid”);
                commit_flag = FALSE;
            }
            if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
        }
        if (dbhasretstat(dbproc))
        {
            if ((retcode = dbretstatus(dbproc)) == -3)
            {
                deadlock = 1;
                display_xction(“Deadlock victim:”);
            }
            else if (retcode<0)
            {
                userlog(“Unknown return status %d:”, retcode);
                display_xction(““);
                strcpy(neworder->status, "SQL error");
            }
            return TRUE;
        }

        cur_ip = &neworder->n_items[i];
        strcpy(cur_ip->i_name, i_name);
        cur_ip->i_price = i_price;
        cur_ip->s_quantity = s_quantity;
        strcpy(cur_ip->brand, b_g);
        cur_ip->ol_amount = ol_amount;

        total_amount += ol_amount;
    }


    if (dbresults(dbproc) != SUCCEED | | deadlock)
        {
        return TRUE;
    }

    dbbind(dbproc, 1, REALBIND, 0,&w_tax);
    dbbind(dbproc, 2, REALBIND, 0, &d_tax);
    dbbind(dbproc, 3, INTBIND,  0, &o_id);
    dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(c_last),c_last);
    dbbind(dbproc, 5, REALBIND,      0,&c_discount);
    dbbind(dbproc, 6, NTBSTRINGBIND, sizeof(c_credit),c_credit);
    dbbind(dbproc, 7, DATETIMEBIND,0,&syb_datetime);
    if (dbnextrow(dbproc) != REG_ROW | | deadlock) return TRUE;
    if (dbcanquery(dbproc) != SUCCEED | | deadlock) return TRUE;
    sybdate2datetime(&syb_datetime, o_entry_d);

    neworder->w_tax = w_tax*100;
    neworder->d_tax = d_tax*100;
    neworder->o_id = o_id;
    strcpy(neworder->c_last, c_last);
    neworder->c_discount = c_discount * 100;
    strcpy(neworder->c_credit, c_credit);
    strcpy(neworder->o_entry_d, o_entry_d);

    if (dbretstatus(dbproc) == -6)
    {
        fprintf(stderr, “This is an invalid transaction\n”);
        invalid_xact++;
    }



    return FALSE;
}

void
payment_byid_rpc()
{
    int try;

    for (try=0; try<MaxTries; try++)
    {
        if (try>0) display_xction(“Repeating”);

        if (payment_byid_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (payment_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }

    if (try >= MaxTries)
    {
            display_xction(“MaxTries Failed”);
            tpreturn(TPFAIL, 0, (char *)paym_rqst->data, paym_rqst->len, 0);
    }
}

int
payment_byid_begin()
{
    deadlock = 0;
    dbrpcinit(dbproc, “payment_byid”, 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &c_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &h_amount);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &global_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &c_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &c_id);
    return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

void
payment_byname_rpc()
{
    int try;

    for (try=0; try<MaxTries; try++)
    {
        if (try>0) display_xction(“Repeating”);

        if (payment_byname_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (payment_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }

    if (try >= MaxTries)
    {
            display_xction("MaxTries Failed");
            tpreturn(TPFAIL, 0, (char *)paym_rqst->data, paym_rqst->len,
0);
    }
}

int
payment_byname_begin()
{
    deadlock = 0;
    dbrpcinit(dbproc, "payment_byname", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &c_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &h_amount);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &global_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &c_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(c_last), c_last);
    return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

int
payment_end()
{
    if (dbsqlok(dbproc) != SUCCEED)
            return TRUE;
    if (dbresults(dbproc) != SUCCEED || deadlock)
            return TRUE;
    else
    {


        dbbind(dbproc,  1, INTBIND,     0, &c_id);
        dbbind(dbproc,  2, NTBSTRINGBIND, sizeof(c_last), c_last);
        dbbind(dbproc,  3, DATETIMEBIND, 0, &syb_datetime);
        dbbind(dbproc,  4, NTBSTRINGBIND, sizeof(w_street_1),
w_street_1);
        dbbind(dbproc,  5, NTBSTRINGBIND, sizeof(w_street_2),
w_street_2);
        dbbind(dbproc,  6, NTBSTRINGBIND, sizeof(w_city), w_city);
        dbbind(dbproc,  7, NTBSTRINGBIND, sizeof(w_state), w_state);
        dbbind(dbproc,  8, NTBSTRINGBIND, sizeof(w_zip), w_zip);

        dbbind(dbproc,  9, NTBSTRINGBIND, sizeof(d_street_1),
d_street_1);
        dbbind(dbproc, 10, NTBSTRINGBIND, sizeof(d_street_2),
d_street_2);
        dbbind(dbproc, 11, NTBSTRINGBIND, sizeof(d_city), d_city);
        dbbind(dbproc, 12, NTBSTRINGBIND, sizeof(d_state), d_state);
        dbbind(dbproc, 13, NTBSTRINGBIND, sizeof(d_zip), d_zip);

        dbbind(dbproc, 14, NTBSTRINGBIND, sizeof(c_first), c_first);
        dbbind(dbproc, 15, NTBSTRINGBIND, sizeof(c_middle),
c_middle);
        dbbind(dbproc, 16, NTBSTRINGBIND, sizeof(c_street_1),
c_street_1);
        dbbind(dbproc, 17, NTBSTRINGBIND, sizeof(c_street_2),
c_street_2);
        dbbind(dbproc, 18, NTBSTRINGBIND, sizeof(c_city), c_city);
        dbbind(dbproc, 19, NTBSTRINGBIND, sizeof(c_state), c_state);
        dbbind(dbproc, 20, NTBSTRINGBIND, sizeof(c_zip), c_zip);
        dbbind(dbproc, 21, NTBSTRINGBIND, sizeof(c_phone), c_phone);
        dbbind(dbproc, 22, DATETIMEBIND, 0, &syb_date);
        dbbind(dbproc, 23, NTBSTRINGBIND, sizeof(c_credit), c_credit);
        dbbind(dbproc, 24, FLT8BIND,      0, &c_credit_lim);
        dbbind(dbproc, 25, REALBIND,      0, &c_discount);
        dbbind(dbproc, 26, FLT8BIND,      0, &c_balance);
        dbbind(dbproc, 27, NTBSTRINGBIND, sizeof(c_data), c_data);
        if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
        if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
        sybdate2datetime(&syb_datetime, h_date);
        sybdate2date(&syb_date, c_since);

    }

    if (dbretstatus(dbproc) == -6)
    {
        fprintf(stderr, "This is an invalid transaction\n");
        invalid_xact++;
    }


    return FALSE;
}

void
order_status_byid_rpc()
{
    int try;


    for (try=0; try<MaxTries; try++)
    {
        if (try>0) display_xction("Repeating");

        if (order_status_byid_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (order_status_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }

    if (try >= MaxTries)
    {
        display_xction("MaxTries Failed");
        tpreturn(TPFAIL, 0, (char *)ords_rqst->data, ords_rqst->len, 0);
    }
}

int
order_status_byid_begin()
{
    deadlock = 0;


    dbrpcinit(dbproc, "order_status_byid", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &c_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &c_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &c_id);
    return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

void
order_status_byname_rpc()
{
    int try;

    for (try=0; try<MaxTries; try++)
    {
        if (try>0) display_xction(“Repeating”);

        if (order_status_byname_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (order_status_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }

    if (try >= MaxTries) {
            display_xction(“MaxTries Failed”);
            tpreturn(TPFAIL, 0, (char *)ords_rqst->data, ords_rqst->len, 0);
    }
}

int
order_status_byname_begin()
{
    deadlock = 0;
    dbrpcinit(dbproc, “order_status_byname”, 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &c_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &c_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(c_last), c_last);
    return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

int
order_status_end()
{
    int      count;

    if (dbsqlok(dbproc) != SUCCEED) return TRUE;

    if (dbresults(dbproc) != SUCCEED || deadlock)
        return TRUE;
    else {


        if (dbrows(dbproc) != SUCCEED )
        {
        invalid_xact++;
        }

        dbbind(dbproc, 1, SMALLBIND,     0,     &ol_supply_w_id);
        dbbind(dbproc, 2, INTBIND,       0,     &ol_i_id);
        dbbind(dbproc, 3, SMALLBIND,     0,     &ol_quantity);
        dbbind(dbproc, 4, FLT8BIND,      0,     &ol_amount);
        dbbind(dbproc, 5, DATETIMEBIND,  0,     &syb_date);

        count = 0;
        while ((code = dbnextrow(dbproc)) == REG_ROW && !deadlock)
        {
            ordstat->o_items[count].ol_supply_w_id = ol_supply_w_id;
            ordstat->o_items[count].ol_i_id = ol_i_id;
            ordstat->o_items[count].ol_quantity = ol_quantity;
            ordstat->o_items[count].ol_amount = ol_amount;

            sybdate2date(&syb_date, ol_delivery_d);
            strcpy(ordstat->o_items[count].ol_delivery_d, ol_delivery_d);
            count++;
        }
        ordstat->item_cnt = count;

        if (code != NO_MORE_ROWS || deadlock) return TRUE;
    }

    if (dbresults(dbproc) != SUCCEED || deadlock)
        return TRUE;
    else
    {

        if (dbrows(dbproc) != SUCCEED )
        {
        invalid_xact++;
        }


        dbbind(dbproc, 1, INTBIND,       0,     &c_id);
        dbbind(dbproc, 2, NTBSTRINGBIND, sizeof(c_last), c_last);
        dbbind(dbproc, 3, NTBSTRINGBIND, sizeof(c_first), c_first);
        dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(c_middle), c_middle);
        dbbind(dbproc, 5, FLT8BIND,      0,     &c_balance);
        dbbind(dbproc, 6, INTBIND,       0,     &o_id);
        dbbind(dbproc, 7, DATETIMEBIND,  0,     &syb_datetime);
        dbbind(dbproc, 8, SMALLBIND,     0,     &o_carrier_id);
        if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
        if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;

        sybdate2datetime(&syb_datetime, o_entry_d);
        strcpy(ordstat->c_first, c_first);
        strcpy(ordstat->c_middle, c_middle);
        strcpy(ordstat->c_last, c_last);
        ordstat->c_balance = c_balance;
        ordstat->o_id = (int)o_id;
        strcpy(ordstat->o_entry_d, o_entry_d);
        ordstat->o_carrier_id = o_carrier_id;
    }

    return FALSE;
}

void
delivery_rpc()
{
    int try;
```

```
        global_d_id = 1;
        for (try = 0; try < MaxTries; try++)
        {
            if (try > 0) display_xction(“Repeating”);

            if (delivery_body() == TRUE)
            {
                dbcancel(dbproc);
                sleep_before_retry();
                continue;
            }
            break;
        }
        if (try >= MaxTries)
        {
            display_xction(“MaxTries Failed”);

            fwrite(outbuf, strlen(outbuf), 1, delfile);
            fflush(delfile);
            tpreturn(TPFAIL, 0, (char *)del_rqst->data, del_rqst->len, 0);
        }
}

int
delivery_body()
{
    deadlock = 0;
    dbrpcinit(dbproc, “delivery”, 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &o_carrier_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &global_d_id);
    if (dbrpcsend(dbproc) != SUCCEED) return TRUE;
    if (dbsqlok(dbproc) != SUCCEED) return TRUE;

    for (; global_d_id <= 10; global_d_id++)
    {
    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;

    dbbind(dbproc, 1, INTBIND,       0,      &o_id);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    if (o_id == NULL)
         sprintf(outbuf+strlen(outbuf),
             “Delivery for District %d skipped\n”, global_d_id);
    else
         sprintf(outbuf+strlen(outbuf),
             “Delivered order %d for district %d, warehouse %d, carrier
%d\n”,
             o_id, global_d_id, global_w_id, o_carrier_id);

    if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
    if (dbhasretstat(dbproc) && dbretstatus(dbproc) != 0) return TRUE;
    }
    return FALSE;
}

void
stock_level_rpc()
{
    int try;


    for (try = 0; try < MaxTries; try ++)
    {
        if (try > 0) display_xction(“Repeating”);

        if (stock_level_body() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }

    if (try >= MaxTries)
    {
        display_xction(“MaxTries Failed”);
        tpreturn(TPFAIL, 0, (char *)stock_rqst->data, stock_rqst->len, 0);
    }
}


int
stock_level_body()
{
    int found, iid, uniq[500];
    int i, j, count;

    deadlock = 0;
    dbrpcinit(dbproc, “stock_level”, 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &global_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &global_d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &threshold);
    if (dbrpcsend(dbproc) != SUCCEED) return TRUE;
    if (dbsqlok(dbproc) != SUCCEED) return TRUE;

    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, INTBIND,  0, &iid);

    count = 0;
    while (dbnextrow(dbproc) == REG_ROW && !deadlock)
    {
        found = 0;
        for (j=0; j<count; j++)
        {
            if (iid == uniq[j])
            {
                found = 1;
                break;
            }
        }

        if (found == 0)
        {
        if (count >= 500)
            display_xction(“Too many rows returned by”);
        else
            uniq[count++] = iid;
        }
    }
    if (deadlock) return TRUE;
    if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;

    low_count = count;
    stocklevel->low_stock = low_count;
    return FALSE;
}

void ins_rpc()
{
    dbfcmd(dbproc,”insert into foo values(%d, ‘kjhkjhkjhkjhkjh’)”,
global_w_id);
    dbsqlexec(dbproc);
    dbresults(dbproc);
}

void
sleep_before_retry()
{
    sleep(1);
}

void
display_xction(msg)
char *msg;
{
    int i;
    userlog(“%s %s “, msg, func_array[xact_type].name);

    switch(xact_type)
    {

    case XACT_NEWO:
        userlog(“w=%d, d=%d, c=%d, %d lines: \n[“,
                        global_w_id, global_d_id, c_id, o_ol_cnt);
        for (i=0; i<(int)o_ol_cnt; i++)
            userlog(“ %d”, ol[i].i_id);
        userlog(“]\n”);
        break;

    case XACT_PAYM_ID:
        userlog(“w=%d/%d, d=%d/%d, c=%d\n”,
                        global_w_id, c_w_id, global_d_id, c_d_id, c_id);
        break;

    case XACT_PAYM_NAME:
        userlog(“w=%d/%d, d=%d/%d, l=%s\n”,
                        global_w_id, c_w_id, global_d_id, c_d_id, c_last);
        break;

    case XACT_ORDS_ID:
        userlog(“cw=%d, cd=%d, c=%d\n”, c_w_id, c_d_id, c_id);
        break;

    case XACT_ORDS_NAME:
        userlog(“cw=%d, cd=%d, l=%s\n”, c_w_id, c_d_id, c_last);
        break;

    case XACT_DEL:
        userlog(“w=%d, carrier=%d\n”, global_w_id, o_carrier_id);
        break;

    case XACT_STOCK:
        userlog(“w=%d, d=%d, th=%d\n”, global_w_id, global_d_id,
threshold);
        break;

    case XACT_BKEND:
        userlog(“w=%d, d=%d, carrier=%d, tx_count=%d\n”,
                        global_w_id, global_d_id, o_carrier_id, tx_count);
        break;

    default:
        userlog(“Unknown xact_type = %d\n”, xact_type);
    }
}

/*
 * Copyright (c)  1995, 1996, 1997, 1998, 1999  by Sun Microsystems, Inc.
 */

XCTION func_array[XCTION_COUNT+1] =
    {
        {“new_order”},
        {“payment_byid”},
        {“payment_byname”},
        {“order_status_byid”},
        {“order_status_byname”},
        {“delivery_qu”},
        {“stock_level”},
        {“delivery”},
        {“NULL”}
    };

int         rollback_pct;
int         lines_per_call = 15;
char        b_g[2];
DBFLT8      total_amount;
DBTINYINT commit_flag;
int         xact_type, prev_xact_type = -9999;
int         deadlock;
int         bad_items;
int         max_ware;
char        *db_name = “tpcc”;
RETCODE code;

DBSMALLINTglobal_w_id;
DBTINYINTglobal_d_id;
ORDER_LINEol[15];

DBINT       c_id;
DBTINYINTc_d_id;
DBSMALLINTc_w_id;
char        c_first[17];
char        c_middle[3];
char        c_last[17];
char        c_street_1[21];
char        c_street_2[21];
char        c_city[21];
char        c_state[3];
char        c_zip[10];
char        c_phone[17];
char        c_since[31];
char        c_credit[3];
DBFLT8      c_credit_lim;
DBREAL      c_discount;
DBFLT8      c_balance;
char        c_data[201];

char        w_name[11];
char        w_street_1[21];
char         w_street_2[21];
char         w_city[21];
char         w_state[3];
char         w_zip[10];
DBREAL   w_tax;

DBTINYINT       d_id;
DBSMALLINTd_w_id;
char    d_name[11];
char    d_street_1[21];
char    d_street_2[21];
char    d_city[21];
char    d_state[3];
char    d_zip[10];
DBREAL  d_tax;

int           i_id;
DBFLT8     i_price;
char         i_name[25];

DBSMALLINTs_quantity;
DBSMALLINTthreshold;
DBINT      low_count;
char         s_dist[25];

int           o_id;
DBTINYINTo_d_id;
DBSMALLINTo_w_id;
DBSMALLINTo_c_id;
char         o_entry_d[31];
DBSMALLINTo_carrier_id;
DBSMALLINTo_ol_cnt, o_ol_now, o_ol_done;
DBTINYINTo_all_local;

int           ol_o_id;
DBTINYINTol_d_id;
DBSMALLINTol_w_id;
DBSMALLINTol_number;
DBINT      ol_i_id;
DBSMALLINTol_supply_w_id;
char         ol_delivery_d[31];
DBSMALLINTol_quantity;
DBFLT8     ol_amount;

int           no_o_id;
DBTINYINTno_d_id;
DBSMALLINTno_w_id;

DBFLT8     h_amount;
char         h_date[20];
/*
 * Copyright (c)  1995, 1996, 1997, 1998, 1999  by Sun Microsystems, Inc.
 */

struct newo_inf  newosp;
struct newo_inf  *neworder;
FBFR                   *newo_fbfr;
TPSVCINFO                *newo_rqst;
int                    newolen;
struct track_mods       mod_array[50],
                    *modptr = mod_array;
```

```
struct req_struct *delp;
char               outbuf[1024];
int                tx_count = 0;
FILE               *delfile;
TPSVCINFO          *del_rqst;


struct ord_inf ordsp,
            *ordstat = &ordsp;
FBFR                 *ordsbuf;
int                  ordslen;
TPSVCINFO *ords_rqst;


struct pay_inf *payment;
struct pay_inf paymsp;
TPSVCINFO *paym_rqst;


struct stock_inf *stocklevel;
struct stock_inf stocksp;
TPSVCINFO *stock_rqst;


 /*
  * Copyright (c)  1995, 1996, 1997, 1998, 1999  by Sun Microsystems, Inc.
  */

#ifndef TPCC_TUXFORMS_H
#define TPCC_TUXFORMS_H

struct items_inf {
int ol_supply_w_id;
int ol_i_id;
char i_name[25];
int     ol_quantity;
int s_quantity;
char brand[2];
double i_price;
double ol_amount;
} ;

struct newo_inf {
        int     w_id;
        int     d_id;
        int     c_id;
        int     o_id;
        int     o_ol_cnt;
        double  c_discount;
        double  w_tax;
        double  d_tax;
        char    o_entry_d[20];
        char    c_credit[3];
        char    c_last[17];
        struct items_inf n_items[15];
        char    status[25];
        double  total;
};


#define DEL_SUCCESS     0
#define DEL_FAIL        1
#define DEL_RETRY       2
```

```
struct req_struct {
        int    w_id;
        int    o_carrier_id;
        time_t qtime;
};


struct ord_itm_inf {
int ol_supply_w_id;
int ol_i_id;
int     ol_quantity;
double ol_amount;
char ol_delivery_d[11];
} ;


struct ord_inf {
        int     item_cnt;
        int     w_id;
        int     d_id;
        int     c_id;
        int     o_id;
        int     o_carrier_id;
        double  c_balance;
        char    c_first[17];
        char    c_middle[3];
        char    c_last[17];
        char    o_entry_d[20];
        struct ord_itm_inf o_items[15];
};

struct pay_inf {
        int     w_id;
        int     d_id;
        int     c_id;
        int     c_w_id;
        int     c_d_id;
        double h_amount;
        double c_credit_lim;
        double c_balance;
        double c_discount;
        char  h_date[20];
        char  w_street_1[21];
        char  w_street_2[21];
        char  w_city[21];
        char  w_state[3];
        char  w_zip[11];
        char  d_street_1[21];
        char  d_street_2[21];
        char  d_city[21];
        char  d_state[3];
        char  d_zip[11];
        char  c_first[17];
        char  c_middle[3];
        char  c_last[17];
        char  c_street_1[21];
        char  c_street_2[21];
        char  c_city[21];
        char  c_state[3];
        char  c_zip[11];
        char  c_phone[17];
        char  c_since[11];
        char  c_credit[3];
        char  c_data_1[51];
        char  c_data_2[51];
        char  c_data_3[51];
        char  c_data_4[51];
};

struct stock_inf {
        int    w_id;
        int    d_id;
        int    threshold;
        int    low_stock;
};

#endif TPCC_TUXFORMS_H
/*
 * Copyright (c)  1995, 1996, 1997, 1998, 1999  by Sun Microsystems, Inc.
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

#include <sybfront.h>
#include <sybdb.h>
#include “SYB_tpcc.h”
#include “SYB_driver.h”

#include “atmi.h”
#include “userlog.h”

#include “tpcc_tux_forms.h”

struct newo_inf *neworder;
struct ord_inf *ordstat;
char blank_mesg[25] = “                        “;


#ifdef NOT_REQ
struct pay_inf {
int    w_id;
int    d_id;
int    c_id;
int    c_w_id;
int    c_d_id;
double h_amount;
double c_credit_lim;
double c_balance;
double c_discount;
char h_date[20];
char w_street_1[21];
char w_street_2[21];
char w_city[21];
char w_state[3];
char w_zip[11];
char d_street_1[21];
char d_street_2[21];
char d_city[21];
char d_state[3];
char d_zip[11];
char c_first[17];
char c_middle[3];
char c_last[17];
char c_street_1[21];
char c_street_2[21];
char c_city[21];
char c_state[3];
char c_zip[11];
char c_phone[17];
char c_since[11];
char c_credit[3];
char c_data_1[51];
char c_data_2[51];
char c_data_3[51];
char c_data_4[51];
};
#endif
struct pay_inf *payp;


DBPROCESS       *dbproc;
LOGINREC        *login;


int
init_all_tx()
{
userlog(“before dberrhandle \n”);
        dberrhandle(err_handler);
        dbmsghandle(msg_handler);

        deadlock = 0;

userlog(“before dblogin \n”);
        login = dblogin();
userlog(“before DBSETLUSER \n”);
        DBSETLUSER(login, USER);

userlog(“before DBSETLPACKET \n”);
        DBSETLPACKET(login, 4096);

userlog(“before DBSETLCHARSET \n”);
        DBSETLCHARSET(login, getenv(“CHARSET”));

userlog(“before dbopen \n”);
        if ((dbproc = dbopen(login, (char *)SERVER  )) == NULL)
        {
                initerr(“Fatal dbopen: Could not open connection\n”);
                return(-1);
        }

userlog(“before dbuse \n”);
        if ( dbuse(dbproc, (char *)DATABASE) != SUCCEED)
        {
                initerr(“Fatal dbuse: Could not use DATABASE\n”);
                return(-1);
        }

userlog(“leaving tpsvrinit \n”);
        return(0);
```

```
}

neworder_tx(rqst)
TPSVCINFO *rqst;
{

        int     i;
int rollback = 0;
int linecnt;
        int ret;
struct items_inf *cur_ip;


        neworder = (struct newo_inf *)(rqst->data);
        linecnt = neworder->o_ol_cnt;

        strncpy(neworder->status, blank_mesg, 24);
again:
        neworder->total = 0;

        global_w_id = neworder->w_id;
        global_d_id = neworder->d_id;
        c_id = neworder->c_id;
        o_ol_cnt = neworder->o_ol_cnt;
        o_all_local = 1;

        for (i = 0; i < (int)o_ol_cnt ; i++) {
                cur_ip = &neworder->n_items[i];

                ol[i].i_id = cur_ip->ol_i_id;
                ol[i].supply_w_id = cur_ip->ol_supply_w_id;
                ol[i].quantity = cur_ip->ol_quantity;

                if (ol[i].supply_w_id != global_w_id)
                        o_all_local = 0;
        }

        new_order_rpc();


        neworder->total = total_amount;

        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct newo_inf), 0);
}

int
tpsvrinit(argc, argv)
char **argv;
{
        return(init_all_tx());
}

void
tpsvrdone()
{
        dbexit();
}

NEWO(rqst)
TPSVCINFO *rqst;
{
        xact_type = XACT_NEWO;
        neworder_tx(rqst);
}


initerr(str)
char *str;
{
        userlog(“init_all_tx ERROR during %s\n”, str);
}

ordstat_tx(rqst)
TPSVCINFO *rqst;
{
        int     byid;

        ordstat = (struct ord_inf *)(rqst->data);
        if (ordstat->c_id == 0) {
                byid = FALSE;
                xact_type = XACT_ORDS_NAME;
        }
        else {
                byid = TRUE;
                xact_type = XACT_ORDS_ID;
        }

        c_w_id = ordstat->w_id;
        c_d_id = ordstat->d_id;
        if (!byid) {
                strcpy(c_last, ordstat->c_last);
                order_status_byname_rpc();
                ordstat->c_id = c_id;
        }
        else {
                c_id = ordstat->c_id;
                order_status_byid_rpc();
                strcpy(ordstat->c_last, c_last);
        }
                tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct ord_inf), 0);
}

ORDS(rqst)
TPSVCINFO *rqst;
{
        ordstat_tx(rqst);
}

payment_tx(rqst)
TPSVCINFO *rqst;
{
        int     byid;

        payp = (struct pay_inf *)(rqst->data);

        global_w_id = payp->w_id;
        c_w_id = payp->c_w_id;
        h_amount = payp->h_amount;
        global_d_id = payp->d_id;
        c_d_id = payp->c_d_id;
        if (payp->c_id == 0) {
                byid = FALSE;
                xact_type = XACT_PAYM_NAME;
        }
        else {
                byid = TRUE;
                xact_type = XACT_PAYM_ID;
        }

        if (byid) {
                c_id = payp->c_id;
                payment_byid_rpc();
        }
        else {
                strcpy(c_last, payp->c_last);
                payment_byname_rpc();
                payp->c_id = c_id;
        }

        strcpy(payp->h_date, h_date);
        strcpy(payp->w_street_1, w_street_1);
        strcpy(payp->w_street_2, w_street_2);
        strcpy(payp->w_city, w_city);
        strcpy(payp->w_state, w_state);
        strcpy(payp->w_zip, w_zip);

        strcpy(payp->d_street_1, d_street_1);
        strcpy(payp->d_street_2, d_street_2);
        strcpy(payp->d_city, d_city);
        strcpy(payp->d_state, d_state);
        strcpy(payp->d_zip, d_zip);

        strcpy(payp->c_first, c_first);
        strcpy(payp->c_middle, c_middle);
        strcpy(payp->c_last, c_last);
        strcpy(payp->c_street_1, c_street_1);
        strcpy(payp->c_street_2, c_street_2);
        strcpy(payp->c_city, c_city);
        strcpy(payp->c_state, c_state);
        strcpy(payp->c_zip, c_zip);
        strcpy(payp->c_phone, c_phone);
        strcpy(payp->c_since, c_since);
        strcpy(payp->c_credit, c_credit);

        payp->c_credit_lim = c_credit_lim;
        payp->c_discount = c_discount;
        payp->c_balance = c_balance;

        if ( c_data == 0 ) {
                payp->c_data_1[0] =
                payp->c_data_2[0] =
                payp->c_data_3[0] =
                payp->c_data_4[0] = 0;
        }
        else {
                strncpy(payp->c_data_1, c_data, 50);
                strncpy(payp->c_data_2, c_data + 50, 50);
                strncpy(payp->c_data_3, c_data + 100, 50);
                strncpy(payp->c_data_4, c_data + 150, 50);
        }
        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct pay_inf), 0);
}

PAYM(rqst)
TPSVCINFO *rqst;
{
        payment_tx(rqst);
}
```

```
/*
 * Copyright (c)  1995, 1996, 1997, 1998, 1999  by Sun Microsystems, Inc.
 */


#include “tpcc_client.h”
#include <stdlib.h>
#include <sys/signal.h>
#include <sys/utsname.h>
#include <errno.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>

#include “tpcc_tux_forms.h”

#include “atmi.h”
#include “userlog.h”

#include <sybfront.h>
#include <sybdb.h>
#include “SYB_tpcc.h”
#include “SYB_driver.h”


static struct req_struct *delp;
extern char outbuf[];
extern int tx_count;
extern FILE *delfile;

struct stock_inf *stocklevel;


DBPROCESS       *dbproc;
LOGINREC        *login;

cleanup()
{
        fclose(delfile);
}


int
init_stockdel_tx()
{
    userlog(“before dberrhandle \n”);
        dberrhandle(err_handler);
        dbmsghandle(msg_handler);

        deadlock = 0;


    userlog(“before dblogin \n”);
        login = dblogin();
    userlog(“before DBSETLUSER \n”);
        DBSETLUSER(login, USER);

    userlog(“before DBSETLPACKET \n”);

        DBSETLPACKET(login, 4096);
    userlog(“before DBSETLCHARSET \n”);
        DBSETLCHARSET(login, getenv(“CHARSET”));

    userlog(“before dbopen \n”);
        if ((dbproc = dbopen(login, (char *)SERVER  )) == NULL)
        {
            initerr(“Fatal dbopen: Could not open connection\n”);
            return(-1);
        }

    userlog(“before dbuse \n”);
        if ( dbuse(dbproc, (char *)DATABASE) != SUCCEED)
        {
            initerr(“Fatal dbuse: Could not use DATABASE\n”);
            return(-1);
        }

    userlog(“leaving tpsvrinit \n”);
        return(0);

}


delivery_tx(rqst)
TPSVCINFO *rqst;
{

        delp = (struct req_struct *)(rqst->data);
        global_w_id = delp->w_id;
        o_carrier_id = delp->o_carrier_id;
        tx_count++;
        sprintf(outbuf, “Starting transaction %d queued at %d\n”,
                tx_count, delp->qtime);

        delivery_rpc();


        sprintf(outbuf+strlen(outbuf), “Transaction completed at %d\n”, time(0));
        fwrite(outbuf, strlen(outbuf), 1, delfile);
        fflush(delfile);
        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct req_struct), 0);
}


initerr(str)
char *str;
{
        userlog(“init_stockdel_tx ERROR %\n”, str);
}


tpsvrinit(argc, argv)
char **argv;
{
        char *p, ident[20];
        char filename[200];
        int proc_no, count;
        struct utsname name;


        if ((p = getenv(“TMPDIR”)) == (char *)NULL) {
                userlog(“TMPDIR environment variable not set\n”);
                exit(1);
        }

        proc_no = (int)getpid();
        uname( &name);
        strcpy(filename, p);
        sprintf(filename+strlen(filename), “/%s.del%d”, name.nodename,
proc_no);
        userlog(“filename = %s \n”,filename);
        delfile = fopen(filename, “w”);
        if (delfile == NULL) {
                userlog(“Cannot create file %s\n”, filename);
        }
        return(init_stockdel_tx());
}

void
tpsvrdone()
{
        cleanup();
        dbexit();
}

DEL(rqst)
TPSVCINFO *rqst;
{
        xact_type = XACT_BKEND;
        delivery_tx(rqst);
}

stocklevel_tx(rqst)
TPSVCINFO *rqst;
{

        stocklevel = (struct stock_inf *)(rqst->data);

        global_w_id = stocklevel->w_id;
        global_d_id = stocklevel->d_id;
        threshold = stocklevel->threshold;

        stock_level_rpc();

        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct stock_inf), 0);
}

STOCK(rqst)
TPSVCINFO *rqst;
{
        xact_type = XACT_STOCK;
        stocklevel_tx(rqst);
}
```

Appendix B: Database Design

This Appendix contains the scripts used to create the database and the load program used to load the database initially.

```
########################################
#       devices.4080w.sql
########################################
disk init
name = ‘logs01’,
physname = ‘/syb_rdsk/logs001’,
vdevno = 1,
size = 4096000
disk init
name = ‘hist01’,
physname = ‘/syb_rdsk/hist001’,
vdevno = 2,
size = 2048000
disk init
name = ‘hist02’,
physname = ‘/syb_rdsk/hist002’,
vdevno = 3,
size = 2048000
disk init
name = ‘orders01’,
physname = ‘/syb_rdsk/order001’,
vdevno = 4,
size = 819200
disk init
name = ‘orders02’,
physname = ‘/syb_rdsk/order002’,
vdevno = 5,
size = 819200
disk init
name = ‘orders03’,
physname = ‘/syb_rdsk/order003’,
vdevno = 6,
size = 819200
disk init
name = ‘orders04’,
physname = ‘/syb_rdsk/order004’,
vdevno = 7,
size = 819200
disk init
name = ‘orders05’,
physname = ‘/syb_rdsk/order005’,
vdevno = 8,
size = 819200
disk init
name = ‘oline01’,
physname = ‘/syb_rdsk/oline001’,
vdevno = 9,
size = 2790400
disk init
name = ‘oline02’,
physname = ‘/syb_rdsk/oline002’,
vdevno = 10,
size = 2790400
disk init
name = ‘oline03’,
physname = ‘/syb_rdsk/oline003’,
vdevno = 11,
size = 2790400
disk init
name = ‘oline04’,
physname = ‘/syb_rdsk/oline004’,
vdevno = 12,
size = 2790400
disk init
name = ‘oline05’,
physname = ‘/syb_rdsk/oline005’,
vdevno = 13,
size = 2790400
disk init
name = ‘oline06’,
physname = ‘/syb_rdsk/oline006’,
vdevno = 14,
size = 2790400
disk init
name = ‘oline07’,
physname = ‘/syb_rdsk/oline007’,
vdevno = 15,
size = 2790400
disk init
name = ‘oline08’,
physname = ‘/syb_rdsk/oline008’,
vdevno = 16,
size = 2790400
disk init
name = ‘oline09’,
physname = ‘/syb_rdsk/oline009’,
vdevno = 17,
size = 2790400
disk init
name = ‘oline10’,
physname = ‘/syb_rdsk/oline010’,
vdevno = 18,
size = 2790400
disk init
name = ‘oline11’,
physname = ‘/syb_rdsk/oline011’,
vdevno = 19,
size = 2790400
disk init
name = ‘oline12’,
physname = ‘/syb_rdsk/oline012’,
vdevno = 20,
size = 2790400
disk init
name = ‘oline13’,
physname = ‘/syb_rdsk/oline013’,
vdevno = 21,
size = 2790400
disk init
name = ‘oline14’,
physname = ‘/syb_rdsk/oline014’,
vdevno = 22,
size = 2790400
disk init
name = ‘oline15’,
physname = ‘/syb_rdsk/oline015’,
vdevno = 23,
size = 2790400
disk init
name = ‘oline16’,
physname = ‘/syb_rdsk/oline016’,
vdevno = 24,
size = 2790400
disk init
name = ‘oline17’,
physname = ‘/syb_rdsk/oline017’,
vdevno = 25,
size = 2790400
disk init
name = ‘cust01’,
physname = ‘/syb_rdsk/cust001’,
vdevno = 26,
size = 2022400
disk init
name = ‘cust02’,
physname = ‘/syb_rdsk/cust002’,
vdevno = 27,
size = 2022400
disk init
name = ‘cust03’,
physname = ‘/syb_rdsk/cust003’,
vdevno = 28,
size = 2022400
disk init
name = ‘cust04’,
physname = ‘/syb_rdsk/cust004’,
vdevno = 29,
size = 2022400
disk init
name = ‘cust05’,
physname = ‘/syb_rdsk/cust005’,
vdevno = 30,
size = 2022400
disk init
name = ‘cust06’,
physname = ‘/syb_rdsk/cust006’,
vdevno = 31,
size = 2022400
disk init
name = ‘cust07’,
physname = ‘/syb_rdsk/cust007’,
vdevno = 32,
size = 2022400
disk init
name = ‘cust08’,
physname = ‘/syb_rdsk/cust008’,
vdevno = 33,
size = 2022400
disk init
name = ‘cust09’,
physname = ‘/syb_rdsk/cust009’,
vdevno = 34,
size = 2022400
disk init
name = ‘cust10’,
physname = ‘/syb_rdsk/cust010’,
vdevno = 35,
size = 2022400
disk init
name = ‘cust11’,
physname = ‘/syb_rdsk/cust011’,
vdevno = 36,
size = 2022400
disk init
name = ‘cust12’,
physname = ‘/syb_rdsk/cust012’,
vdevno = 37,
size = 2022400
disk init
name = ‘cust13’,
physname = ‘/syb_rdsk/cust013’,
vdevno = 38,
size = 2022400
disk init
name = ‘cust14’,
physname = ‘/syb_rdsk/cust014’,
vdevno = 39,
size = 2022400
disk init
name = ‘cust15’,
physname = ‘/syb_rdsk/cust015’,
vdevno = 40,
size = 2022400
disk init
name = ‘cust16’,
physname = ‘/syb_rdsk/cust016’,
vdevno = 41,
size = 2022400
disk init
name = ‘cust17’,
physname = ‘/syb_rdsk/cust017’,
vdevno = 42,
size = 2022400
disk init
name = ‘cust18’,
physname = ‘/syb_rdsk/cust018’,
vdevno = 43,
size = 2022400
disk init
name = ‘cust19’,
physname = ‘/syb_rdsk/cust019’,
vdevno = 44,
size = 2022400
disk init
name = ‘cust20’,
physname = ‘/syb_rdsk/cust020’,
vdevno = 45,
size = 2022400
disk init
name = ‘cust21’,
physname = ‘/syb_rdsk/cust021’,
vdevno = 46,
size = 2022400
disk init
name = ‘cust22’,
physname = ‘/syb_rdsk/cust022’,
vdevno = 47,
size = 2022400
disk init
name = ‘cust23’,
physname = ‘/syb_rdsk/cust023’,
vdevno = 48,
size = 2022400
disk init
name = ‘stock01’,
physname = ‘/syb_rdsk/stock001’,
vdevno = 49,
size = 2073600
disk init
name = ‘stock02’,
physname = ‘/syb_rdsk/stock002’,
vdevno = 50,
size = 2073600
disk init
name = ‘stock03’,
physname = ‘/syb_rdsk/stock003’,
vdevno = 51,
size = 2073600
disk init
name = ‘stock04’,
physname = ‘/syb_rdsk/stock004’,
vdevno = 52,
size = 2073600
disk init
name = ‘stock05’,
physname = ‘/syb_rdsk/stock005’,
vdevno = 53,
size = 2073600
disk init
name = ‘stock06’,
physname = ‘/syb_rdsk/stock006’,
vdevno = 54,
size = 2073600
disk init
name = ‘stock07’,
physname = ‘/syb_rdsk/stock007’,
vdevno = 55,
size = 2073600
disk init
name = ‘stock08’,
physname = ‘/syb_rdsk/stock008’,
vdevno = 56,
size = 2073600
disk init
name = ‘stock09’,
physname = ‘/syb_rdsk/stock009’,
vdevno = 57,
size = 2073600
disk init
name = ‘stock10’,
physname = ‘/syb_rdsk/stock010’,
vdevno = 58,
size = 2073600
disk init
name = ‘stock11’,
physname = ‘/syb_rdsk/stock011’,
vdevno = 59,
size = 2073600
disk init
name = ‘stock12’,
physname = ‘/syb_rdsk/stock012’,
vdevno = 60,
size = 2073600
disk init
name = ‘stock13’,
physname = ‘/syb_rdsk/stock013’,
vdevno = 61,
size = 2073600
disk init
name = ‘stock14’,
physname = ‘/syb_rdsk/stock014’,
vdevno = 62,
size = 2073600
disk init
name = ‘stock15’,
physname = ‘/syb_rdsk/stock015’,
vdevno = 63,
size = 2073600
disk init
name = ‘stock16’,
physname = ‘/syb_rdsk/stock016’,
vdevno = 64,
size = 2073600
disk init
name = ‘stock17’,
physname = ‘/syb_rdsk/stock017’,
vdevno = 65,
size = 2073600
disk init
name = ‘stock18’,
physname = ‘/syb_rdsk/stock018’,
vdevno = 66,
size = 2073600
disk init
name = ‘stock19’,
physname = ‘/syb_rdsk/stock019’,
vdevno = 67,
size = 2073600
disk init
name = ‘stock20’,
physname = ‘/syb_rdsk/stock020’,
vdevno = 68,
size = 2073600
disk init
name = ‘stock21’,
physname = ‘/syb_rdsk/stock021’,
vdevno = 69,
size = 2073600
disk init
name = ‘stock22’,
physname = ‘/syb_rdsk/stock022’,
vdevno = 70,
size = 2073600
disk init
name = ‘stock23’,
physname = ‘/syb_rdsk/stock023’,
vdevno = 71,
size = 2073600
disk init
name = ‘stock24’,
physname = ‘/syb_rdsk/stock024’,
vdevno = 72,
size = 2073600
disk init
name = ‘stock25’,
physname = ‘/syb_rdsk/stock025’,
vdevno = 73,
size = 2073600
disk init
name = ‘stock26’,
physname = ‘/syb_rdsk/stock026’,
vdevno = 74,
size = 2073600
disk init
name = ‘stock27’,
physname = ‘/syb_rdsk/stock027’,
vdevno = 75,
size = 2073600
disk init
name = ‘stock28’,
physname = ‘/syb_rdsk/stock028’,
vdevno = 76,
size = 2073600
disk init
name = ‘stock29’,
physname = ‘/syb_rdsk/stock029’,
vdevno = 77,
size = 2073600
disk init
name = ‘stock30’,
physname = ‘/syb_rdsk/stock030’,
vdevno = 78,
size = 2073600
disk init
name = ‘stock31’,
physname = ‘/syb_rdsk/stock031’,
vdevno = 79,
size = 2073600
disk init
name = ‘stock32’,
physname = ‘/syb_rdsk/stock032’,
vdevno = 80,
size = 2073600
disk init
name = ‘stock33’,
physname = ‘/syb_rdsk/stock033’,
vdevno = 81,
size = 2073600
disk init
name = ‘stock34’,
physname = ‘/syb_rdsk/stock034’,
vdevno = 82,
size = 2073600
disk init
name = ‘stock35’,
physname = ‘/syb_rdsk/stock035’,
vdevno = 83,
size = 2073600
disk init
name = ‘stock36’,
physname = ‘/syb_rdsk/stock036’,
vdevno = 84,
size = 2073600
disk init
name = ‘stock37’,
physname = ‘/syb_rdsk/stock037’,
vdevno = 85,
size = 2073600
disk init
name = ‘stock38’,
physname = ‘/syb_rdsk/stock038’,
vdevno = 86,
size = 2073600
disk init
name = ‘stock39’,
physname = ‘/syb_rdsk/stock039’,
vdevno = 87,
size = 2073600
disk init
name = ‘stock40’,
physname = ‘/syb_rdsk/stock040’,
vdevno = 88,
size = 2073600
disk init
name = ‘stock41’,
physname = ‘/syb_rdsk/stock041’,
vdevno = 89,
size = 2073600
########################################
#       extendlog.sql
########################################
#!/bin/ksh
isql -Usa -P << EOF
```

```
disk init name = “logs02”,
physname = “/syb_rdsk/logs002” ,
vdevno = 90, size = 4096000
go
alter database tpcc log on logs02 = 8000
go
disk init name = “logs03” ,
physname = “/syb_rdsk/logs003” ,
vdevno = 91,
size = 4096000
go
alter database tpcc log on logs03 = 8000
go
EOF

########################################
#       segments.sh
########################################
use tpcc
sp_addsegment Scache , tpcc , master
sp_addsegment Shist , tpcc , hist01
sp_extendsegment Shist , tpcc , hist02
sp_addsegment Sorders , tpcc , orders01
sp_extendsegment Sorders , tpcc , orders02
sp_extendsegment Sorders , tpcc , orders03
sp_extendsegment Sorders , tpcc , orders04
sp_extendsegment Sorders , tpcc , orders05
sp_addsegment Soline , tpcc , oline01
sp_extendsegment Soline , tpcc , oline02
sp_extendsegment Soline , tpcc , oline03
sp_extendsegment Soline , tpcc , oline04
sp_extendsegment Soline , tpcc , oline05
sp_extendsegment Soline , tpcc , oline06
sp_extendsegment Soline , tpcc , oline07
sp_extendsegment Soline , tpcc , oline08
sp_extendsegment Soline , tpcc , oline09
sp_extendsegment Soline , tpcc , oline10
sp_extendsegment Soline , tpcc , oline11
sp_extendsegment Soline , tpcc , oline12
sp_extendsegment Soline , tpcc , oline13
sp_extendsegment Soline , tpcc , oline14
sp_extendsegment Soline , tpcc , oline15
sp_extendsegment Soline , tpcc , oline16
sp_extendsegment Soline , tpcc , oline17
sp_addsegment Scust , tpcc , cust01
sp_extendsegment Scust , tpcc , cust02
sp_extendsegment Scust , tpcc , cust03
sp_extendsegment Scust , tpcc , cust04
sp_extendsegment Scust , tpcc , cust05
sp_extendsegment Scust , tpcc , cust06
sp_extendsegment Scust , tpcc , cust07
sp_extendsegment Scust , tpcc , cust08
sp_extendsegment Scust , tpcc , cust09
sp_extendsegment Scust , tpcc , cust10
sp_extendsegment Scust , tpcc , cust11
sp_extendsegment Scust , tpcc , cust12
sp_extendsegment Scust , tpcc , cust13
sp_extendsegment Scust , tpcc , cust14
sp_extendsegment Scust , tpcc , cust15
sp_extendsegment Scust , tpcc , cust16
sp_extendsegment Scust , tpcc , cust17
sp_extendsegment Scust , tpcc , cust18
sp_extendsegment Scust , tpcc , cust19
```

```
sp_extendsegment Scust , tpcc , cust20
sp_extendsegment Scust , tpcc , cust21
sp_extendsegment Scust , tpcc , cust22
sp_extendsegment Scust , tpcc , cust23
sp_addsegment Sstock , tpcc , stock01
sp_extendsegment Sstock , tpcc , stock02
sp_extendsegment Sstock , tpcc , stock03
sp_extendsegment Sstock , tpcc , stock04
sp_extendsegment Sstock , tpcc , stock05
sp_extendsegment Sstock , tpcc , stock06
sp_extendsegment Sstock , tpcc , stock07
sp_extendsegment Sstock , tpcc , stock08
sp_extendsegment Sstock , tpcc , stock09
sp_extendsegment Sstock , tpcc , stock10
sp_extendsegment Sstock , tpcc , stock11
sp_extendsegment Sstock , tpcc , stock12
sp_extendsegment Sstock , tpcc , stock13
sp_extendsegment Sstock , tpcc , stock14
sp_extendsegment Sstock , tpcc , stock15
sp_extendsegment Sstock , tpcc , stock16
sp_extendsegment Sstock , tpcc , stock17
sp_extendsegment Sstock , tpcc , stock18
sp_extendsegment Sstock , tpcc , stock19
sp_extendsegment Sstock , tpcc , stock20
sp_extendsegment Sstock , tpcc , stock21
sp_extendsegment Sstock , tpcc , stock22
sp_extendsegment Sstock , tpcc , stock23
sp_extendsegment Sstock , tpcc , stock24
sp_extendsegment Sstock , tpcc , stock25
sp_extendsegment Sstock , tpcc , stock26
sp_extendsegment Sstock , tpcc , stock27
sp_extendsegment Sstock , tpcc , stock28
sp_extendsegment Sstock , tpcc , stock29
sp_extendsegment Sstock , tpcc , stock30
sp_extendsegment Sstock , tpcc , stock31
sp_extendsegment Sstock , tpcc , stock32
sp_extendsegment Sstock , tpcc , stock33
sp_extendsegment Sstock , tpcc , stock34
sp_extendsegment Sstock , tpcc , stock35
sp_extendsegment Sstock , tpcc , stock36
sp_extendsegment Sstock , tpcc , stock37
sp_extendsegment Sstock , tpcc , stock38
sp_extendsegment Sstock , tpcc , stock39
sp_extendsegment Sstock , tpcc , stock40
sp_extendsegment Sstock , tpcc , stock41
sp_dropsegment ‘default’, tpcc , hist01
sp_dropsegment ‘system’, tpcc , hist01
sp_dropsegment ‘default’, tpcc , hist02
sp_dropsegment ‘system’, tpcc , hist02
sp_dropsegment ‘default’, tpcc , orders01
sp_dropsegment ‘system’, tpcc , orders01
sp_dropsegment ‘default’, tpcc , orders02
sp_dropsegment ‘system’, tpcc , orders02
sp_dropsegment ‘default’, tpcc , orders03
sp_dropsegment ‘system’, tpcc , orders03
sp_dropsegment ‘default’, tpcc , orders04
sp_dropsegment ‘system’, tpcc , orders04
sp_dropsegment ‘default’, tpcc , orders05
sp_dropsegment ‘system’, tpcc , orders05
sp_dropsegment ‘default’, tpcc , oline01
sp_dropsegment ‘system’, tpcc , oline01
sp_dropsegment ‘default’, tpcc , oline02
sp_dropsegment ‘system’, tpcc , oline02
sp_dropsegment ‘default’, tpcc , oline03
sp_dropsegment ‘system’, tpcc , oline03
sp_dropsegment ‘default’, tpcc , oline04
sp_dropsegment ‘system’, tpcc , oline04
sp_dropsegment ‘default’, tpcc , oline05
sp_dropsegment ‘system’, tpcc , oline05
sp_dropsegment ‘default’, tpcc , oline06
sp_dropsegment ‘system’, tpcc , oline06
sp_dropsegment ‘default’, tpcc , oline07
sp_dropsegment ‘system’, tpcc , oline07
sp_dropsegment ‘default’, tpcc , oline08
sp_dropsegment ‘system’, tpcc , oline08
sp_dropsegment ‘default’, tpcc , oline09
sp_dropsegment ‘system’, tpcc , oline09
sp_dropsegment ‘default’, tpcc , oline10
sp_dropsegment ‘system’, tpcc , oline10
sp_dropsegment ‘default’, tpcc , oline11
sp_dropsegment ‘system’, tpcc , oline11
sp_dropsegment ‘default’, tpcc , oline12
sp_dropsegment ‘system’, tpcc , oline12
sp_dropsegment ‘default’, tpcc , oline13
sp_dropsegment ‘system’, tpcc , oline13
sp_dropsegment ‘default’, tpcc , oline14
sp_dropsegment ‘system’, tpcc , oline14
sp_dropsegment ‘default’, tpcc , oline15
sp_dropsegment ‘system’, tpcc , oline15
sp_dropsegment ‘default’, tpcc , oline16
sp_dropsegment ‘system’, tpcc , oline16
sp_dropsegment ‘default’, tpcc , oline17
sp_dropsegment ‘system’, tpcc , oline17
sp_dropsegment ‘default’, tpcc , cust01
sp_dropsegment ‘system’, tpcc , cust01
sp_dropsegment ‘default’, tpcc , cust02
sp_dropsegment ‘system’, tpcc , cust02
sp_dropsegment ‘default’, tpcc , cust03
sp_dropsegment ‘system’, tpcc , cust03
sp_dropsegment ‘default’, tpcc , cust04
sp_dropsegment ‘system’, tpcc , cust04
sp_dropsegment ‘default’, tpcc , cust05
sp_dropsegment ‘system’, tpcc , cust05
sp_dropsegment ‘default’, tpcc , cust06
sp_dropsegment ‘system’, tpcc , cust06
sp_dropsegment ‘default’, tpcc , cust07
sp_dropsegment ‘system’, tpcc , cust07
sp_dropsegment ‘default’, tpcc , cust08
sp_dropsegment ‘system’, tpcc , cust08
sp_dropsegment ‘default’, tpcc , cust09
sp_dropsegment ‘system’, tpcc , cust09
sp_dropsegment ‘default’, tpcc , cust10
sp_dropsegment ‘system’, tpcc , cust10
sp_dropsegment ‘default’, tpcc , cust11
sp_dropsegment ‘system’, tpcc , cust11
sp_dropsegment ‘default’, tpcc , cust12
sp_dropsegment ‘system’, tpcc , cust12
sp_dropsegment ‘default’, tpcc , cust13
sp_dropsegment ‘system’, tpcc , cust13
sp_dropsegment ‘default’, tpcc , cust14
sp_dropsegment ‘system’, tpcc , cust14
sp_dropsegment ‘default’, tpcc , cust15
sp_dropsegment ‘system’, tpcc , cust15
sp_dropsegment ‘default’, tpcc , cust16
sp_dropsegment ‘system’, tpcc , cust16
sp_dropsegment ‘default’, tpcc , cust17
sp_dropsegment ‘system’, tpcc , cust17
sp_dropsegment ‘default’, tpcc , cust18
sp_dropsegment ‘system’, tpcc , cust18
sp_dropsegment ‘default’, tpcc , cust19
sp_dropsegment ‘system’, tpcc , cust19
sp_dropsegment ‘default’, tpcc , cust20
sp_dropsegment ‘system’, tpcc , cust20
sp_dropsegment ‘default’, tpcc , cust21
sp_dropsegment ‘system’, tpcc , cust21
sp_dropsegment ‘default’, tpcc , cust22
sp_dropsegment ‘system’, tpcc , cust22
sp_dropsegment ‘default’, tpcc , cust23
sp_dropsegment ‘system’, tpcc , cust23
sp_dropsegment ‘default’, tpcc , stock01
sp_dropsegment ‘system’, tpcc , stock01
sp_dropsegment ‘default’, tpcc , stock02
sp_dropsegment ‘system’, tpcc , stock02
sp_dropsegment ‘default’, tpcc , stock03
sp_dropsegment ‘system’, tpcc , stock03
sp_dropsegment ‘default’, tpcc , stock04
sp_dropsegment ‘system’, tpcc , stock04
sp_dropsegment ‘default’, tpcc , stock05
sp_dropsegment ‘system’, tpcc , stock05
sp_dropsegment ‘default’, tpcc , stock06
sp_dropsegment ‘system’, tpcc , stock06
sp_dropsegment ‘default’, tpcc , stock07
sp_dropsegment ‘system’, tpcc , stock07
sp_dropsegment ‘default’, tpcc , stock08
sp_dropsegment ‘system’, tpcc , stock08
sp_dropsegment ‘default’, tpcc , stock09
sp_dropsegment ‘system’, tpcc , stock09
sp_dropsegment ‘default’, tpcc , stock10
sp_dropsegment ‘system’, tpcc , stock10
sp_dropsegment ‘default’, tpcc , stock11
sp_dropsegment ‘system’, tpcc , stock11
sp_dropsegment ‘default’, tpcc , stock12
sp_dropsegment ‘system’, tpcc , stock12
sp_dropsegment ‘default’, tpcc , stock13
sp_dropsegment ‘system’, tpcc , stock13
sp_dropsegment ‘default’, tpcc , stock14
sp_dropsegment ‘system’, tpcc , stock14
sp_dropsegment ‘default’, tpcc , stock15
sp_dropsegment ‘system’, tpcc , stock15
sp_dropsegment ‘default’, tpcc , stock16
sp_dropsegment ‘system’, tpcc , stock16
sp_dropsegment ‘default’, tpcc , stock17
sp_dropsegment ‘system’, tpcc , stock17
sp_dropsegment ‘default’, tpcc , stock18
sp_dropsegment ‘system’, tpcc , stock18
sp_dropsegment ‘default’, tpcc , stock19
sp_dropsegment ‘system’, tpcc , stock19
sp_dropsegment ‘default’, tpcc , stock20
sp_dropsegment ‘system’, tpcc , stock20
sp_dropsegment ‘default’, tpcc , stock21
sp_dropsegment ‘system’, tpcc , stock21
sp_dropsegment ‘default’, tpcc , stock22
sp_dropsegment ‘system’, tpcc , stock22
sp_dropsegment ‘default’, tpcc , stock23
sp_dropsegment ‘system’, tpcc , stock23
sp_dropsegment ‘default’, tpcc , stock24
sp_dropsegment ‘system’, tpcc , stock24
sp_dropsegment ‘default’, tpcc , stock25
sp_dropsegment ‘system’, tpcc , stock25
sp_dropsegment ‘default’, tpcc , stock26
sp_dropsegment ‘system’, tpcc , stock26
sp_dropsegment ‘default’, tpcc , stock27
sp_dropsegment ‘system’, tpcc , stock27
sp_dropsegment ‘default’, tpcc , stock28
sp_dropsegment ‘system’, tpcc , stock28
sp_dropsegment ‘default’, tpcc , stock29
sp_dropsegment ‘system’, tpcc , stock29
sp_dropsegment ‘default’, tpcc , stock30
sp_dropsegment ‘system’, tpcc , stock30
sp_dropsegment ‘default’, tpcc , stock31
sp_dropsegment ‘system’, tpcc , stock31
sp_dropsegment ‘default’, tpcc , stock32
sp_dropsegment ‘system’, tpcc , stock32
sp_dropsegment ‘default’, tpcc , stock33
sp_dropsegment ‘system’, tpcc , stock33
sp_dropsegment ‘default’, tpcc , stock34
sp_dropsegment ‘system’, tpcc , stock34
sp_dropsegment ‘default’, tpcc , stock35
sp_dropsegment ‘system’, tpcc , stock35
sp_dropsegment ‘default’, tpcc , stock36
sp_dropsegment ‘system’, tpcc , stock36
sp_dropsegment ‘default’, tpcc , stock37
sp_dropsegment ‘system’, tpcc , stock37
sp_dropsegment ‘default’, tpcc , stock38
sp_dropsegment ‘system’, tpcc , stock38
sp_dropsegment ‘default’, tpcc , stock39
sp_dropsegment ‘system’, tpcc , stock39
sp_dropsegment ‘default’, tpcc , stock40
sp_dropsegment ‘system’, tpcc , stock40
sp_dropsegment ‘default’, tpcc , stock41
sp_dropsegment ‘system’, tpcc , stock41

########################################3
#       tpcc_indexes.sh
########################################3

#!/bin/sh -f
# This script will create the TPC-C indexes that are best
# created after the load.
#
isql -e -Usa -P$PASSWORD << EOF
use tpcc
go

create unique clustered index w_clu
        on warehouse(w_id)
        on Scache
go
dbcc tune(indextrips, 100, warehouse)
go

create unique clustered index d_clu
        on district(d_w_id, d_id)
        on Scache
go
dbcc tune(indextrips, 100, district)
go

select getdate()
go
create unique nonclustered index c_non1
        on customer(c_w_id, c_d_id, c_last, c_first, c_id)
```

```
        on Scust
go

checkpoint
go
EOF
########################################
# tpcc_tables.sh
########################################
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident “@(#)tpcc_tables.sh1.196/01/09SMI”
#

isql -Usa -P$PASSWORD -e << EOF
/* This script will create all the tables required for TPC-C benchmark */
/* It will also create some of the indexes. */
sp_dboption tpcc,”select into/bulkcopy”,true
go
use tpcc
go
checkpoint
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_1char(20),
        w_street_2char(20),
        w_city        char(20),
        w_state       char(2),
        w_zip         char(9),
        w_tax         real,
        w_ytd         float         /*- Updated by PID, PNM */
) with max_rows_per_page = 1 on Scache
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_1char(20),
        d_street_2char(20),
        d_city        char(20),
        d_state       char(2),
        d_zip         char(9),
        d_tax         real,
        d_ytd         float,        /*- Updated by PID, PNM */
        d_next_o_idint  /*- Updated by NO */
) with max_rows_per_page = 7 on Scache
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        float,
        i_data         char(50)
) with max_rows_per_page = 23 on Scache
go
create unique clustered index i_clu
        on item(i_id)
        with fillfactor = 100 on Scache
go
dbcc tune(indextrips, 10, item)
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 Scache
go
create unique clustered index no_clu
        on new_order(no_w_id, no_d_id, no_o_id)
        on Scache
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips, 100, new_order)
go

sp_helpsegment Scache
go

if exists ( select name from sysobjects where name = ‘history’ )
        drop table history
go
create table history (
        h_c_id         int,
        h_c_d_idtinyint,
        h_c_w_idsmallint,
        h_d_id         tinyint,
        h_w_id         smallint,
        h_date         datetime,
        h_amountfloat,
        h_data         char(24)
) on Shist
go
alter table history partition 128
go

sp_helpsegment Shist
go

if exists ( select name from sysobjects where name = ‘orders’ )
        drop table orders
go
create table orders (
        o_id           int,
        o_c_id         int,
        o_d_id         tinyint,
        o_w_id         smallint,
        o_entry_ddatetime,
        o_carrier_idsmallint,/*- Updated by D */
        o_ol_cnttinyint,
        o_all_localtinyint
) on Sorders
go
create unique clustered index o_clu
        on orders(o_w_id, o_d_id, o_id)
        on Sorders
go
dbcc tune(ascinserts, 1, orders)
go
dbcc tune(oamtrips, 100, orders)
go

sp_helpsegment Sorders
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_numbertinyint,
        ol_i_id        int,
        ol_supply_w_idsmallint,
        ol_delivery_ddatetime,/*- Updated by D */
        ol_quantitysmallint,
        ol_amountfloat,
        ol_dist_infochar(24)
) on Soline
go
create unique clustered index ol_clu
        on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
        on Soline
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go

sp_helpsegment Soline
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_middlechar(2),
        c_last         char(16),
        c_street_1char(20),
        c_street_2char(20),
        c_city         char(20),
        c_state        char(2),
        c_zip          char(9),
        c_phone        char(16),
        c_since      datetime,
        c_creditchar(2),
        c_credit_limnumeric(12,2),
        c_discountreal,
        c_delivery_cntsmallint,
        c_payment_cntsmallint,/*- Updated by PNM, PID */
        c_balancefloat,    /*- Updated by PNM, PID */
        c_ytd_paymentfloat,/*- Updated by PNM, PID */
        c_data1      char(250),/*- Updated (?) by PNM, PID */
        c_data2      char(250)/*- Updated (?) by PNM, PID */
) on Scust
go
create unique clustered index c_clu
        on customer(c_w_id, c_id, c_d_id)
        on Scust
go

sp_helpsegment Scust
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_quantitysmallint,/*- Updated by NO */
        s_ytd         int,             /*- Updated by NO */
        s_order_cntsmallint,/*- Updated by NO */
        s_remote_cntsmallint,/*- Updated by NO */
        s_dist_01char(24),
        s_dist_02char(24),
        s_dist_03char(24),
        s_dist_04char(24),
        s_dist_05char(24),
        s_dist_06char(24),
        s_dist_07char(24),
        s_dist_08char(24),
        s_dist_09char(24),
        s_dist_10char(24),
        s_data       char(50)
) with max_rows_per_page = 5 on Sstock
go
create unique clustered index s_clu
        on stock(s_i_id, s_w_id)
        on Sstock
go
dbcc tune(indextrips, 10, stock)
go
checkpoint
go
EOF



******
BEGIN erro.c  HERE:

******

##################################
#       error.c #
##################################
```

```
#define BLK_SLOW                    43
#define BLK_SLOW_LVL           CLN_NOT_ASSIGNED



#if ! lint
static char *sddsId = "@(#)  error.c  1.1  4/30/91  19:47:32";
#endif /* ! lint */

/*
** Confidential property of Sybase, Inc.
** (c) Copyright Sybase, Inc. 1991
** All rights reserved
*/

/* Required standard include files */
#include <stdio.h>
#ifdef _NTINTEL
#include <stdlib.h>
#include <windows.h>
#endif

/* Required Sybase include files */
#include <sybfront.h>
#include <sybdb.h>

/* message numbers that we don't want to deal with */
#defineDUMB_MESSAGE5701
#defineABORT_ERROR6104


#ifdef _NTINTEL
int
 err_handler(dbproc, severity, errno, oserr, errstr, oserrstr)
  DBPROCESS *dbproc;
  int severity;
  int errno;
  int oserr;
  char*errstr;
  char*oserrstr;
{

        /* changing databases message */
        if (errno == DUMB_MESSAGE || errno == ABORT_ERROR)
                return(INT_CANCEL);

        fprintf(stderr,"DB-LIBRARY Error: \n\t%s\n",errstr);

        if (oserr != DBNOERR)
                fprintf(stderr,"O/S Error: \n\t%s\n",oserrstr);

        /* exit on any error */
        exit(-100);
}
#else
int
err_handler(dbproc, severity, errno, oserr)
  DBPROCESS *dbproc;
  int severity;
  int errno;
  int oserr;
{
        /* changing databases message */
        if (errno == DUMB_MESSAGE || errno == ABORT_ERROR)
                return(INT_CANCEL);

        fprintf(stderr,"DB-LIBRARY Error: \n\t%s\n",dberrstr(errno));

        if (oserr != DBNOERR)
                fprintf(stderr,"O/S Error: \n\t%s\n",dboserrstr(oserr));

        /* exit on any error */
        exit(-100);
}
#endif


int
msg_handler(dbproc,msgno,msgstate,severity,msgtext,servername,pr
ocname,line)
DBPROCESS*dbproc;
int             msgno;
int             msgstate;
int             severity;
char            *msgtext;
char              *servername;
char              *procname;
int              line;
{

        /* changing database messages */
        if (msgno == DUMB_MESSAGE || msgno == ABORT_ERROR
|| msgno == 5703 || msgno == 5704 || msgno == 4843)
                return(SUCCEED);

        /* Is this a deadlock message */
        if (msgno == 1205)
        {
                /* Set the deadlock indicator */
                *((DBBOOL *) dbgetuserdata(dbproc)) = TRUE;

                /* Sleep a few seconds before going back */
#ifdef _NTINTEL
                Sleep((DWORD) 2000);
#else
                sleep((unsigned) 2);
#endif
                return(SUCCEED);

        }

        fprintf(stderr, "msg no %d -\n%s", msgno, msgtext);

        /* exit on any error */
        exit(-101);
}
##################################
#       load.c #
##################################
typedefunsigned longBitVector;
#define WSZ(sizeof(BitVector)*8)
#ifndef WAREBATCH
#defineWAREBATCH1300
#endif
#definenthbit(map,n)map[(n)/WSZ] &  (((BitVector)0x1)<<
((n)%WSZ))
```

```
#definesetbit(map,n)map[(n)/WSZ] |= (((BitVector)0x1)<< ((n)%WSZ))

/**********************************************************
Load TPCC tables
**********************************************************/
#include "stdio.h"
#include "string.h"
#include "loader.h"

int load_item;
int load_warehouse;
int load_district;
int load_history;
int load_orders;
int load_new_order;
int load_order_line;
int load_customer;
int load_stock;
ID  w1, w2;
ID  warehouse;
int batch_size = 1000;
char   password[10];

int main(argn, argv)
        int argn;
        char **argv;
{

        dbsetversion(DBVERSION_100);

        getargs(argn, argv);
        Randomize();

        if (load_item)LoadItems();
        if (load_warehouse)LoadWarehouse(w1, w2);
        if (load_district)LoadDistrict(w1, w2);
        if (load_history)LoadHist(w1, w2);
        if (load_customer)LoadCustomer(w1, w2);
        if (load_stock)LoadStock(w1, w2);
        if (load_orders)LoadOrd(w1, w2);
        if (load_new_order)LoadNew(w1, w2);
        return 0;
}



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

Warehouse

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


ID   w_id;
TEXT w_name[10+1];
TEXT w_street_1[20+1];
TEXT w_street_2[20+1];
TEXT w_city[20+1];
TEXT w_state[2+1];
TEXT w_zip[9+1];
FLOAT w_tax;
```

```
MONEY w_ytd;


int bulk_w;

LoadWarehouse(w1, w2)
        ID w1, w2;
{

        begin_warehouse_load();
        for (warehouse=w1; warehouse<=w2; warehouse++)
        {
                printf(“Loading warehouse for warehouse %d\n”,
warehouse);

                w_id = warehouse;
                MakeAlphaString(6, 10, w_name);
                MakeAddress(w_street_1, w_street_2, w_city, w_state,
w_zip);

                w_tax = RandomNumber(0, 2000) / 10000.0;
                w_ytd = 300000.00;

                warehouse_load();

                printf(“loaded warehouse for warehouse %d\n”,
warehouse);
        }
        end_warehouse_load();
        return;
}


begin_warehouse_load()
{
        int     i = 1;

        bulk_w = bulk_open(“tpcc”, “warehouse”, password);

        bulk_bind(bulk_w, i++, “w_id”,   &w_id, ID_T);
        bulk_bind(bulk_w, i++, “w_name”, w_name, TEXT_T);
        bulk_bind(bulk_w, i++, “w_street_1”, w_street_1, TEXT_T);
        bulk_bind(bulk_w, i++, “w_street_2”, w_street_2, TEXT_T);
        bulk_bind(bulk_w, i++, “w_city”, w_city, TEXT_T);
        bulk_bind(bulk_w, i++, “w_state”, w_state, TEXT_T);
        bulk_bind(bulk_w, i++, “w_zip”, w_zip, TEXT_T);
        bulk_bind(bulk_w, i++, “w_tax”, &w_tax, FLOAT_T);
        bulk_bind(bulk_w, i++, “w_ytd”, &w_ytd, MONEY_T);
}


warehouse_load()
{
        debug(“Loading Warehouse %d\n”, w_id);
        bulk_load(bulk_w);
}


end_warehouse_load()
{
        bulk_close(bulk_w);
}
```

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

District

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



ID d_id;
ID d_w_id;
TEXT d_name[10+1];
TEXT d_street_1[20+1];
TEXT d_street_2[20+1];
TEXT d_city[20+1];
TEXT d_state[2+1];
TEXT d_zip[9+1];
FLOAT d_tax;
MONEY d_ytd;
ID    d_next_o_id;

int bulk_d;

LoadDistrict(w1, w2)
        ID w1, w2;
{
        ID w_id;

        begin_district_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {
                printf(“Loading districts for warehouse %d\n”, w_id);

                d_w_id = w_id;
                d_ytd = 30000.00;
                d_next_o_id = 3001;

                for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
                {

                        MakeAlphaString(6, 10, d_name);
                        MakeAddress(d_street_1, d_street_2, d_city, d_state, d_zip);
                        d_tax = RandomNumber(0, 2000) / 10000.0;

                        district_load();
                }
                printf(“loaded district for warehouse %d\n”, w_id);
        }
        end_district_load();
        return;
}



begin_district_load()
{
        int     i = 1;
        bulk_d = bulk_open(“tpcc”, “district”, password);

        bulk_bind(bulk_d, i++, “d_id”, &d_id, ID_T);
        bulk_bind(bulk_d, i++, “d_w_id”, &d_w_id, ID_T);
        bulk_bind(bulk_d, i++, “d_name”, d_name, TEXT_T);
        bulk_bind(bulk_d, i++, “d_street_1”, d_street_1, TEXT_T);
        bulk_bind(bulk_d, i++, “d_street_2”, d_street_2, TEXT_T);
        bulk_bind(bulk_d, i++, “d_city”, d_city, TEXT_T);
        bulk_bind(bulk_d, i++, “d_state”, d_state, TEXT_T);
        bulk_bind(bulk_d, i++, “d_zip”, d_zip, TEXT_T);
        bulk_bind(bulk_d, i++, “d_tax”, &d_tax, FLOAT_T);
        bulk_bind(bulk_d, i++, “d_ytd”, &d_ytd, MONEY_T);
        bulk_bind(bulk_d, i++, “d_next_o_id”, &d_next_o_id, ID_T);
}

district_load()
{
        debug(“District %d  w_id=%d\n”, d_id, d_w_id);
        bulk_load(bulk_d);
}

end_district_load()
{
        bulk_close(bulk_d);
}



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

Item

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



ID i_id;
ID i_im_id;
TEXT i_name[24+1];
MONEY i_price;
TEXT  i_data[50+1];

int bulk_i;

LoadItems()
{

        int perm[MAXITEMS+1];
        int i, r, t;

        printf(“Loading items\n”);

        begin_item_load();

        /* select exactly 10% of items to be labeled “original” */
        RandomPermutation(perm, MAXITEMS);

        /* do for each item */
        for (i_id=1;   i_id <= MAXITEMS; i_id++)
        {

        /* Generate Item Data */
        MakeAlphaString(14, 24, i_name);
        i_price = RandomNumber(100,10000)  /  100.0;
        MakeAlphaString(26, 50, i_data);
        if (perm[i_id] <= (MAXITEMS+9)/10)
                Original(i_data);

        /* Generate i_im_id for V 3.0 */
        i_im_id = RandomNumber(1, 10000);

         item_load();
         }

        end_item_load();
        return;
}



begin_item_load()
{
        int     i = 1;

        bulk_i = bulk_open(“tpcc”, “item”, password);

        bulk_bind(bulk_i, i++, “i_id”, &i_id, ID_T);
        bulk_bind(bulk_i, i++, “i_im_id”, &i_im_id, ID_T);
        bulk_bind(bulk_i, i++, “i_name”, i_name, TEXT_T);
        bulk_bind(bulk_i, i++, “i_price”, &i_price, MONEY_T);
        bulk_bind(bulk_i, i++, “i_data”, i_data, TEXT_T);
}



item_load()
{
        debug(“i_id=%3d price=%5.2f data=%s\n”,
           i_id, i_price, i_data);
        bulk_load(bulk_i);
}


end_item_load()
{
        bulk_close(bulk_i);
}



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

History

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

```
ID h_c_id;
ID h_c_d_id;
ID h_c_w_id;
ID h_d_id;
ID h_w_id;
DATE h_date;
MONEY h_amount;
TEXT h_data[24+1];

int bulk_h;

LoadHist(w1, w2)
        ID w1, w2;
{
        ID w_id;
        ID d_id, c_id;

        begin_history_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {

                for (d_id=1;  d_id <= DIST_PER_WARE; d_id++)
                {
                        for (c_id=1; c_id <= CUST_PER_DIST; c_id++)
                                LoadCustHist(w_id, d_id, c_id);
                }

                printf(“\nLoaded history for warehouse %d\n”, w_id);
        }
        end_history_load();
}


LoadCustHist(w_id, d_id, c_id)
        ID w_id, d_id, c_id;
{

        h_c_id = c_id;
        h_c_d_id = d_id;
        h_c_w_id = w_id;
        h_d_id = d_id;
        h_w_id = w_id;
        h_amount = 10.0;
        MakeAlphaString(12, 24,h_data);
        datetime(&h_date);
        history_load();
}



begin_history_load()
{
        int     i = 1;

        bulk_h = bulk_open(“tpcc”, “history”, password);

        bulk_bind(bulk_h, i++, “h_c_id”,   &h_c_id,   ID_T);
        bulk_bind(bulk_h, i++, “h_c_d_id”, &h_c_d_id, ID_T);
        bulk_bind(bulk_h, i++, “h_c_w_id”, &h_c_w_id, ID_T);
        bulk_bind(bulk_h, i++, “h_d_id”,   &h_d_id,   ID_T);
        bulk_bind(bulk_h, i++, “h_w_id”,   &h_w_id,   ID_T);
        bulk_bind(bulk_h, i++, “h_date”,   &h_date,   DATE_T);
        bulk_bind(bulk_h, i++, “h_amount”, &h_amount, MONEY_T);
        bulk_bind(bulk_h, i++, “h_data”,   h_data,    TEXT_T);
}

history_load()
{

        debug(“h_c_id=%d  h_amount=%g\n”, h_c_id, h_amount);
        bulk_load(bulk_h);
}


end_history_load()
{
        bulk_close(bulk_h);
}




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

Customer

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

/* static variables containing fields for customer record */
ID c_id;
ID c_d_id;
ID c_w_id;
TEXT c_first[16+1];
TEXT c_middle[2+1] = “OE”;
TEXT c_last[16+1];
TEXT c_street_1[20+1];
TEXT c_street_2[20+1];
TEXT c_city[20+1];
TEXT c_state[2+1];
TEXT c_zip[9+1];
TEXT c_phone[16+1];
DATE c_since;
TEXT c_credit[2+1] = “?C”;
MONEY c_credit_lim = 50000.0;
FLOAT c_discount;
MONEY c_balance = -10.0;
MONEY c_ytd_payment = 10.0;
COUNT c_payment_cnt = 1;
COUNT c_delivery_cnt = 0;
TEXT c_data[500+1];
TEXT c_data1[250+1];
TEXT c_data2[250+1];
ID     len;

int bulk_c;


LoadCustomer(w1, w2)
        ID w1, w2;
{
        ID w_id;
        begin_customer_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {
                Customer(w_id);
                printf(“\nLoaded customer for warehouse %d\n”, w_id);
        }
        end_customer_load();
}

Customer(w_id)
        int w_id;
{
        BitVector badcredit[DIST_PER_WARE][(3000+WSZ-1)/WSZ], *
bmp;
        int i, j;
        ID d_id;

        /* Mark exactly 10% of customers as having bad credit */
        for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
        {
                bmp = badcredit[d_id-1];
                for (i=0; i<(3000+WSZ-1)/WSZ; i++)
                        bmp[i] = (BitVector)0x0000;
                for (i=0; i<(3000+9)/10; i++)
                {
                        do {
                                j = RandomNumber(0,3000-1);
                        } while (nthbit(bmp,j));
                        setbit(bmp,j);
                }
        }

        c_w_id = w_id;
        for (i=0; i<CUST_PER_DIST; i++)
        {
                c_id = i+1;
                for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
                {
                        c_d_id = d_id;

LastName(i<1000?i:NURandomNumber(255,NURAND_C,0,999),c_la
st);
                        MakeAlphaString(8, 16, c_first);

MakeAddress(c_street_1,c_street_2,c_city,c_state,c_zip);
                        MakeNumberString(16, 16, c_phone);
                        MakeAlphaString(300, 500, c_data);
                        datetime(&c_since);
                        c_credit[0] = nthbit(badcredit[d_id-1],i) ? ‘B’ : ‘G’;
                        c_discount = RandomNumber(0, 5000)  /  10000.0;
                        c_balance = -10.0;
                        customer_load();
                }
        }
}

begin_customer_load()
{
        int     i = 1;

        bulk_c = bulk_open(“tpcc”, “customer”, password);

        bulk_bind(bulk_c, i++, “c_id”,  &c_id,   ID_T);
        bulk_bind(bulk_c, i++, “c_d_id”,&c_d_id, ID_T);
        bulk_bind(bulk_c, i++, “c_w_id”,&c_w_id, ID_T);
        bulk_bind(bulk_c, i++, “c_first”,    c_first, TEXT_T);
        bulk_bind(bulk_c, i++, “c_middle”,  c_middle, TEXT_T);
        bulk_bind(bulk_c, i++, “c_last”,c_last,    TEXT_T);
        bulk_bind(bulk_c, i++, “street_1”,  c_street_1 , TEXT_T);
        bulk_bind(bulk_c, i++, “street_2”,  c_street_2,  TEXT_T);
        bulk_bind(bulk_c, i++, “c_city”,c_city,  TEXT_T);
        bulk_bind(bulk_c, i++, “c_state”,    c_state, TEXT_T);
        bulk_bind(bulk_c, i++, “c_zip”, c_zip,    TEXT_T);
        bulk_bind(bulk_c, i++, “c_phone”,    c_phone, TEXT_T);
        bulk_bind(bulk_c, i++, “c_since”,    &c_since, DATE_T);
        bulk_bind(bulk_c, i++, “c_credit”,  c_credit,TEXT_T);
        bulk_bind(bulk_c, i++, “c_credit_lim”, &c_credit_lim, MONEY_T);
        bulk_bind(bulk_c, i++, “c_discount”, &c_discount, FLOAT_T);
        bulk_bind(bulk_c, i++, “c_delivery_cnt”, &c_delivery_cnt, COUNT_T);
        bulk_bind(bulk_c, i++, “c_payment_cnt”,&c_payment_cnt, COUNT_T);
        bulk_bind(bulk_c, i++, “c_balance”, &c_balance, MONEY_T);
        bulk_bind(bulk_c, i++, “c_ytd_payment”, &c_ytd_payment, MONEY_T);
        bulk_bind(bulk_c, i++, “c_data_1”, c_data1, TEXT_T);
        bulk_bind(bulk_c, i++, “c_data_2”, c_data2, TEXT_T);
}

customer_load()
{
        debug(“c_id=%-5d  d_id=%-5d  w_id=%-5d  c_last=%s\n”,
          c_id,c_d_id,  c_w_id,  c_last);


        /* Break the string c_data into 2 pieces */
        len = strlen(c_data);
        if (len > 250)
                {
                memcpy(c_data1, c_data, 250);
                c_data1[250]=’\0’;
                memcpy(c_data2, c_data+250, len-250 +1);
                }

        else
                {
                memcpy(c_data1, c_data, 250+1);
                strcpy(c_data2,””);
                }

        /* load the data */
        bulk_load(bulk_c);
}

end_customer_load()
{
        bulk_close(bulk_c);
}



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

Order, Order line, New order

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

```
/* Order row */
ID o_id;
ID o_c_id;
ID o_d_id;
ID o_w_id;
DATE o_entry_d;
ID o_carrier_id;
COUNT o_ol_cnt;
LOGICAL o_all_local;

/* Order line row */
ID   ol_o_id;
ID   ol_d_id;
ID   ol_w_id;
ID   ol_number;
ID   ol_i_id;
ID   ol_supply_w_id;
DATE ol_delivery_d;
COUNT ol_quantity;
MONEY ol_amount;
TEXT  ol_dist_info[24+1];

/* new order row */
ID no_o_id;
ID no_d_id;
ID no_w_id;


int  o_bulk;
int ol_bulk;
int no_bulk;



LoadOrd(w1, w2)
        ID w1, w2;
{
        ID w_id;
        ID d_id;

        begin_order_load();
        begin_order_line_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {

                for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
                        Orders(w_id, d_id);

                printf(“\nLoaded order + order_line for warehouse
%d\n”, w_id);
        }
        end_order_line_load();
        end_order_load();
}

LoadNew(w1, w2)
        ID w1, w2;
{
        ID w_id;
        ID d_id;

        begin_new_order_load();
```

```
        for (w_id=w1; w_id<=w2; w_id++)
        {
                for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
                {
                        no_d_id = d_id;
                        no_w_id = w_id;
                        for (no_o_id=2101;  no_o_id <= ORD_PER_DIST; no_o_id++)
                                new_order_load();
                }
                printf(“\nLoaded new_order for warehouse %d\n”, w_id);
        }
        end_new_order_load();
}

Orders(w_id, d_id)
        ID w_id;
        ID d_id;
{
        int cust[ORD_PER_DIST+1];
        int ol_cnt[ORD_PER_DIST+1], sum;
        ID ol;

        printf(“\nLoading orders and order lines  for warehouse %d district %d\n”,
         w_id, d_id);

        RandomPermutation(cust, ORD_PER_DIST);


        for (o_id = 1, sum=0;  o_id <= ORD_PER_DIST;  o_id++)
                sum += (ol_cnt[o_id] = RandomNumber(5, 15));

        while (sum > 10*ORD_PER_DIST)
        {
                do {
                        o_id = RandomNumber(1,ORD_PER_DIST);
                } while (ol_cnt[o_id]==5);
                ol_cnt[o_id]--;
                sum--;
        }

        while (sum < 10*ORD_PER_DIST)
        {
                do {
                        o_id = RandomNumber(1,ORD_PER_DIST);
                } while (ol_cnt[o_id]==15);
                ol_cnt[o_id]++;
                sum++;
        }

        for (o_id = 1;  o_id <= ORD_PER_DIST;  o_id++)
        {
        o_c_id = cust[o_id];
        o_d_id = d_id;
        o_w_id = w_id;
        datetime(&o_entry_d);
        if (o_id <= 2100)
        o_carrier_id = RandomNumber(1,10);
        else o_carrier_id = -1;
        o_ol_cnt = ol_cnt[o_id];
        o_all_local = 1;
        order_load();

        for (ol=1;  ol<=o_ol_cnt; ol++)
```

```
                OrderLine(ol);

        }
}




OrderLine(ol)
        ID ol;
{

        ol_o_id = o_id;
        ol_d_id = o_d_id;
        ol_w_id = o_w_id;
        ol_number = ol;
        ol_i_id = RandomNumber(1, MAXITEMS);
        ol_supply_w_id = o_w_id;

  if (o_id <= 2100)
        ol_delivery_d = o_entry_d;
  else
  {
        ol_delivery_d.x[0] = 0;
        ol_delivery_d.x[1] = 0;
  }



        ol_quantity = 5;
        if (o_id <= 2100) ol_amount = 0;
        else            ol_amount = RandomNumber(1, 999999) / 100.0;
        MakeAlphaString(24, 24, ol_dist_info);
        order_line_load();
}


NewOrder(w_id, d_id)
        ID w_id, d_id;
{

        no_d_id = o_d_id;
        no_w_id = o_w_id;
        for (no_o_id=2101;  no_o_id <= ORD_PER_DIST; no_o_id++)
        new_order_load();
}




begin_order_load()
{
        int     i = 1;

        o_bulk = bulk_open(“tpcc”, “orders”, password);

        bulk_bind(o_bulk, i++, “o_id”, &o_id, ID_T);
        bulk_bind(o_bulk, i++, “o_c_id”, &o_c_id, ID_T);
        bulk_bind(o_bulk, i++, “o_d_id”, &o_d_id, ID_T);
        bulk_bind(o_bulk, i++, “o_w_id”, &o_w_id, ID_T);
        bulk_bind(o_bulk, i++, “o_entry_d”, &o_entry_d, DATE_T);
        bulk_bind(o_bulk, i++, “o_carrier_id”, &o_carrier_id, ID_T);
        bulk_bind(o_ol_cnt, i++, “o_ol_cnt”, &o_ol_cnt, COUNT_T);
```

```
        bulk_bind(o_all_local, i++, “o_all_local”, &o_all_local, LOGICAL_T);
}

order_load()
{
        debug(“o_id=%d o_c_id=%d  count=%d\n”, o_id, o_c_id, o_ol_cnt);
        bulk_load(o_bulk);
}

end_order_load()
{
        bulk_close(o_bulk);
}




begin_order_line_load()
{
        int     i = 1;

        ol_bulk = bulk_open(“tpcc”, “order_line”, password);

        bulk_bind(ol_bulk, i++, “ol_o_id”, &ol_o_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_d_id”, &ol_d_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_w_id”, &ol_w_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_number”, &ol_number, ID_T);
        bulk_bind(ol_bulk, i++, “ol_i_id”, &ol_i_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_supply_w_id”, &ol_supply_w_id, ID_T);
        bulk_bind(ol_bulk, i++, “ol_delivery_d”, &ol_delivery_d, DATE_T);
        bulk_bind(ol_bulk, i++, “ol_quantity”, &ol_quantity, COUNT_T);
        bulk_bind(ol_bulk, i++, “ol_amount”, &ol_amount, MONEY_T);
        bulk_bind(ol_bulk, i++, “ol_dist_info”, ol_dist_info, TEXT_T);
}

order_line_load()
{
        static int ol_count = 0;
        debug(“  ol_o_id=%d  ol_number=%d  ol_amount=%g\n”,
                ol_o_id,ol_number,ol_amount);
        bulk_load(ol_bulk);
}


end_order_line_load()
{
        bulk_close(ol_bulk);
}

begin_new_order_load()
{
        int     i = 1;

        no_bulk = bulk_open(“tpcc”, “new_order”, password);

        bulk_bind(no_bulk, i++, “no_o_id”, &no_o_id, ID_T);
        bulk_bind(no_bulk, i++, “no_d_id”, &no_d_id, ID_T);
        bulk_bind(no_bulk, i++, “no_w_id”, &no_w_id, ID_T);
}

new_order_load()
{
        debug(“   no_o_id=%d   \n”, no_o_id);
```

```
    bulk_load(no_bulk);
}


end_new_order_load()
{
    bulk_close(no_bulk);
}


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

Stock

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



ID s_i_id;
ID s_w_id;
COUNT s_quantity;
TEXT s_dist_01[24+1];
TEXT s_dist_02[24+1];
TEXT s_dist_03[24+1];
TEXT s_dist_04[24+1];
TEXT s_dist_05[24+1];
TEXT s_dist_06[24+1];
TEXT s_dist_07[24+1];
TEXT s_dist_08[24+1];
TEXT s_dist_09[24+1];
TEXT s_dist_10[24+1];
COUNT s_ytd;
COUNT s_order_cnt;
COUNT s_remote_cnt;
TEXT s_data[50+1];

int bulk_s;

/*
** On loading stock in major order of item_id:
** 10% of the MAXITEMS items in each warehouse need to marked as
original
** (i.e., s_data like '%ORIGINAL%'.)  This is a bit harder to do when
we
** load by item number, rather than by warehouses.  The trick is to
first
** generate a huge WAREBATCH * MAXITEMS bitmap, initialize all
bits to zero,
** and then set 10% of bits in each row to 1.  While loading item i in
** warehouse w, we simply lookup bitmap[w][i] to see whether it
needs to
** be marked as original.
*/

LoadStock(w1, w2)
    ID w1, w2;
{
    ID w_id;

    BitVector original[WAREBATCH][((MAXITEMS+(WSZ-
1))/WSZ)], * bmp;
    int w, i, j;

    if (w2-w1+1 > WAREBATCH)
    {
        fprintf(stderr, "Can't load stock for %d warehouses.\n",
            w2-w1+1);
        fprintf(stderr, "Please use batches of %d.\n", WAREBATCH);
    }

    for (w=w1; w<=w2; w++)
    {
        bmp = original[w-w1];
        /* Mark all items as not "original" */
        for (i=0; i<(MAXITEMS+(WSZ-1))/WSZ; i++)
            bmp[i] = (BitVector)0x0000;
        /* Mark exactly 10% of items as "original" */
        for (i=0; i<(MAXITEMS+9)/10; i++)
        {
            do {
                j = RandomNumber(0,MAXITEMS-1);
            } while (nthbit(bmp,j));
            setbit(bmp,j);
        }
    }

    printf("Loading stock for warehouse %d to %d.\n", w1, w2);
    begin_stock_load();
    /* do for each item */
    for (s_i_id=1;   s_i_id <= MAXITEMS; s_i_id++)
    {
        for (w_id=w1; w_id<=w2; w_id++)
        {
        /* Generate Stock Data */
            s_w_id = w_id;
            s_quantity = RandomNumber(10,100);
            MakeAlphaString(24, 24, s_dist_01);
            MakeAlphaString(24, 24, s_dist_02);
            MakeAlphaString(24, 24, s_dist_03);
            MakeAlphaString(24, 24, s_dist_04);
            MakeAlphaString(24, 24, s_dist_05);
            MakeAlphaString(24, 24, s_dist_06);
            MakeAlphaString(24, 24, s_dist_07);
            MakeAlphaString(24, 24, s_dist_08);
            MakeAlphaString(24, 24, s_dist_09);
            MakeAlphaString(24, 24, s_dist_10);
            s_ytd = 0;
            s_order_cnt = 0;
            s_remote_cnt = 0;
            MakeAlphaString(26, 50, s_data);
            if (nthbit(original[w_id-w1],s_i_id-1))
            {
                Original(s_data);
            }
            stock_load();
        }
    }
    end_stock_load();
    printf("\nLoaded stock for warehouses %d to %d.\n", w1, w2);
}

begin_stock_load()
{
    int    i = 1;
```

```
        bulk_s = bulk_open(“tpcc”, “stock”, password);

        bulk_bind(bulk_s, i++, “s_i_id”, &s_i_id, ID_T);
        bulk_bind(bulk_s, i++, “s_w_id”, &s_w_id, ID_T);
        bulk_bind(bulk_s, i++, “s_quantity”, &s_quantity, COUNT_T);
        bulk_bind(bulk_s, i++, “s_ytd”, &s_ytd,COUNT_T);
        bulk_bind(bulk_s, i++, “s_order_cnt”, &s_order_cnt,
COUNT_T);
        bulk_bind(bulk_s, i++, “s_remote_cnt”, &s_remote_cnt,
COUNT_T);
        bulk_bind(bulk_s, i++, “s_dist_01”, s_dist_01, TEXT_T);
        bulk_bind(bulk_s, i++, “s_dist_02”, s_dist_02, TEXT_T);
        bulk_bind(bulk_s, i++, “s_dist_03”, s_dist_03, TEXT_T);
        bulk_bind(bulk_s, i++, “s_dist_04”, s_dist_04, TEXT_T);
        bulk_bind(bulk_s, i++, “s_dist_05”, s_dist_05, TEXT_T);
        bulk_bind(bulk_s, i++, “s_dist_06”, s_dist_06, TEXT_T);
        bulk_bind(bulk_s, i++, “s_dist_07”, s_dist_07, TEXT_T);
        bulk_bind(bulk_s, i++, “s_dist_08”, s_dist_08, TEXT_T);
        bulk_bind(bulk_s, i++, “s_dist_09”, s_dist_09, TEXT_T);
        bulk_bind(bulk_s, i++, “s_dist_10”, s_dist_10, TEXT_T);
        bulk_bind(bulk_s, i++, “s_data”,s_data, TEXT_T);
}



stock_load()
{
        debug(“s_i_id=%d w_id=%d s_data=%s\n”,
           s_i_id, s_w_id,   s_data);
        bulk_load(bulk_s);
}


end_stock_load()
{
        bulk_close(bulk_s);
}



test(){}


getargs(argc, argv)

/*****************************************************************
configure configures the load stuff
   By default, loads all the tables for a the specified warehouse.
                   When loading warehouse 1, also loads the item table.
*******************************************************************/
        int argc;
        char **argv;
{
        char ch;

        /* define the defaults */
        load_item = load_warehouse = load_district = load_history =
        load_orders = load_new_order = load_order_line =
        load_customer = load_stock = NO;
```

```
        if       (strcmp(argv[1], “warehouse”) == 0) load_warehouse = YES;
        else if (strcmp(argv[1], “district”) == 0)  load_district = YES;
        else if (strcmp(argv[1], “stock”) == 0)  load_stock = YES;
        else if (strcmp(argv[1], “item”) == 0)  load_item = YES;
        else if (strcmp(argv[1], “history”) == 0)  load_history = YES;
        else if (strcmp(argv[1], “orders”) == 0)  load_orders = YES;
        else if (strcmp(argv[1], “customer”) == 0)  load_customer = YES;
        else if (strcmp(argv[1], “new_order”) ==0) load_new_order = YES;
        else
        {
                 printf(“%s is not a valid table name\n”, argv[1]);
                 exit(0);

        }

        /* Set the w1 and w2 to argv[2] and arg[3] */
        if (argc < 3)
        {
                printf(“Usage: %s <table> <w_first> [<w_last>]\n”, argv[0]);
                exit(1);
        }
        {
                w1 = atoi(argv[2]);
                if (argc >= 3)
                        w2 = atoi(argv[3]);
                else
                        w2 = w1;
        }

        /* Get the password for sa */
        if (argc > 4)
        strcpy(password,argv[4]);

        /* Check if warehouse is within the range */
        if (w1 <= 0 || w1 > w2)
        {
                printf(“Warehouse id is out of range\n”);
                exit(0);
        }
}

double drand48();

MakeAddress(str1, str2, city, state, zip)
        TEXT str1[20+1];
        TEXT str2[20+1];
        TEXT city[20+1];
        TEXT state[2+1];
        TEXT zip[9+1];
{
        MakeAlphaString(10,20,str1);
        MakeAlphaString(10,20,str2);
        MakeAlphaString(10,20,city);
        MakeAlphaString(2,2,state);
        MakezipString(0,9999,zip);

        /* Changed for TPCC V 3.0 */
        strcat(zip, “11111”);

}
```

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

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


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

        length = RandomNumber(min, max);

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

        return length;
}

int MakezipString(min, max, num)
        int min;
        int max;
        TEXT num[];
{
        static char digit[]=”0123456789”;
        int length;
        int i;

        length = 4;

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

        return length;
}



int MakeAlphaString(min, max, str)
        int min;
        int max;
        TEXT str[];
{
        static char character[] =
“abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
”;
        int length;
        int i;

        length = RandomNumber(min, max);

        for (i=0;  i<length;  i++)
        str[i] = character[RandomNumber(0, sizeof(character)-2)];
        str[length] = ‘\0’;

        return length;
}


Original(str)
        TEXT str[];
{
        int pos;
        int len;

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

        pos = RandomNumber(0,len-8);

        str[pos+0] = ‘O’;
        str[pos+1] = ‘R’;
        str[pos+2] = ‘I’;
        str[pos+3] = ‘G’;
        str[pos+4] = ‘I’;
        str[pos+5] = ‘N’;
        str[pos+6] = ‘A’;
        str[pos+7] = ‘L’;
}



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

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

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


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



int RandomNumber(min, max)
        int min;
        int max;
{
        int r;
        r = (int)(drand48() * (max - min + 1)) + min;
        return r;
}


int NURandomNumber(a, c, min, max)
        int a;
        int c;
        int min;
        int max;
{
        int r;

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

        return r;
}
####################################
#       bulk_sybase.c #
####################################
/****************************************************************
*****************************************************************

Sybase Specific Routines

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

#include <stdio.h>
#include <sys/time.h>
#include <string.h>
#include “loader.h”

datetime(date)
        DBDATETIME *date;
{
        struct timeval time;
        gettimeofday(&time, NULL);
        date->dtdays = time.tv_sec / (60*60*24)
                + (1970-1900)*365 + (1970-1900)/4;
        date->dttime = (time.tv_sec % (60*60*24))*300
                + time.tv_usec*300/1000000;
}


/* define the type information for each field */
typedef struct
{
        char *terminator;
        int termlen;
        int type;
} bind_parm;
```

```
bind_parm parm[MAX_T] =
{
        /* COUNT */{NULL, 0, SYBINT4},
        /* ID*/     {NULL, 0, SYBINT4},
        /* MONEY */{NULL, 0, SYBFLT8},
        /* FLOAT */{NULL, 0, SYBFLT8},
        /* TEXT */{““,   1, SYBCHAR},
        /* DATE */{NULL, 0, SYBDATETIME},
        /* LOGICAL */{NULL, 0, SYBINT4}
};

#define MAXOPENS 10

DBPROCESS *dbproc[MAXOPENS];
int     count[MAXOPENS];

int bulk_open(database, table, password)
        char database[];
        char table[];
        char password[];
{
        LOGINREC *login;
        int db;

        /* make note we have established a connection */
        for (db=0; db<MAXOPENS; db++)
                if (dbproc[db] == NULL) break;
        count[db] = 0;

        /* Install an error and Message handler */
        dbmsghandle(msg_handler);
        dberrhandle(err_handler);

        /* initialize dblib */
        if (dbinit() != SUCCEED)
                printf(“Can’t initialize the DB library\n”);

        /* allocate a login record and fill it in */
        login = dblogin();
        if (login == NULL)
                printf(“Can’t allocate a login record.\n”);
        DBSETLUSER(login, “sa”);

        if(strlen(password) > 0)
        DBSETLPWD(login, password);

        DBSETLAPP(login, table);
        BCP_SETL(login, TRUE);

        /* Set Packet Size to 4096 */
        DBSETLPACKET(login, 4096);

        /* establish a connection with the server specified by DSQUERY */
        dbproc[db] = dbopen(login, NULL);
        if (dbproc[db] == NULL)
                printf(“Can’t establish connection. Is DSQUERY set?\n”);

        /* select the database to use */
        if (database != NULL)
                if (dbuse(dbproc[db], database) != SUCCEED)
                        printf(“Can’t select database: %s\n”, database);

        /* release the login record */
        dbloginfree(login);

        /* prepare to do a bulk copy */
        if (bcp_init(dbproc[db], table, NULL, NULL, DB_IN) !=
SUCCEED)
        printf(“Can’t initialize the bulk copy to table %s\n”, table);

        return db;
}



bulk_bind(db, column, name, address, type)
        int db;
        int column;
        char name[];
        void *address;
        int type;
{
        if (bcp_bind(dbproc[db], address, 0, -1, parm[type].terminator,
                        parm[type].termlen, parm[type].type, column) !=
SUCCEED)
                printf(“Can’t bind column %d to 0x%x, type=%d\n”,
                        column,address,type);
}

bulk_null(db, column)
        int db;
        int column;
{
        if (bcp_collen(dbproc[db], 0, column) != SUCCEED)
                printf(“Can’t null column %d\n”, column);
}


bulk_non_null(db, column)
        int db;
        int column;
{
        if (bcp_collen(dbproc[db], -1, column) != SUCCEED)
                printf(“Can’t non-null column %d\n”, column);
}


bulk_load(db)
        int db;
{
        count[db]++;
        if (bcp_sendrow(dbproc[db]) != SUCCEED)
                printf(“bulk_load: Can’t load row\n”);
        if (count[db]%batch_size == 0 && (bcp_batch(dbproc[db]) == -
1))
        printf(“bulk_load: Can’t post rows\n”);
        if (count[db]%1000 == 0) write(1,”.”,1);
        if (count[db]%50000 == 0) write(1,”\n”,1);

}

bulk_close(db)
        int db;
{
        if (bcp_done(dbproc[db]) == -1)
                printf(“Problems completing the bulk copy.\n”);
```

```
        dbproc[db] = NULL;
        if (count[db] >= 1000) write(1,”\n”,1);
}
###################################
#       loader.h #
###################################
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED

#include <sybfront.h>
#include <sybdb.h>
#include <time.h>


/* Population constants */
#ifdef CACHED
#define MAXITEMS 10000
#define CUST_PER_DIST 300
#define DIST_PER_WARE 10
#define ORD_PER_DIST 300
#else
#define MAXITEMS 100000
#define CUST_PER_DIST 3000
#define DIST_PER_WARE 10
#define ORD_PER_DIST 3000
#endif

#defineNURAND_C 123

/* Types of application variables */
typedef int     COUNT;
typedef int     ID;
typedef double  MONEY;
typedef double  FLOAT;
typedef char    TEXT;
typedef struct { int x[2];} DATE;
typedef int     LOGICAL;

typedef enum
    {COUNT_T, ID_T, MONEY_T, FLOAT_T, TEXT_T,
     DATE_T, LOGICAL_T, MAX_T}
    DATA_TYPE;

typedef struct timeval TIME;





#define YES 1
#define NO 0
#define EOF (-1)

#ifndef NULL
#define NULL ((void *)0)
#endif

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

```
/* define function types */
externint   msg_handler();
externint   err_handler();
externint   batch_size;

#endif /* TPCC_INCLUDED */
##################################
#       makefile #
##################################
# @(#) Makefile 1.3@(#)
.KEEP_STATE:

SQL_RELEASE=/export/home/sybase
NSL          =      -lnsl
#axpflag=  -ldnet_stub
#hpflag     =      +b
/local_home/u/products/sql_server/S1001_940125/lib

INCLUDE =      $(SQL_RELEASE)/include
LIBDIR     =      $(SQL_RELEASE)/lib
#DEBUG   =      -g
OPT         =      -O
CC               =       /usr/dist/pkgs/devpro,v4.2/5.x-sparc/bin/cc

LDFLAGS= -L$(LIBDIR) -lsybdb -lm -lc $(NSL) $(axpflag) -
R$(LIBDIR)
#LDFLAGS= -Wl,-a,archive $(LIBDIR)/libsybdb.a -lm -lc $(NSL) -Q
#LDFLAGS        = $(LIBDIR)/libsybdb.a -lm -lc $(NSL) -Q

CFLAGS    =      $(OPT) $(DEBUG) -I$(INCLUDE) $(FLAGS)
#CFLAGS = $(OPT) $(DEBUG) $(INCLUDE) -DRPC
#CFLAGS = $(OPT) $(DEBUG) $(INCLUDE)

LOAD_OBJS= error.o load.o bulk_sybase.o
L_HEADERS= loader.h

#FLAGS    =      -DDELAY
#FLAGS    =      -DACID
#FLAGS    =      -DCACHED

all:    load

load: $(LOAD_OBJS) $(L_HEADERS)
        $(CC) $(DEBUG) -o load $(FLAGS) $(LOAD_OBJS)
$(LDFLAGS)

clean:
        rm -f *.o

panic:
        rm -f *.o load
```

Appendix C: Tunable Parameters

This Appendix contains the configuration information for the operating system, the RDBMS and Tuxedo.

Operating System Configuration Values

The Solaris 7 kernel configuration parameters set in the file /etc/system are given below.

Solaris 7 Configuration File for Enterprise 4500 Server

```
set maxusers = 40
set ufs_ninode=1280
set ncsize=512
set bufhwm=1024

set segspt_minfree = 0x4000
set minfree=512
set desfree=1024
set lotsfree=2048

* # for sql server
set shmsys:shminfo_shmmax=0xffffffffff
set rlim_fd_max=4096
set rlim_fd_cur=1024

set ssd:ssd_error_level=0

* set forthdebug=1
* disable for now
```

```
set tune_t_fsflushr = 50
set autoup = 300

* vxvm_START (do not remove)
forceload: drv/vxio
forceload: drv/vxspec
* vxvm_END (do not remove)
set vxio:vol_maxio=2048
```

Solaris 7 configuration file for the client systems:

```
set pt_cnt=4096

set shmsys:shminfo_shmmax=0xffffffff

set shmsys:shminfo_shmseg=600
set shmsys:shminfo_shmmni=10

set msgsys:msginfo_msgmni=4096
set msgsys:msginfo_msgmax=2048
set msgsys:msginfo_msgmnb=200000
set msgsys:msginfo_msgmap=200000
set msgsys:msginfo_msgseg=10000
set msgsys:msginfo_msgssz=2048
set msgsys:msginfo_msgtql=5000

set semsys:seminfo_semmns=5000
set semsys:seminfo_semmni=5000
set semsys:seminfo_semmsl=5000
set semsys:seminfo_semmap=5000
set semsys:seminfo_semume=1
set semsys:seminfo_semmnu=5000
set tune_t_fsflushr = 50
set autoup = 300
```

RDBMS Configuration values

```
##############################################################################
#
#               Configuration File for the Sybase SQL Server
#
#               Please read the System Administration Guide (SAG)
#               before changing any of the values in this file.
#
##############################################################################



[Configuration Options]
```

```
[General Information]

[Backup/Recovery]
        recovery interval in minutes = 2000
        print recovery information = DEFAULT
        tape retention in days = DEFAULT

[Cache Manager]
        number of oam trips = DEFAULT
        number of index trips = DEFAULT
        procedure cache percent = 1
        memory alignment boundary = DEFAULT
        global async prefetch limit = DEFAULT
        global cache partition number = 16

[Named Cache:c_customer]
        cache size = 1M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 2

[2K I/O Buffer Pool]
        pool size = 1M
        wash size = 256 K
        local async prefetch limit = 0

[Named Cache:c_index]
        cache size = 432M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 8

[2K I/O Buffer Pool]
        pool size = 432M
        wash size = 256 K
        local async prefetch limit = 0

[Named Cache:c_index_nc]
        cache size = 2048M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[2K I/O Buffer Pool]
        pool size = 2048M
        wash size = 512 K
        local async prefetch limit = 0

[Named Cache:c_log]
        cache size = 7168K
        cache status = log only
        cache replacement policy = DEFAULT
        local cache partition number = 1
```

```
[2K I/O Buffer Pool]
        pool size = 512K
        wash size = 128 K
        local async prefetch limit = 0


[4K I/O Buffer Pool]
        pool size = 6656K
        wash size = 512 K
        local async prefetch limit = 1


[Named Cache:c_new_order]
        cache size = 192M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[2K I/O Buffer Pool]
        pool size = 192M
        wash size = 64 K
        local async prefetch limit = 0

[Named Cache:c_no_index]
        cache size = 6M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[2K I/O Buffer Pool]
        pool size = 6M
        wash size = 20 K
        local async prefetch limit = 0

[Named Cache:c_order_line]
        cache size = 2560M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 16

[2K I/O Buffer Pool]
        pool size = 2560M
        wash size = 64 K
        local async prefetch limit = 0

[Named Cache:c_ol_index]
        cache size = 288M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 16

[2K I/O Buffer Pool]
```

```
        pool size = 288M
        wash size = 20 K
        local async prefetch limit = 0

[Named Cache:c_orders]
        cache size = 2048M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 8

[2K I/O Buffer Pool]
        pool size = 512M
        wash size = DEFAULT
        local async prefetch limit = 0

[8K I/O Buffer Pool]
        pool size = 1536M
        wash size = DEFAULT
        local async prefetch limit = 0

[Named Cache:c_ord_index]
        cache size = 40M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 8

[2K I/O Buffer Pool]
        pool size = 40M
        wash size = 20 K
        local async prefetch limit = 0

[Named Cache:c_stock]
        cache size = 15616M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 8

[2K I/O Buffer Pool]
        pool size = 15616M
        wash size = 32 M
        local async prefetch limit = 0

[Named Cache:c_stock_index]
        cache size = 1088M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 16

[2K I/O Buffer Pool]
        pool size = 1088M
        wash size = 20 K
        local async prefetch limit = 0
```

```
[Named Cache:c_tinyhot]
        cache size = 58M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 16

[2K I/O Buffer Pool]
        pool size = 58M
        wash size = 20 K
        local async prefetch limit = 100



[Named Cache:default data cache]
        cache size = 16M
        cache status = default data cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[2K I/O Buffer Pool]
        pool size = 16M
        wash size = 20 K
        local async prefetch limit = 0

[Meta-Data Caches]
        number of open databases = 5
        number of open objects = 100
        open object spinlock ratio = 32
        number of open indexes = 100
        open index hash spinlock ratio = 8
        open index spinlock ratio = 16

[Disk I/O]
        allow sql server async i/o = DEFAULT
        disk i/o structures = 4096
        page utilization percent = DEFAULT
        number of devices = 100
        disable disk mirroring = DEFAULT
        disable character set conversions = DEFAULT
        enable unicode conversions = DEFAULT
        size of unilib cache = DEFAULT

[Network Communication]
        default network packet size = DEFAULT
        max network packet size = 4096
        remote server pre-read packets = DEFAULT
        number of remote connections = DEFAULT
        allow remote access = DEFAULT
        number of remote logins = DEFAULT
        number of remote sites = 2
        max number network listeners = 1
        tcp no delay = DEFAULT
        allow sendmsg = DEFAULT
        syb_sendmsg port number = DEFAULT
```

```
[O/S Resources]
        max async i/os per engine = 128
        max async i/os per server = 1024

[Parallel Query]
        number of worker processes = DEFAULT
        memory per worker process = DEFAULT
        max parallel degree = DEFAULT
        max scan parallel degree = DEFAULT

[Physical Resources]

[Physical Memory]
        total memory = 14057472
        additional network memory = 11059200
        lock shared memory = DEFAULT
        shared memory starting address = DEFAULT
        max SQL text monitored = DEFAULT

[Processors]
        max online engines = 14
        min online engines = DEFAULT

[SQL Server Administration]
        default database size = DEFAULT
        identity burning set factor = DEFAULT
        allow nested triggers = DEFAULT
        allow updates to system tables = 1
        print deadlock information = DEFAULT
        default fill factor percent = DEFAULT
        default exp_row_size percent = DEFAULT
        number of mailboxes = DEFAULT
        number of messages = DEFAULT
        number of alarms = DEFAULT
        number of pre-allocated extents = DEFAULT
        event buffers per engine = DEFAULT
        cpu accounting flush interval = 2147483647
        i/o accounting flush interval = 2147483647
        sql server clock tick length = DEFAULT
        runnable process search count = DEFAULT
        i/o polling process count = DEFAULT
        time slice = DEFAULT
        deadlock retries = DEFAULT
        cpu grace time = DEFAULT
        number of sort buffers = DEFAULT
        number of large i/o buffers = 32
        size of auto identity column = DEFAULT
        identity grab size = DEFAULT
        page lock promotion HWM = DEFAULT
        page lock promotion LWM = DEFAULT
        page lock promotion PCT = DEFAULT
        housekeeper free write percent = 0
        enable housekeeper GC = DEFAULT
        partition groups = DEFAULT
        partition spinlock ratio = DEFAULT
        allow resource limits = DEFAULT
        number of aux scan descriptors = DEFAULT
        SQL Perfmon Integration = DEFAULT
        allow backward scans = DEFAULT
        row lock promotion HWM = DEFAULT
        row lock promotion LWM = DEFAULT
        row lock promotion PCT = DEFAULT
        license information = DEFAULT

[User Environment]
        number of user connections = 900
        stack size = DEFAULT
        stack guard size = DEFAULT
        permission cache entries = DEFAULT
        user log cache size = 4096
        user log cache spinlock ratio = DEFAULT

[Lock Manager]
        number of locks = 70000
        deadlock checking period = DEFAULT
        freelock transfer block size = DEFAULT
        max engine freelocks = 32
        lock spinlock ratio = 10
        lock hashtable size = 16384
        lock scheme = DEFAULT
        lock wait period = DEFAULT
        read committed with lock = DEFAULT

[Security Related]
        systemwide password expiration = DEFAULT
        audit queue size = DEFAULT
        curread change w/ open cursors = DEFAULT
        allow procedure grouping = DEFAULT
        select on syscomments.text = DEFAULT
        auditing = DEFAULT
        current audit table = DEFAULT
        suspend audit when device full = DEFAULT
        max roles enabled per user = DEFAULT
        unified login required = DEFAULT
        use security services = DEFAULT
        msg confidentiality reqd = DEFAULT
        msg integrity reqd = DEFAULT
        msg replay detection reqd = DEFAULT
        msg origin checks reqd = DEFAULT
        msg out-of-seq checks reqd = DEFAULT
        secure default login = DEFAULT
        dump on conditions = DEFAULT

[Extended Stored Procedure]
        esp unload dll = DEFAULT
        esp execution priority = DEFAULT
        esp execution stacksize = DEFAULT
        xp_cmdshell context = DEFAULT
        start mail session = DEFAULT

[Error Log]
```

```
        event logging = DEFAULT
        log audit logon success = DEFAULT
        log audit logon failure = DEFAULT
        event log computer name = DEFAULT

[Rep Agent Thread Administration]
        enable rep agent threads = DEFAULT
        maximum dump conditions = DEFAULT

[Component Integration Services]
        enable cis = DEFAULT
        cis connect timeout = DEFAULT
        cis bulk insert batch size = DEFAULT
        max cis remote connections = DEFAULT
        max cis remote servers = DEFAULT
        cis packet size = DEFAULT
        cis cursor rows = DEFAULT
        cis rpc handling = DEFAULT
```

Tuxedo Configuration values

```
*RESOURCES
IPCKEY          40001# IPC KEY from 32,768 to 16,777,215
MASTER          client1# machine on which master copy is found
UID             30# user id as displayed by command "id"
GID             5432# group id as displayed by command "id"
PERM            0666# UNIX permission from 0001 to 0777 in octal
MAXACCESSERS    3700# max no of processes accesing bulleting board
MAXGTT          1000# maximum simultaneous global transactions
MAXSERVERS      400# maximum number of servers
MAXSERVICES     400# maximum number of services
MAXCONV         1
MODEL           SHM# SHM=single processor, MP=multi processor
LDBAL           N# load balancing, Y=yes, N=no
CMTRET          COMPLETE
#MAXBUFTYPE     16# maximum buffer types
#MAXBUFSTYPE    32# maximum buffer subtypes
SCANUNIT        30# scan program wake-up time in secs.
SANITYSCAN      5# sanity scan wake-up
DBBLWAIT        1# scanunit multiplier for DBBL max time wait
BBLQUERY        60# check out wake-up time
BLOCKTIME       10# blocking call time-out
NOTIFY          DIPIN
SYSTEM_ACCESS   FASTPATH
USIGNAL         SIGUSR2

*MACHINES
"client1"LMID="client1"
        TUXCONFIG="/export/home/dbbench/tuxedo/tuxconfig.client1"
        ROOTDIR="/export/home/tuxedo"
        APPDIR="/export/home/dbbench/tuxedo"
        ULOGPFX="/export/home/dbbench/tuxedo/ULOGclient1"
        ENVFILE="/export/home/dbbench/tuxedo/client1.env"
```

```
*GROUPS
"group1"LMID="client1"GRPNO=1

*SERVERS

"tpcc_srv_newordpay"SRVGRP="group1"SRVID=1 CLOPT="-A" RQADDR="newopayq1" REPLYQ=NMIN=12
RESTART=Y

"tpcc_srv_stockdel"SRVGRP="group1"SRVID=101 CLOPT="-A" RQADDR="stockdelq1" REPLYQ=NMIN=3
RESTART=Y

*SERVICES
"DEL"
"NEWO"
"ORDS"
"PAYM"
"STOCK"

*ROUTING
```

Compilation Flags

These are the compilation flags used to compile the application code:

```
-O -L/export/home/sybase/lib -lsybdb -lm -lc -lnsl
```

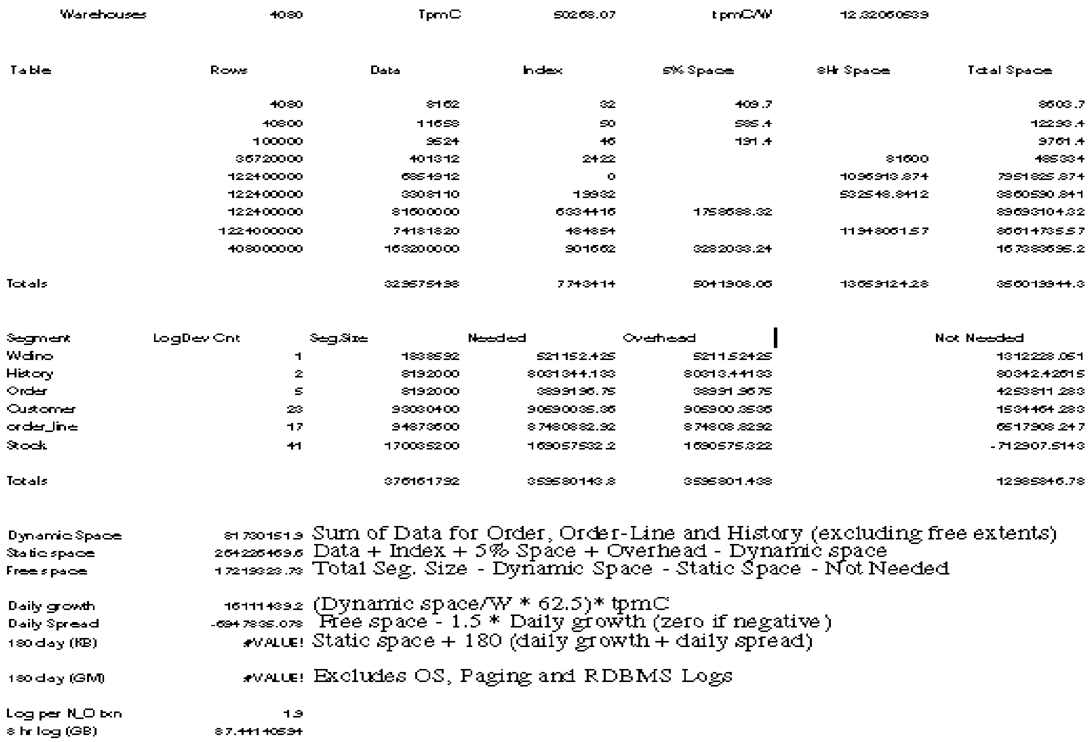
Appendix D: Disk Storage D

The calculations used to determine the storage requirements for the 8 hour logical log and the 180-day space calculations are contained in this appendix.

The calculations for the 8 hour recovery log was determined as follows :

The number of logpages used during the measurement run was determined by using the Sybase stored procedure *sp_helpdb* tpcc before and after the run. We found the amount of log space used by the DBMS during the benchmark run. This was 15801328 KB. The amount of log disk used per transaction was 15801328/4114487 = 3.84 KB. Therefore we need 50268.07*60*8*3.84*1KB = 88.36 GB

We allocated 48 * 9GB disks for the logs and the same for the mirrors.



Warehouses	4080	TpmC	50268.07	tpmC/W	12.32060539	

Table	Rows	Data	Index	5% Space	8Hr Space	Total Space
	4080	8162	32	409.7		8603.7
	40800	11658	50	585.4		12293.4
	100000	9524	46	191.4		9761.4
	36720000	401312	2422		81600	485334
	122400000	6854912	0		1096913.874	7951825.874
	122400000	3308110	19932		532548.8412	3860590.841
	122400000	81600000	6334416	1758688.32		89693104.32
	1224000000	74181820	484854		11348061.57	86014735.57
	408000000	163200000	901662	3282033.24		167383695.2
Totals		329575498	7743414	5041908.06	13059124.28	355419944.3

Segment	LogDev Cnt	Seg.Size	Needed	Overhead	Not Needed
Wdino	1	1838592	521152.425	5211.52425	1312228.051
History	2	8192000	8031344.133	80313.44133	80342.42515
Order	5	8192000	3899196.75	38991.9675	4253811.283
Customer	23	93030400	90590035.36	905900.3536	1534464.283
order_line	17	94873600	87480882.92	874808.8292	6517908.247
Stock	41	170035200	169057532.2	1690575.322	-712907.5143
Totals		376161792	359580143.8	3595801.438	12985846.78

Dynamic Space	81730151.9	Sum of Data for Order, Order-Line and History (excluding free extents)
Static space	264226469.6	Data + Index + 5% Space + Overhead - Dynamic space
Free space	17219323.73	Total Seg. Size - Dynamic Space - Static Space - Not Needed
Daily growth	16111439.2	(Dynamic space/W * 62.5)* tpmC
Daily Spread	-6947835.078	Free space - 1.5 * Daily growth (zero if negative)
180 day (KB)	#VALUE!	Static space + 180 (daily growth + daily spread)
180 day (GM)	#VALUE!	Excludes OS, Paging and RDBMS Logs
Log per N_O txn	1.9	
8 hr log (GB)	87.44140534	

Appendix E: Driver Scripts

The following code sections show how the transactions are generated and how statistics are gathered. Each of the transaction functions generates the input data for that transaction, sends it to the client, reads the output form and computes keying, response and think time statistics.

This is the main loop of the RTE:

```
/* run for ramp up without capturing the stats */
        i=0;
        in_ramp = 1;
        while (1)
        {
                tx_type = do_menu(); /* Select transaction */
                switch (tx_type) {
                case NEWORDER:
                        do_neworder();
                        break;
                case PAYMENT:
                        do_payment();
                        break;
                case DELIVERY:
                        do_delivery();
                        break;
                case ORDSTAT:
                        do_ordstat();
                        break;
                case STOCKLEVEL:
                        do_stocklevel();
                        break;
                default:
                        fprintf(stderr, "%s: Slave %d: Internal error. Tx-type = %d\n",
                        hostname, slave_num, tx_type);
                        cleanup(-1);
                }
                end_time = gettime();
        if ( end_time >= control->end_rampup &&
                end_time < control->end_stdystate )
                in_ramp = 0;
        else
            in_ramp = 1;
        if (end_time >= control->end_rampdown)
            break;
    }
```

The do_menu function selects the transaction to execute based on the weighted distribution algorithm.

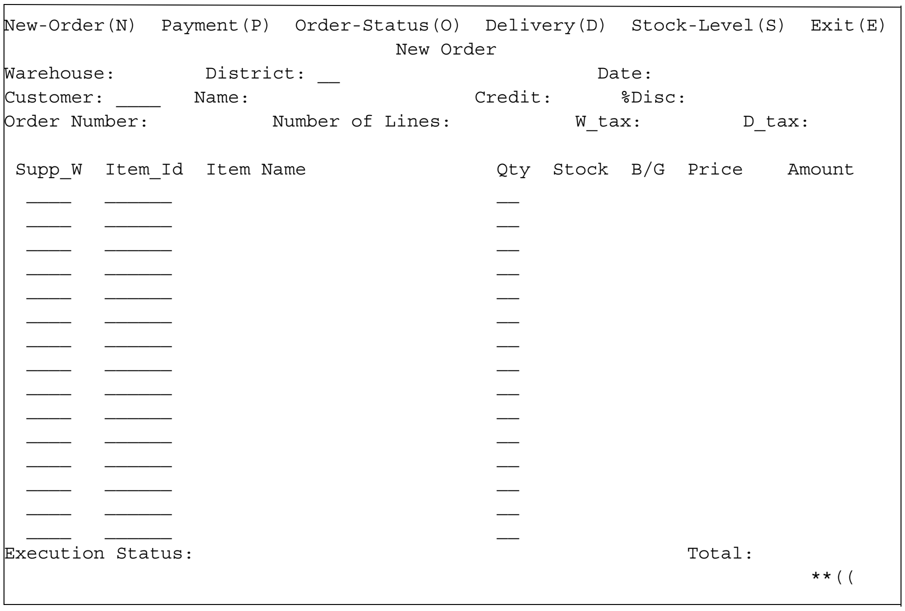
```
int
do_menu()
{
    int val, result, menu_start, menu_end, menu_resp;
    char ch;
    /* Read menu line from client */
    /* Choose tx. type*/
    /* Now select menu and compute menu response time */
    menu_start = gettime();
    /* Write menu selection to client */
    /* Read input form for this transaction type */
    menu_end = gettime();
    menu_resp = menu_end - menu_start;
    if ( ! in_ramp) {
        statsp->menu_resp += menu_resp;
        /* Post in histogram bucket */
        if ((menu_resp / MENU_BUCKET) < MENU_MAX)
            statsp->menu_hist[menu_resp / MENU_BUCKET]++;
        else
            statsp->menu_hist[MENU_MAX - 1]++;
        if (menu_resp > statsp->menu_max)
            statsp->menu_max = menu_resp;
    }
    return(result);
}
/*
 * Function: do_neworder
 * This function executes the neworder transaction
 * It generates all the input fields, sends it to the
 * client over the keying time, measures the response
 * time, reads the results and delays for the think time.
 */
 /* The code for the other transactions is similar */
do_neworder()
{
    struct newo_fld no;
    struct items_fld *itemp = no.items;
    int ol_cnt, rbk, remote = 0, i, x;
    char *bufp = fldbuf;
    int start_time, end_time, key_time, resp_time, elapse_time, del;
    start_time = gettime();
    /* Now wait for keying time */
    poll (0, 0, NEWO_KEY);
    /* Generate all input data */
    no.d_id = random(1, 10);
    no.c_id = NURand(1023, 1, 3000, CONST_CID);
```

```
ol_cnt = random(5, 15);
rbk = random(1, 100);/* trans. to be rolledback */
sprintf(bufp, "%02d%04d", no.d_id, no.c_id);
bufp += strlen(bufp);
/* Generate all the item fields */
for (i=0; i < ol_cnt; i++, itemp++) {
        itemp->ol_i_id = NURand(8191, 1, 100000, CONST_IID);
        /* If last item and rbk, select unused item */
        if (i == ol_cnt - 1 && rbk == 1) {
                itemp->ol_i_id = 100001;
        }
        x = random(1, 100);
        if ( x > 1)
                itemp->ol_supply_w_id = W_ID;
        else {
                /* Select a warehouse other than w_id */
                do {
                        x = random(1, control->scale);
                } while (x == W_ID);
                itemp->ol_supply_w_id = x;
                remote++;
        }
        itemp->ol_quantity = random(1, 10);
        sprintf(bufp, "%04d%06d%02d",itemp->ol_supply_w_id,
        itemp->ol_i_id, itemp->ol_quantity);
        bufp += strlen(bufp);
}
strcpy(bufp, leave_key);
bufp += 2;
/* Compute keying time info */
end_time = gettime();
key_time = end_time - start_time;
start_time = end_time;

/* Now send fields to client */
/* Read output screen from client */
end_time = gettime();
/* Store elapse time info for thruput */
elapse_time = end_time - control->start_time;
/* compute the how long it took to run the tx */
resp_time = end_time - start_time + control->newo_delta;
/* Wait think time */
del = delay(control->newo_think, 5*control->newo_think);
poll(0, 0, del + control->newo_delta);
end_time = gettime();
/* Now post all stats */
if ( ! in_ramp && end_time <= control->end_stdystate) {
        statsp->newo_cnt++; /* another one bytes the dust */
        if ( rbk == 1 )
                statsp->newo_rbkcnt++;
        statsp->newo_remote += remote;
        statsp->newo_olcnt += ol_cnt;
        statsp->newo_key += key_time;
        /* Save keying time in histogram bucket */
        statsp->newo_resp += (double) resp_time;/* sum up the response time */
        /* Save response time in histogram bucket */
```

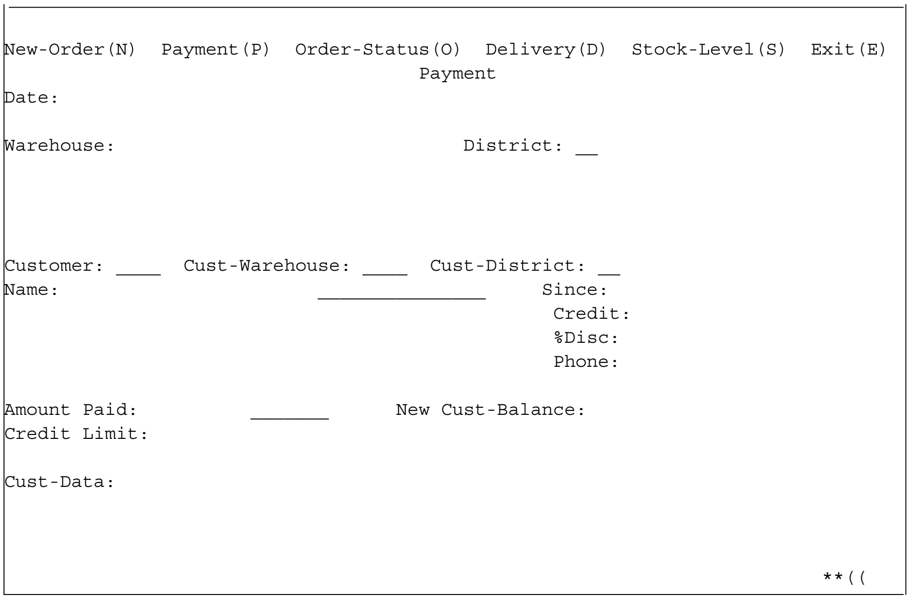
```
                statsp->newo_think += (double) del;
                /* Save think time in histogram bucket */
        }
}
```

Appendix F: Screen Layouts

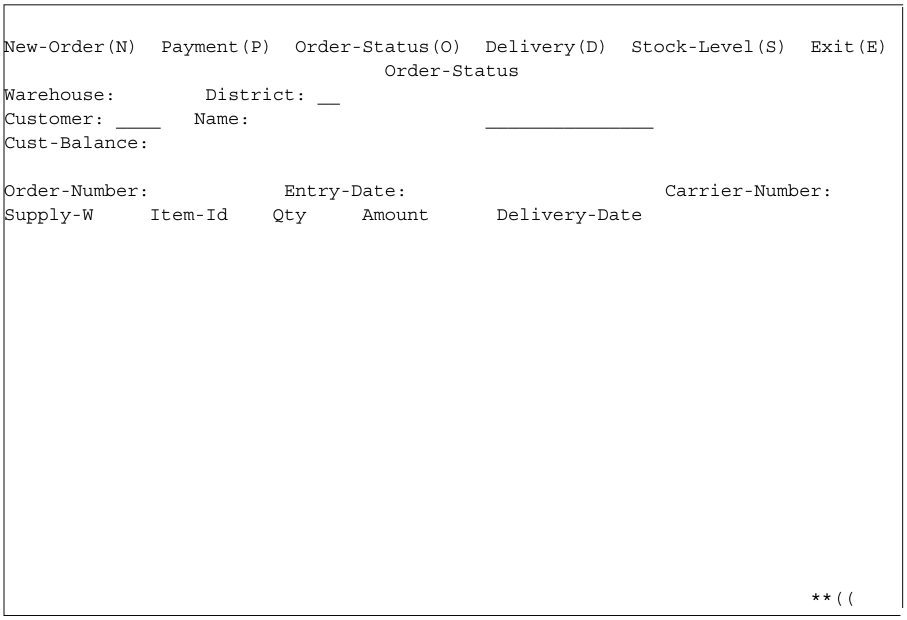
This Appendix contains the screen form layouts for the 5 transactions.



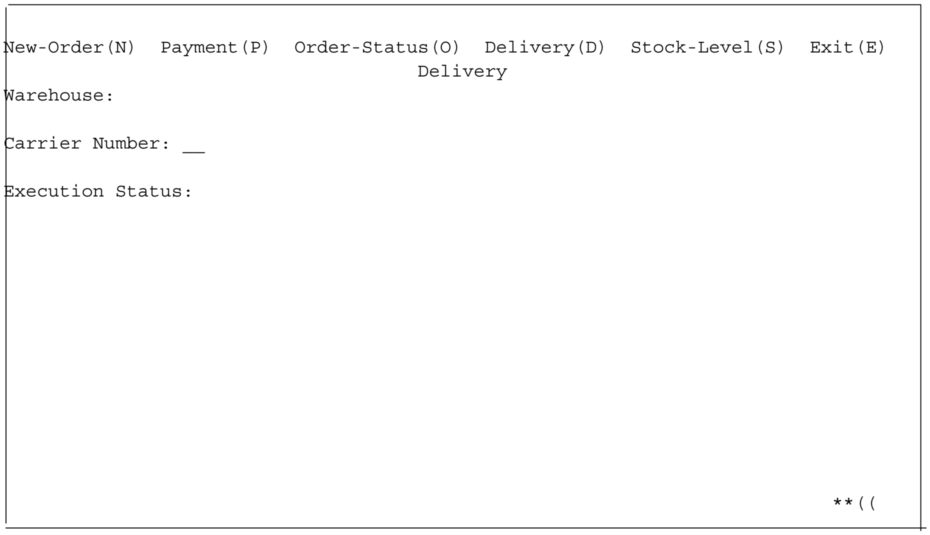
```
New-Order(N)  Payment(P)  Order-Status(O)  Delivery(D)  Stock-Level(S)  Exit(E)
                                   New Order
Warehouse:        District: __                       Date:
Customer: ____   Name:                    Credit:      %Disc:
Order Number:           Number of Lines:           W_tax:         D_tax:

 Supp_W  Item_Id  Item Name                 Qty  Stock  B/G  Price    Amount
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
  ____   ______                             __
Execution Status:                                            Total:
                                                                        **((
```



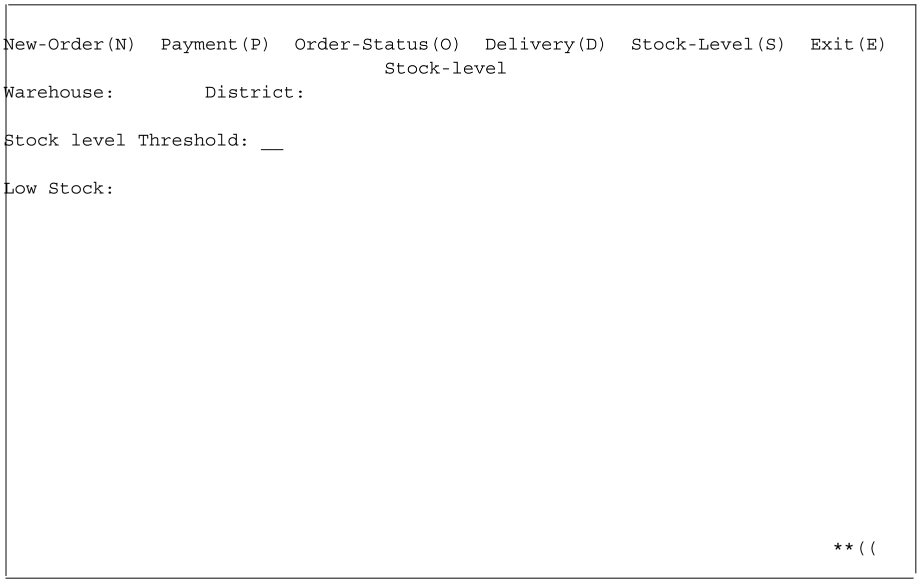
```
New-Order(N)  Payment(P)  Order-Status(O)  Delivery(D)  Stock-Level(S)  Exit(E)
                                     Payment
Date:

Warehouse:                               District: __




Customer: ____  Cust-Warehouse: ____  Cust-District: __
Name:                       _______________      Since:
                                                  Credit:
                                                  %Disc:
                                                  Phone:

Amount Paid:          _______      New Cust-Balance:
Credit Limit:

Cust-Data:




                                                                        **((
```



```
New-Order(N)  Payment(P)  Order-Status(O)  Delivery(D)  Stock-Level(S)  Exit(E)
                                  Order-Status
Warehouse:        District: __
Customer: ____     Name:                    _______________
Cust-Balance:

Order-Number:            Entry-Date:                      Carrier-Number:
Supply-W      Item-Id     Qty      Amount      Delivery-Date





















                                                                       **((
```



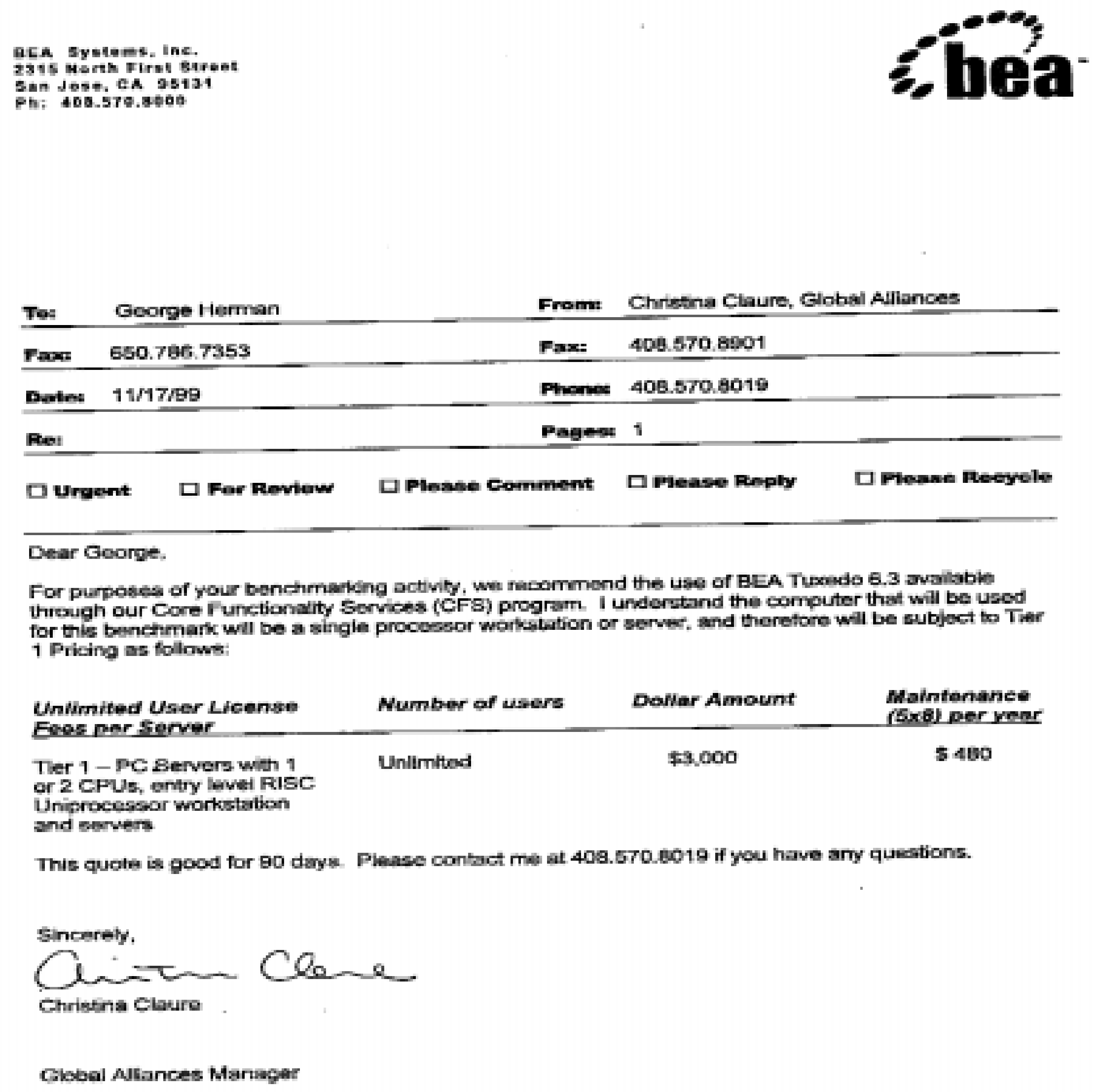
```
New-Order(N)   Payment(P)   Order-Status(O)   Delivery(D)   Stock-Level(S)   Exit(E)
                                      Delivery
Warehouse:

Carrier Number: __

Execution Status:












                                                                              **((
```



```
New-Order(N)   Payment(P)   Order-Status(O)   Delivery(D)   Stock-Level(S)   Exit(E)
                                   Stock-level
Warehouse:        District:

Stock level Threshold: __

Low Stock:













                                                                              **((
```

Appendix G: Price Quotes G

The following pages contain the pricing quotes for the hardware and software included in this FDR.



BEA Systems, Inc.
2315 North First Street
San Jose, CA 95131
Ph: 408.570.8000

To:	George Herman	From:	Christina Claure, Global Alliances
Fax:	650.786.7353	Fax:	408.570.8901
Date:	11/17/99	Phone:	408.570.8019
Re:		Pages:	1

□ Urgent □ For Review □ Please Comment □ Please Reply □ Please Recycle

Dear George,

For purposes of your benchmarking activity, we recommend the use of BEA Tuxedo 6.3 available through our Core Functionality Services (CFS) program. I understand the computer that will be used for this benchmark will be a single processor workstation or server, and therefore will be subject to Tier 1 Pricing as follows:

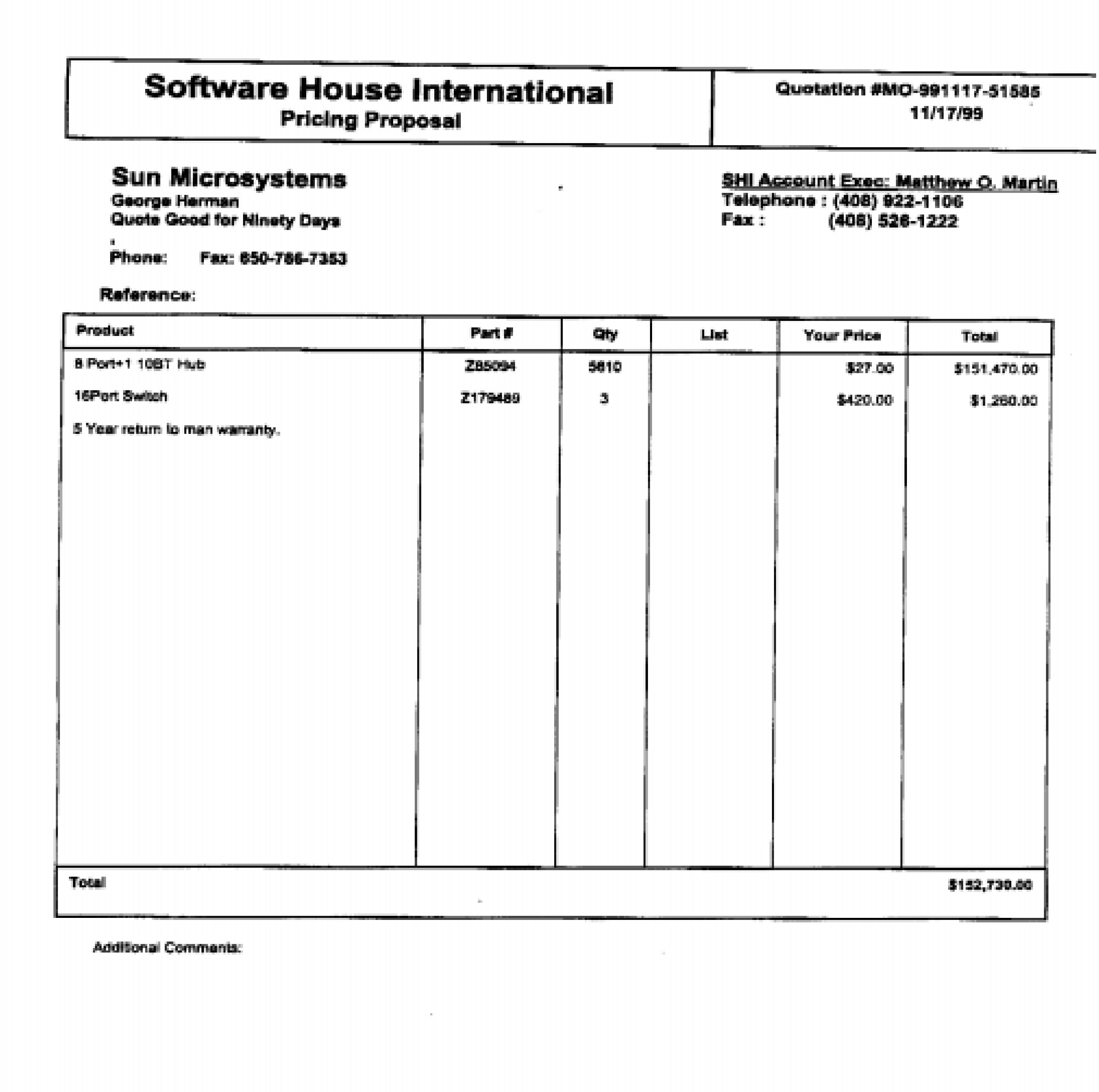
Unlimited User License Fees per Server	*Number of users*	*Dollar Amount*	*Maintenance (5x8) per year*
Tier 1 – PC Servers with 1 or 2 CPUs, entry level RISC Uniprocessor workstation and servers	Unlimited	$3,000	$ 480

This quote is good for 90 days. Please contact me at 408.570.8019 if you have any questions.

Sincerely,

Christina Claure

Global Alliances Manager



Software House International
Pricing Proposal

Quotation #MO-991117-51585
11/17/99

Sun Microsystems
George Herman
Quote Good for Ninety Days

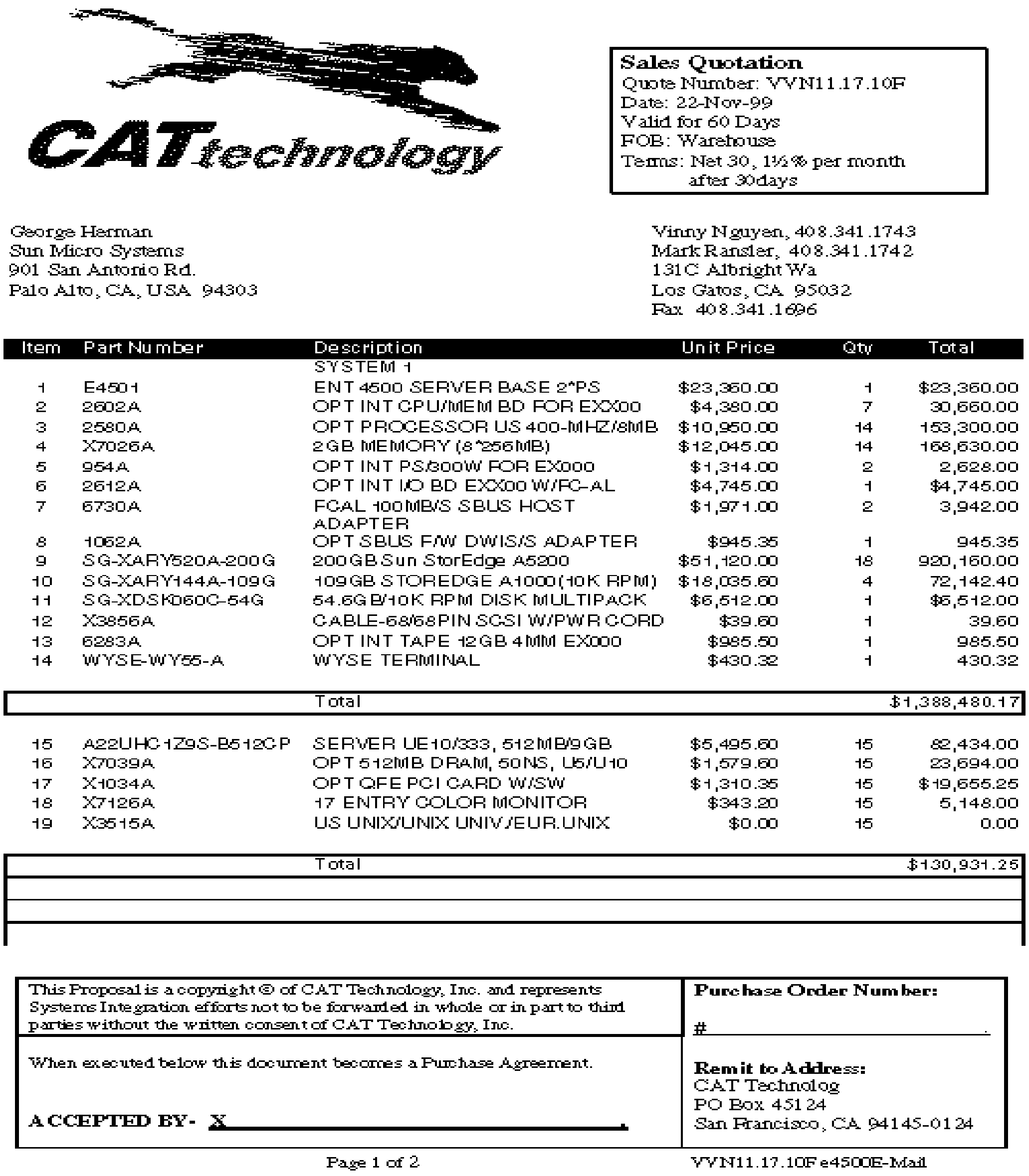
Phone: Fax: 650-786-7353

Reference:

SHI Account Exec: Matthew O. Martin
Telephone : (408) 922-1106
Fax : (408) 526-1222

Product	Part #	Qty	List	Your Price	Total
8 Port+1 10BT Hub	Z85094	5610		$27.00	$151,470.00
16Port Switch	Z179489	3		$420.00	$1,260.00
5 Year return to man warranty.					
Total					$152,730.00

Additional Comments:



CATtechnology

Sales Quotation
Quote Number: VVN11.17.10F
Date: 22-Nov-99
Valid for 60 Days
FOB: Warehouse
Terms: Net 30, 1½% per month
after 30days

George Herman
Sun Micro Systems
901 San Antonio Rd.
Palo Alto, CA, USA 94303

Vinny Nguyen, 408.341.1743
Mark Ransler, 408.341.1742
131C Albright Wa
Los Gatos, CA 95032
Fax 408.341.1696

Item	Part Number	Description	Unit Price	Qty	Total
		SYSTEM 1			
1	E4501	ENT 4500 SERVER BASE 2*PS	$23,360.00	1	$23,360.00
2	2602A	OPT INT CPU/MEM BD FOR EXX00	$4,380.00	7	30,660.00
3	2580A	OPT PROCESSOR US 400-MHZ/8MB	$10,950.00	14	153,300.00
4	X7026A	2GB MEMORY (8*256MB)	$12,045.00	14	168,630.00
5	954A	OPT INT PS/300W FOR EX000	$1,314.00	2	2,628.00
6	2612A	OPT INT I/O BD EXX00 W/FC-AL	$4,745.00	1	$4,745.00
7	6730A	FCAL 100MB/S SBUS HOST ADAPTER	$1,971.00	2	3,942.00
8	1062A	OPT SBUS F/W DWIS/S ADAPTER	$945.35	1	945.35
9	SG-XARY520A-200G	200GB Sun StorEdge A5200	$51,120.00	18	920,160.00
10	SG-XARY144A-109G	109GB STOREDGE A1000(10K RPM)	$18,035.60	4	72,142.40
11	SG-XDSK060C-54G	54.6GB/10K RPM DISK MULTIPACK	$6,512.00	1	$6,512.00
12	X3856A	CABLE-68/68PIN SCSI W/PWR CORD	$39.60	1	39.60
13	6283A	OPT INT TAPE 12GB 4MM EX000	$985.50	1	985.50
14	WYSE-WY55-A	WYSE TERMINAL	$430.32	1	430.32
		Total			$1,388,480.17
15	A22UHC1Z9S-B512CP	SERVER UE10/333, 512MB/9GB	$5,495.60	15	82,434.00
16	X7039A	OPT 512MB DRAM, 50NS, U5/U10	$1,579.60	15	23,694.00
17	X1034A	OPT QFE PCI CARD W/SW	$1,310.35	15	$19,655.25
18	X7126A	17 ENTRY COLOR MONITOR	$343.20	15	5,148.00
19	X3515A	US UNIX/UNIX UNIV/EUR.UNIX	$0.00	15	0.00
		Total			$130,931.25

This Proposal is a copyright © of CAT Technology, Inc. and represents Systems Integration efforts not to be forwarded in whole or in part to third parties without the written consent of CAT Technology, Inc.

When executed below this document becomes a Purchase Agreement.

ACCEPTED BY- X________________________

Purchase Order Number:

#________________________

Remit to Address:
CAT Technolog
PO Box 45124
San Francisco, CA 94145-0124

Page 1 of 2

VVN11.17.10F e4500E-Mail