Sybase®, Inc.

TPC Benchmark™ C
Full Disclosure Report

Sun Microsystems Starfire™ Enterprise™ 10000 Server Using Sybase Adaptive Server Enterprise 12.0.0.2 RDBMS

Revision 1, August 29, 2000
Submitted for Review
Compliant with Revision 3.5 of the TPC-C specification

TPC Benchmark C Full Disclosure Report

First Printing

2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

TRADEMARKS

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark™ C test conducted on the Sun Starfire Enterprise 10000 server system (also known as the Starfire™ server), running Sybase Adaptive Server Enterprise 12.0.0.2 RDBMS and BEA Systems, Inc. Tuxedo 6.3.

TPC Benchmark C Metrics

The standard TPC Benchmark 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 Enterprise 10000 server.

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 August 29, 2000. 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.

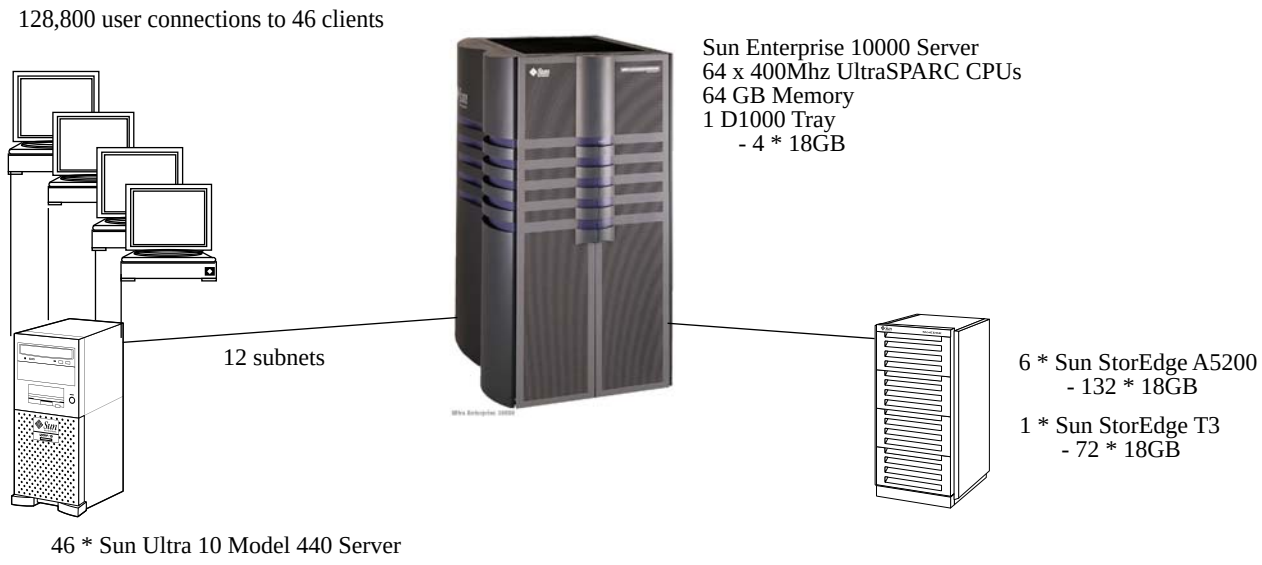
Sybase, Inc.

Sun Starfire Enterprise 10000 with Sybase ASE 12.0.0.2

TPC-C 3.5

Report Date:
August 29, 2000

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date	
$7,657,324	**156,873.03 tpmC**	**$48.81 per tpmC**	**February 28, 2001**	
Processors	Database Manager	Operating System	Other Software	Number of Users
64 * 400MHz	**Sybase ASE 12.0.0.2**	**Solaris 7**	**BEA Tuxedo 6.3 Veritas Vol. Mgr. 3.0.2**	**128,800**



Configuration

	Server System	Front End Systems
Database Nodes:	1 Sun Enterprise 10000 Server	46 * Ultra 10 Model 440
Processors	64 * 400 MHz UltraSPARC II	1 * 440 MHz UltraSPARC II each
Cache memory	32KB (D+I), 8MB external	32KB (D+I), 512KB external, each
Main memory	64 GB	1 GB each
Disk controllers	1 Fast/Wide SCSI-2 , 26 Dual Port FC-AL	1 * SCSI-2 each
Disk Drives	4 * 18GB SCSI-2, 864 * 18GB FC-AL	1 * 9GB SCSI-2 each
Total Disk Storage	15,624GB	9GB each
Terminals	1 Console	1 Console each
10 BaseT Hub	None	16,100 x 8-Port Hubs
100 Base T Hubs	12 x 8-Port Hubs	None

Sybase, Inc.

Sun Starfire Enterprise 10000 with Sybase ASE 12.0.0.2

TPC-C 3.5

Report Date: August 29, 2000

Pricing Summary

Description	Part Number	Source	Unit Price	Qty	Ext. Price	5 Yr. Maint.
Server Hardware						
E10000 Base Cabinet	E10000-4	2	129600	1	129600	337920
System Board: unpopulated	2761A	2	39600	16	633600	581376
CPU 400MHz 8MB L2$	2580A	2	10800	64	691200	
Memory Board unpopulated	7025A	2	7200	16	115200	
1GB for Ex000	7023A	2	6120	64	391680	
Dual SBus I/O daughter card	2730A	2	5400	16	86400	
System Service Proc	2754A	2	7560	1	7560	3264
AC input module	3875A	2	2160	4	8640	
48 volt power supply	9685A	2	2160	8	17280	
Power Control Module	9681A	2	720	1	720	
Fan tray	9671A	2	1080	16	17280	
Control Board w/ Enet Cable	2720A	2	10800	1	10800	
Power Cord for System	3865A	2	0	4	0	
Rack Mount D1000 W/4x18GB	SG-ARY154A-72GR5	2	8406	1	8406	
FCAL Host Adapter	6730A	2	1944	26	50544	
SBus Quad Fast Eth Host Adapter	1049A	2	1436	3	4309	
Sbus Fast Eth 2.0 Host Adapter	1059A	2	572	1	572	
12 Meter SCSI cable	979A	2	265	1	265	
2400GB StorEdge A5200	SG-ARY543A-2400G	2	376909	6	2261455	483840
N Am/Uni/Can Type 6 Country kit	3508A	2	0	1	0	
U.S. Power Cord for StorEdge	3858A	2	0	12	0	
Power Cord for Data Center Cab (US)	3800A	2	0	1	0	
1310GB StorEdge T3 ES	XT3ES-RK-88-1310	2	263376	1	263376	50688
SBus Ultra DWIS/S Host Adapter	X1065A	2	932	1	932	
15M Fibre Channel Cable	978A	2	175	24	4200	
Symmetra 16kva UPS	SX-SY16K	1	15845	1	15845	1315
12GB 4mm Tape Drive	6283A	2	972	1	972	
Subtotal					4720837	1458403
SunService Discount (10% Volume + 5% yearly prepay)						-211468
Server Hardware Subtotal					4720837	1246935
Server Software						
Solaris 7 Media	SOLMS-070W9999	1	100	1	100	
SPARC Compiler C/C++ 5.0	WCCIS-500-T999	1	995	1	995	1380
Sun StorEdge Comp Mgr V2.1	SCMMS-210-9P99	2	0	1	0	
Veritas Vol Mgr 3.0.4 for A5x000's	VVMGS-304-9999	2	0	1	0	
E10000 Sys Serv Proc 3.1	E10000-SW-SSP-3.1	2	0	1	0	
Sybase Adapt Serv Enterprise	Version 12.0.0.2 SWR 9197	3	295000	1	295000	236000
Sybase Development		3	11400	1	11400	
Sybase Open Client		3		1	0	
Subtotal					307495	237380
SunService Discount (10% Volume + 5% yearly prepay)						-200
Sybase Discount (10%)						-23600
Server Software Subtotal					307495	213580
Client Hardware						
Ultra 10 Model 444	A22UKC1A9P-C512P	2	4357	46	200401	192979
512 MB memory for Ultra 10	7039A	2	1598	46	73487	
PCI Quad Fast Eth Controller	1034A	2	1292	46	59450	
Color Monitor	X7126	2	289	46	13306	
Subtotal					346645	192979
SunService Discount (10% Volume + 5% yearly prepay)						-27982
Client Hardware Subtotal					346645	164997
Client Software						
BEA Tuxedo CFS 6.3		4	3000	46	138000	110400
Client Software Subtotal					138000	110400
User Connectivity						
8-port 10/100 Mbps Fast Eth Hub	Z222147	5	79	14	1106	
8-port+1 10 Mbps Eth Hub	Z222146	5	23	17710	407330	
User Connectivity Subtotal					408436	
			Total		5921412	1735912
Service for all Sun Products is from Sun Microsystems Inc.			5Yr. cost		7657324	
Service for Sybase products is from Sybase Inc.			tpmC Rating		156873.03	
Service for Tuxedo is from BEA Systems Inc.			$/tpmC		48.81	

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.

Note: (Source): 1) Sun Microsystems Inc; 2) CAT Technology Inc.; 3) Sybase Corp.; 4) BEA Systems Inc; 5) Software House International

Note: Hardware Available and Sun Software Available Now; Sybase Software Available Feb 28, 2001.

Sybase, Inc.

Sun Starfire Enterprise 10000 with Sybase ASE 12.0.0.2

TPC-C 3.5

Report Date: August 29, 2000

Numerical Quantity Summary

MQTh, Computed Maximum Qualified Throughput **= 156,873.03 tpmC**

% throughput difference, reported & reproducibility runs = < 0.1%

Response Times (in secs)	90th Percentile	Average	Maximum
Menu	0.500	0.224	0.701
New-Order	1.600	0.623	40.195
Payment	1.800	0.740	39.073
Order-Status	1.800	0.896	28.956
Delivery(interactive)	0.260	0.245	4.077
Delivery(deferred)	3.000	0.982	18.000
Stock-level	2.000	1.044	17.149

Transaction Mix, in percent of total transactions

New-Order	44.84%
Payment	43.06%
Order-Status	4.02%
Delivery	4.04%
Stock-level	4.04%

Keying/Think Times (in secs)	Average	Min.	Maximum
New-Order	18.015 / 12.199	18.010 / 0	18.081 / 122.00
Payment	3.016 / 12.191	3.010 / 0	3.075 / 122.00
Order-Status	2.016 / 10.261	2.010 / 0	2.074 / 102.50
Delivery	2.016 / 5.201	2.010 / 0	2.043 / 52.00
Stock-level	2.016 / 5.207	2.010 / 0	2.048 / 52.00

Test Duration	
Ramp-up time	40 minutes
Measurement Interval	30 minutes
Number of checkpoints	1
Checkpoint Interval	29:59 minutes
Number of transactions (all types) completed in measurement interval	10,495,490

Preface

This report documents the compliance of the Sun Microsystems TPC Benchmark C testing on the Sun Starfire Enterprise 10000 running Sybase Adaptive Server Enterprise 12.0.0.2 with the *TPC Benchmark C Standard Revision 3.5*

These tests were run using the Sybase Adaptive Server Enterprise 12.0.0.2 RDBMS running with Solaris 7 on the Sun Starfire Enterprise 10000 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.

Additional Copies

Additional copies of this report may be ordered from the administrator of the TPC:

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

The Sun Enterprise 10000 server
TPC Benchmark C Full Disclosure

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 Enterprise 10000 server running the Sybase Adaptive Server Enterprise 12.0.0.2 RDBMS.

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 Benchmark C 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., "Application Code" contains the application source code that implements the transactions and forms modules.

1.2 Benchmark Sponsor

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

This benchmark 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., "Tunable Parameters" 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.

Figure 1 on page 4 is a diagram of the benchmark configuration. Figure 2 on page 5 is a configuration diagram of the priced system.

1.4.1 Configuration Items for the Sun Starfire Enterprise10000

For the priced configuration, the server machine was a Sun Enterprise 10000 server that consisted of the following:

- 64 UltraSPARC II 400 MHz processors.
- 64GB of main memory.

- 1 D1000 Tray (4 x 18GB drives each).
- 6 Sun StorEdge A5200 Arrays (6 * 22 * 18GB each).
- 1 Sun StorEdge T3 Array (8 * 9 * 18GB each).
- 26 Ultra SCSI Dual Port FC-AL host adaptors.
- 1 SCSI-2 host adaptor.
- 12 GB Backup Tape Device.
- 12 100BaseT networks for connecting to the client systems.

For the benchmark configuration, we used the same configuration as above.

For the benchmark configuration, the client machines were 46 Sun Ultra 10 Model 333's each containing:

- One UltraSPARC II 333 MHz processor.
- Bus speed 111 MHz
- 32KB (D+I) internal cache, 512KB external cache.
- 1GB of main memory.
- One internal SCSI-2 controller.
- One internal 9GB SCSI disk.

For the priced configuration, the client machines were 46 Sun Ultra 10 Model 440's each containing:

- One UltraSPARC II 440 MHz processor.
- Bus speed 110 MHz
- 32KB (D+I) internal cache, 2MB external cache.
- 1GB of main memory.
- One internal SCSI-2 controller.
- One internal 9GB SCSI disk.

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 Ultra 10 Model 333's which emulated the database client sessions.

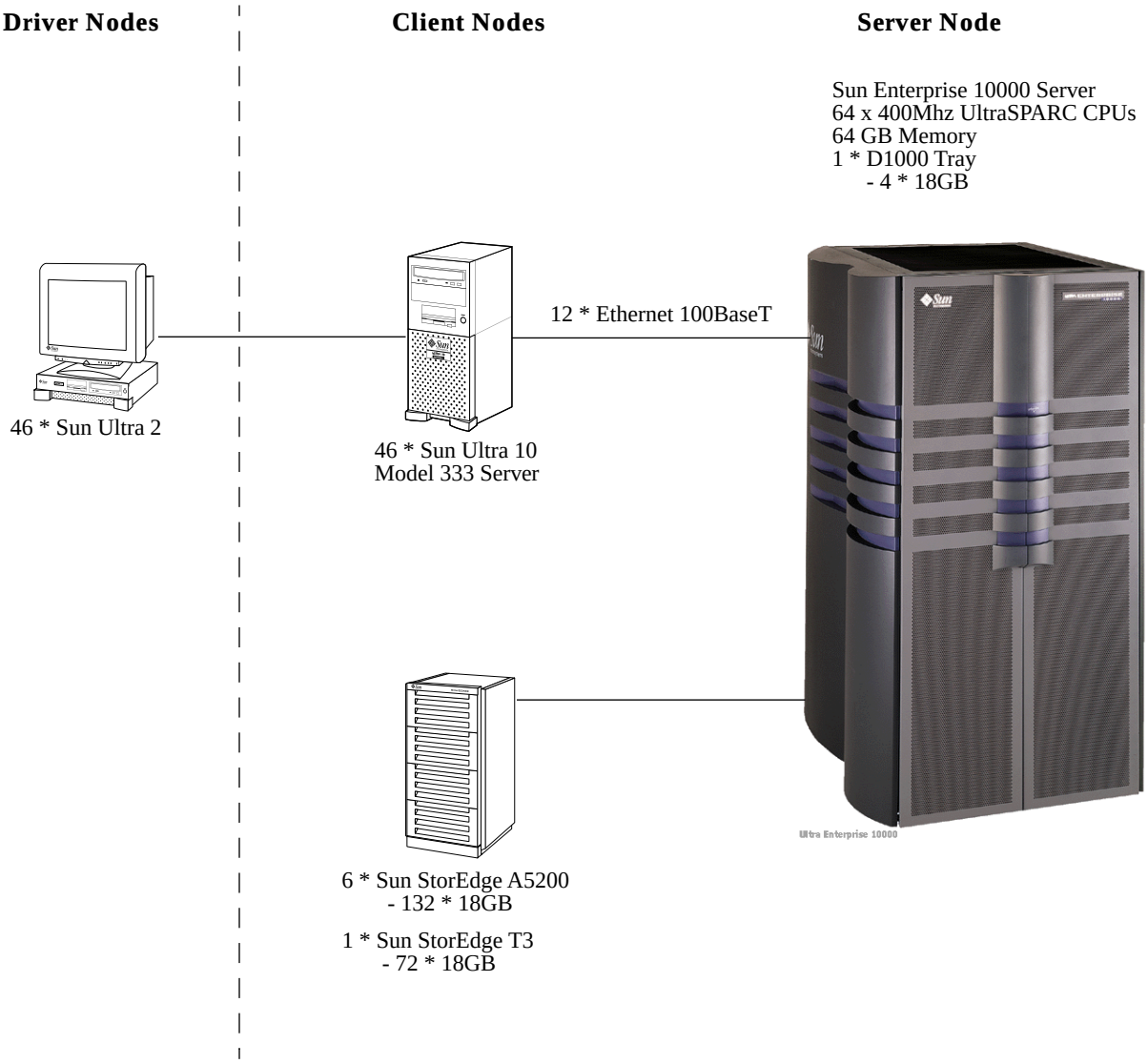


Figure 1 The Sun Enterprise 10000 Benchmark Configuration

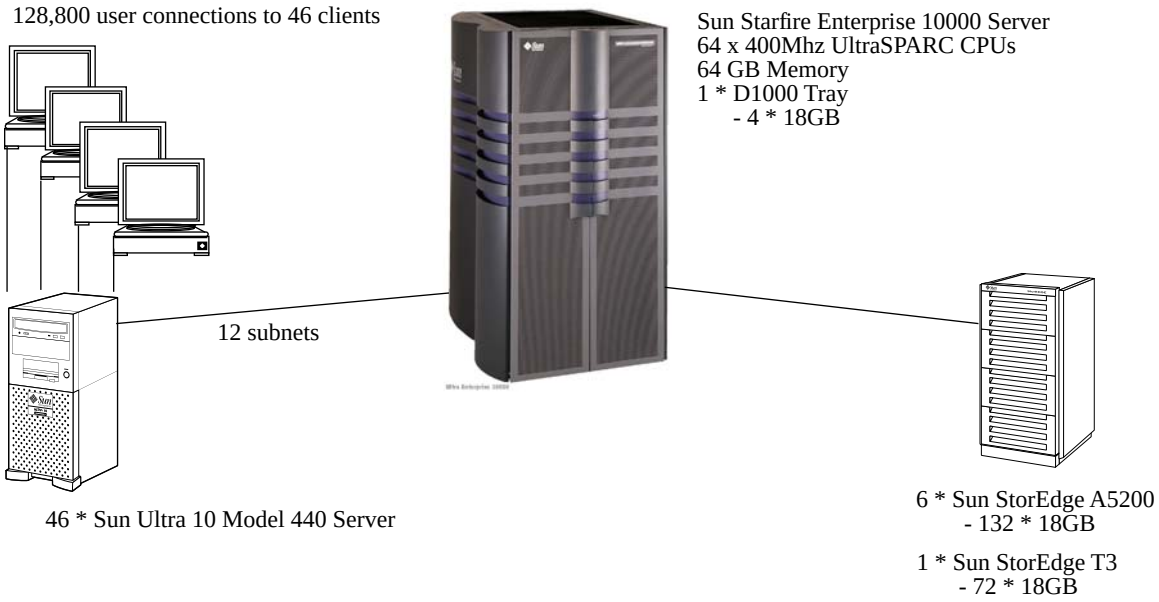


Figure 2 The Sun Enterprise 10000 Priced Configuration.

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., "Database Design" describes the programs that define, create, and populate an Sybase Adaptive Server Enterprise 12.0.0.2 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 12.0.0.2 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., “Screen Layout”.

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 manually exercising each specification on a representative Sun Ultra 10 Model 333 server running Solaris 7.

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 require.

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

TABLE 1. Transaction Statistics

Transaction Type	Statistics	Percentage
Payment	Home warehouse	85.00
	Remote warehouse	15.00
	Non-primary key access	60.00
Order Status	Non-primary key access	60.02
Delivery	Skipped transactions	0.00
Transaction Mix	New order	44.84
	Payment	43.06
	Order status	4.02
	Delivery	4.04
	Stock level	4.04

3.6 Queuing Mechanism

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

Delivery transactions were submitted to servers using the same Tuxedo 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 sqlplus 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 of Clause 3.2.2.
2. A fully-scaled run with a 30-minute steady state period was executed.
3. 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 12.0.0.2 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.

4.5 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. COMMITed 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.

4.6 Isolation Test 2

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

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.

4.7 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.

4.8 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 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. 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.

4.9 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.

4.10 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.

4.11 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 retrieving the prices of all items. The prices of items x and y retrieved matched those retrieved in step 1.
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 (step3) to stall. Therefore, Case A of Clause 3.4.2.7 occurred.

4. Continued terminal 2 transaction. The prices 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.

4.12 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 3000 warehouse database.

4.12.1 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.

4.12.2 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 3000 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 30000 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.

4.12.3 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 30000 terminals. On the driver system, the committed and rolled back New-Order transactions were recorded in a “success” file.
3. After a 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 until 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.

4.12.4 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 12.0.0.2 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.

This database was built with 16,500 warehouses. Prior to running this test, inactive warehouse ids were deleted out of the warehouse table in accordance with clause 4.2.2.

TABLE 2. Cardinality of Tables

Table	Occurrences
Warehouse	12,880
District	165,000
Customer	495,000,000
History	495,000,000
Orders	495,000,000
New order	148,500,000
Order line	4,950,000,000
Stock	1,650,000,000
Item	100,000

5.2 Database Layout

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

The distribution of database tables over the 792 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 on page 5 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 (8hr)	2	Each side of the mirror is striped across 4 LUNs, 9 disks per LUN
customer	72	Each striped across 18 physical disks
stock	100	Each striped across 18 physical disks
order-line	40	Each striped across 36 physical disks
order	8	Each striped across 18 physical disks
master	1	Each striped across 9 physical disks
wdino	4	Each striped across 18 physical disks
history	7	Each striped across 9 physical disks

The data was striped evenly across 792 disks located in 6 A5200 StorEdge cabinets, each with 6 22 disk arrays of 18 GB disks. An additional 4x18 GB disks located in a StorEdge D1000 tray were used for the Operating System, swap disks, and Sybase binaries.

The logs and mirrors were located in a T3 StorEdge cabinet containing 8 bricks, each with nine 18 GB drives.

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 12.0.0.2 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 or partitioning 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., “Disk Storage”.

6. Clause 5 Related Items

6.1 Measured tpmC

Measured tpmC must be reported.

The measured tpmC was 156,873.03.

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
Menu	0.224	0.701	0.500
New-Order	0.623	40.195	1.600
Payment	0.740	39.073	1.800
Order-Status	0.896	28.956	1.800
Interactive Delivery	0.245	4.077	0.260
Deferred Delivery	0.982	18.000	3.000
Stock-Level	1.044	17.149	2.000

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	18.015	18.010	18.081
Payment	3.016	3.010	3.075
Order-Status	2.016	2.010	2.074
Interactive Delivery	2.016	2.010	2.043
Stock-Level	2.016	2.010	2.048

TABLE 6. Think Times

Type	Average	Minimum	Maximum
New-Order	12.199	0.000	122.000
Payment	12.191	0.000	122.000
Order-Status	10.261	0.000	102.500
Interactive Delivery	5.201	0.000	52.000
Stock-Level	5.207	0.000	52.000

6.4 Response Time Frequency Distribution Curves

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

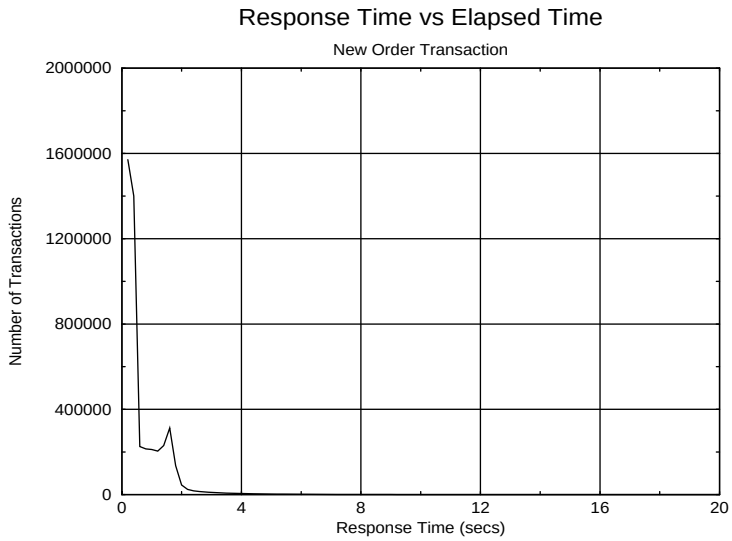


Figure 3 New Order Response Time Distribution

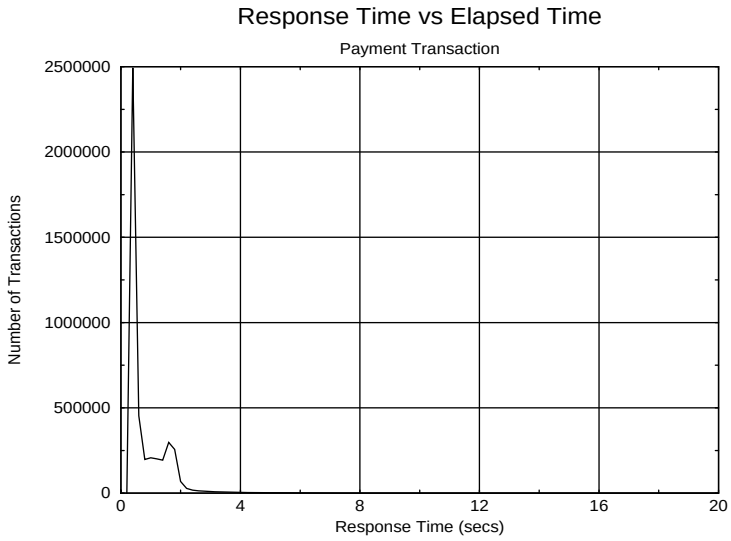


Figure 4 Payment Response Time Distribution

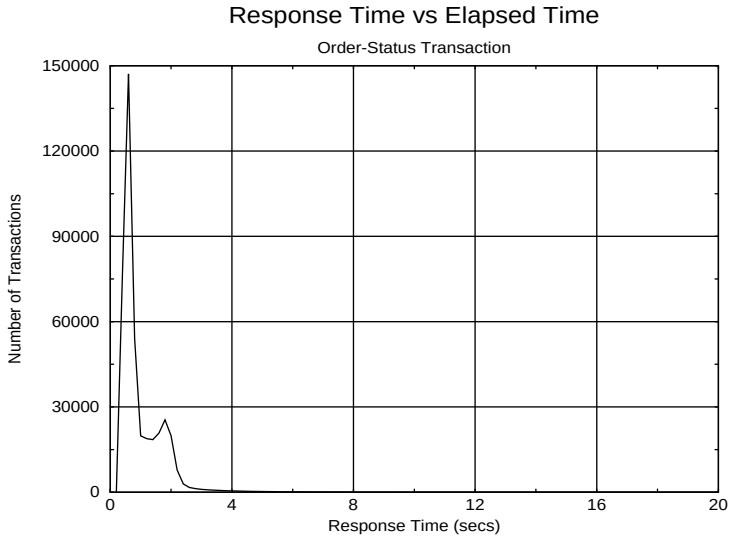


Figure 5 Order Status Response Time Distribution

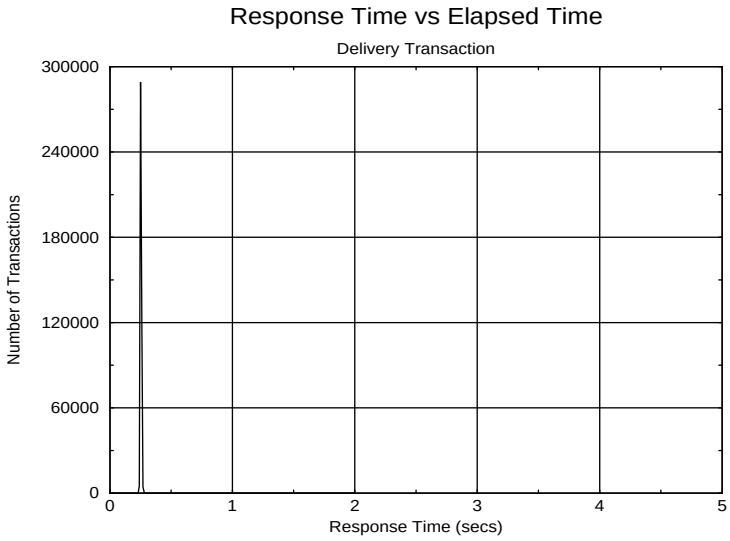


Figure 6 Delivery Response Time Distribution

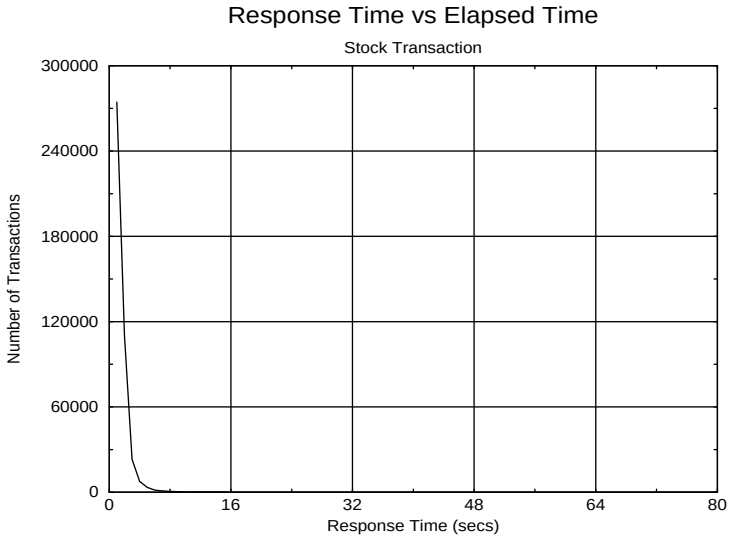


Figure 7 Stock Level Response Time Distribution

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.

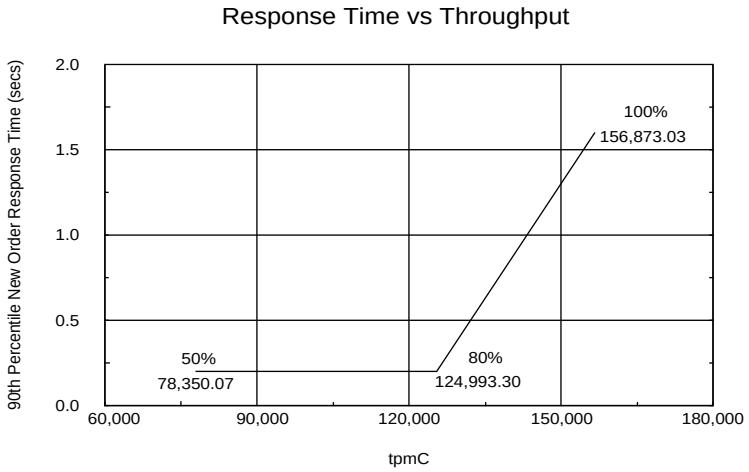


Figure 8 Response Time versus Throughput

6.6 Think Time distribution curves

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

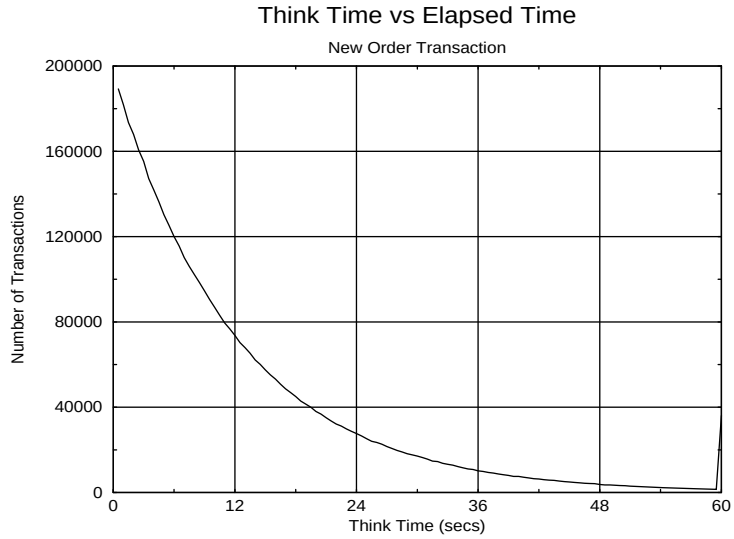


Figure 9 New Order Think Time Distribution

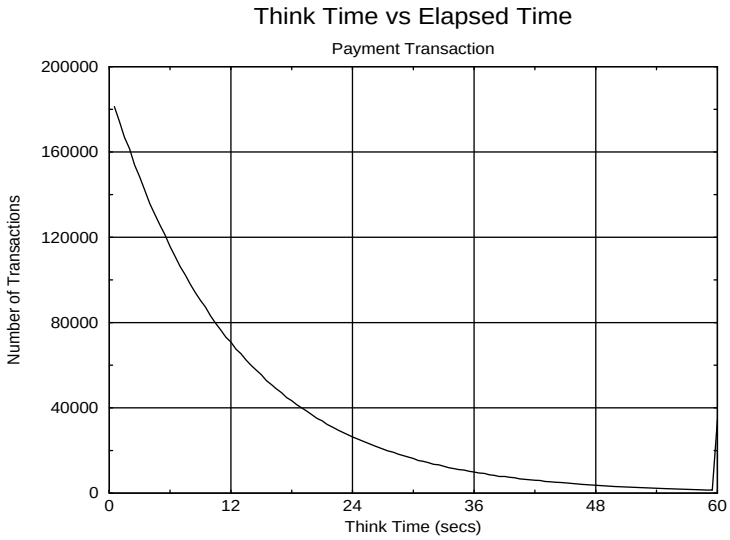


Figure 10 Payment Think Time Distribution

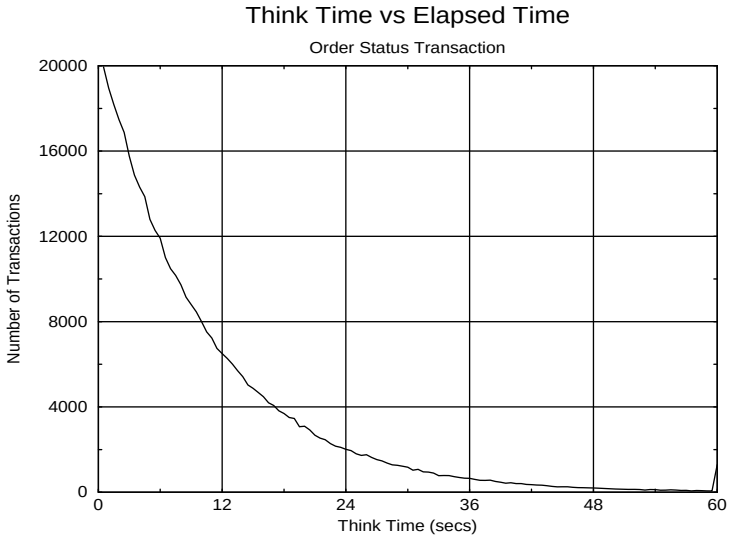


Figure 11 Order Status Think Time Distribution

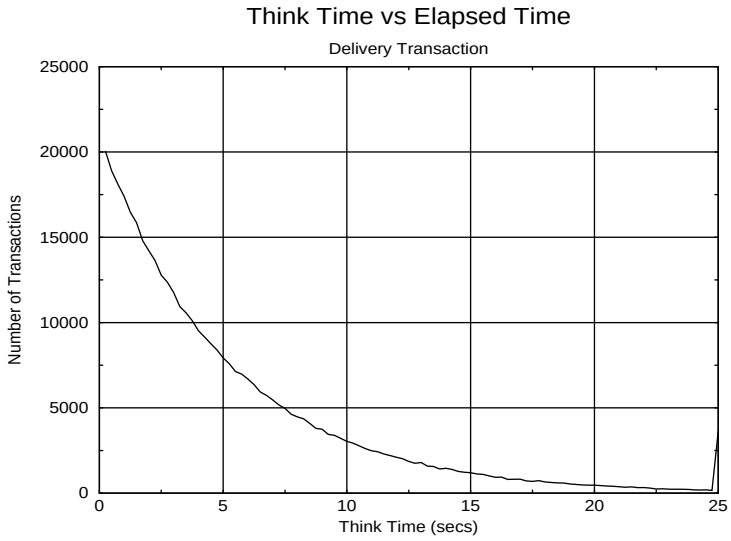


Figure 12 Delivery Think Time Distribution

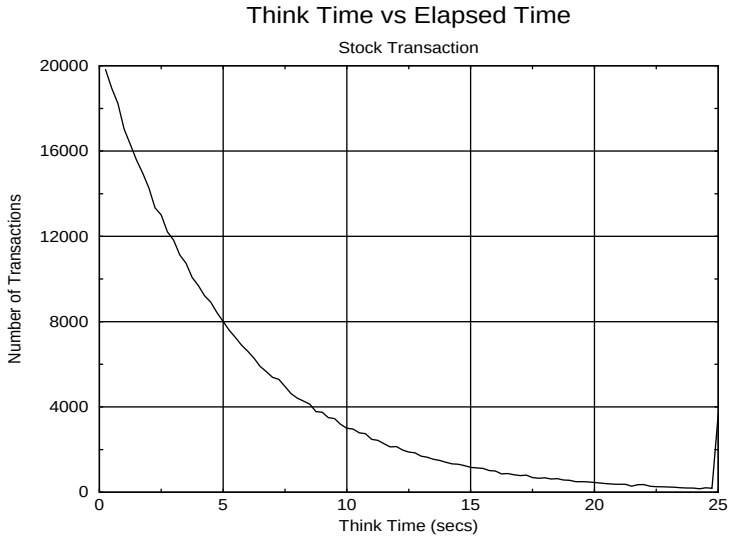


Figure 13 Stock Level Think Time Distribution

6.7 Throughput versus Elapsed Time

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

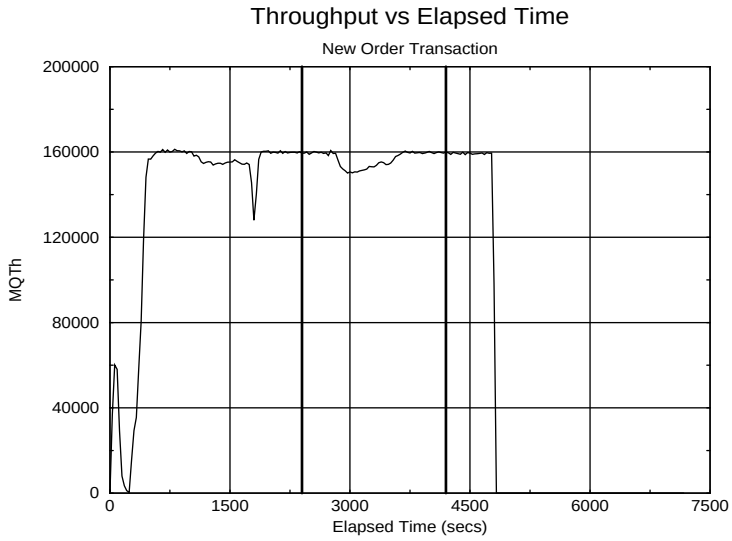


Figure 14 Throughput versus Time

6.8 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 in Figure 14.

6.9 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.

6.9.1 Checkpoint

A Sybase Adaptive Server Enterprise 12.0.0.2 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.

6.10 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 156,647.97 tpmC was achieved.

6.11 Measurement Period Duration

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

The measurement interval was 30 minutes.

6.12 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.

6.13 Numerical Results

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

See Table 1 on page 7 for results.

6.14 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 on page 7 for results.

6.15 Order-Line Average

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

See Table 1 on page 7 for results.

6.16 Remote Order-Lines

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

See Table 1 on page 7 for results.

6.17 Remote Payments

The percentage of remote payment transactions must be disclosed.

See Table 1 on page 7 for results.

6.18 Customer Lastname

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

See Table 1 on page 7 for results.

6.19 Deliveries 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 on page 7 for results.

6.20 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 1220 seconds after the start of ramp-up and one at 511 seconds after the start of the measurement interval. The interval between the two checkpoints was 29:59 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 Computer Company 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         2400    # ramp_up interval (secs)
setenv  stdy_state      1800    # steady-state/measurement interval (secs)
setenv  ramp_down       600     # ramp_down interval (secs)
setenv  trigger_time    2100    # Trigger time for users to login
setenv  scale           12880   # of warehouses
set users = ( 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800
2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800
2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800 2800
2800 )
set rte_machines =  ( r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r14 r15 r16 r17 r18
r19 r20 r21 r22 r23 r24 r25 r26 r27 r28 r29 r30 r31 r32 r33 r34 r35 r36 r37 r38
r39 r40 r41 r42 r43 r44 r45 r46 r47 r48 )
set clnt_machines = ( c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c14 c15 c16 c17 c18
c19 c20 c21 c22 c23 c24 c25 c26 c27 c28 c29 c30 c31 c32 c33 c34 c35 c36 c37 c38
c39 c40 c41 c42 c43 c44 c45 c46 c47 c48 )
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., “Driver Scripts”.

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

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 system and 12 100BaseT LAN's connecting all the 46 client systems to the server. There were 2800 workstations "terminals" on each.

7.5 Operator Intervention

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

The Sun Enterprise 10000 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 v 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., “Price 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 Gold and Silver Programs of the SunService Support Program were 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.

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 contract discount
- a 5% Sun support pre-payment discount
- a 10% Sybase service 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 are available now. Sybase Adaptive Server Enterprise 12.0.0.2 will be available February 28, 2001.

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 Sun Starfire Enterprise 10000 was 156,873.03 tpmC at $48.81 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 letter indicating compliance must be included in the Full Disclosure Report.

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsors:

Brad Carlile
Mgr., Strategic Appl. Engineering
Sun Microsystems, Inc.
8305 SW Creekside Pl.
Beaverton, OR 97008

Prasanta Ghosh
Mgr., S/W Engineering Devlpmt.
Sybase, Inc.
1650 65th St
Emeryville, Ca 94608

August 28, 2000

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

Platform:	**Sun Starfire Enterprise 10000 c/s**
Operating system:	**Solaris 7**
Database Manager:	**Sybase ASE 12.0.0.2**
Transaction Manager:	**Bea Tuxedo 6.3**

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	**tpmC**
Server: Sun Starfire Enterprise 10000				
64 x UltraSPARC II (400 MHz)	64 GB (8 MB L2 Cache per processor)	4 x 18 GB SCSI-2 864 x 18 GB FC-AL	1.6 Seconds	**156,873.03**
Fourtysix (46) Clients: Ultra 10 Model 333 (specification for each)				
1 x UltraSPARC II (333 MHz)	1 GB	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

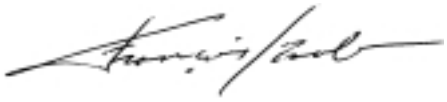
1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office: 719/473-7555 • Fax: 719/473-7554

- ¥ 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 30 minutes
- The reported measurement interval was representative of steady state conditions
- One checkpoints was 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 (46) SUN Ultra 10 Model 333 clients with 333Mhz processors that were substituted by (46) SUN Ultra 10 Model 440 clients with 440Mhz processors in the priced configuration. Based on the specifications of these client systems and on the fact the bus speed for the client systems is 111Mhz and 110Mhz, respectively, it is my opinion that this substitution does not have a material effect on the reported performance.

Respectfully Yours,



François Raab
President

1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office: 719/473-7555 • Fax: 719/473-7554

Appendix A. Application Code

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

tpcc_client.c

```
/*************************************************************************
tpcc_client.c
*************************************************************************/
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/time.h>
#include<sys/procset.h>
#include<sys/priocntl.h>
#include<sys/rtpriocntl.h>
#include<sys/param.h>
#include<limits.h>
#include<errno.h>

#include"priocntl.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;
display_screen(num);
status = get_inputs(num);
if (status == 3)
return;

if (Snd_Txn_To_Monitor(num)){
 syserr("\033[24;1H\033[mTPM Error detected -- See $TPCC_HOME/CLIENTLOG for
details\n");
return;
}

display_output(num);
}
```

tpcc_client.h

```
/*************************************************** ********************
***************************** tpcc_client.h ***********************************
*************************************************** ********************/
#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;
/*
 * * Data structures of input data for each transaction type
 */
/*
 * * Data structures descriptions for IO data for each transaction 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;
doubleol_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 {
intw_id;
intd_id;
};
/*
 * * Data structure for input & output data
 */
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_tinfo;
};
```

tpcc_forms.c

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

#pragma ident "@(#)tpcc_forms.c1.297/07/15SMI"
#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()
{
/** put screen in raw mode **/

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()
{/** restore terminal flags **/
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;
floatfloat_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;
intblank_line = 0, i, j;
intblank_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, 5, 3, j+FIRST_OL_ROW, ol_ptr2->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)
/* shishir - changed for 11k wid
 *bufp += DISPLAY_INT(bufp, 4, 12, 4, w_id);
 */
bufp += DISPLAY_INT(bufp, 5, 12, 4, w_id);
else if (txn_type != 5)
/* shishir - changed for 11k wid
 *bufp += DISPLAY_INT(bufp, 4, 12, 2, w_id);
 */
bufp += DISPLAY_INT(bufp, 5, 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;
/* Position cursor at start of field */

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) {

/*
 * Below we read from standard input into the array curbuf.
 * curbuf_read is the pointer to the array curbuf indicating
 * the position upto which the curbuf has been parsed.
 * curbuf_consumed is the number of elements in the buffer
 * temp that holds the array that is to be displayed.
 * Elements of curbuf_consumed is selectively copied from
 * curbuf Note:read_count is the total number of characters
 * in the buffer curbuf. curbuf_read is always less than or
 * equal to read_count.
 */

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");
}
/*
 * int message prevents unnecessary display of warning
 * messages
 */

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);
/*
 * bufp = screen_buf;
 */
}
}
}/* if previous_data_exists */
while ((curbuf_read < read_count) && (exit_read_function == FALSE)) {
/*
 * intermediate variable size1 for cases when
 * floating point field whose size is less than
 * actual size by 1 because of decimal.
 */
if (payment_input == TRUE)
size1 = size - 1;
/*
 * Test for integer
 */
if ((curbuf[curbuf_read] >= '0' && curbuf[curbuf_read] <= '9') ||
(curbuf[curbuf_read] == '.')) {
/*
 * Consume all integers in buffer
 */
for (; curbuf_read < read_count &&
     ((curbuf[curbuf_read] >= '0'
     && curbuf[curbuf_read] <= '9') || curbuf[curbuf_read] == '.'); curbuf_read++) {
/*
 * below we fill up temp making sure
 * the size limit is not exceeded
 */
if (curbuf_consumed < size1) {
temp[curbuf_consumed] = curbuf[curbuf_read];
curbuf_consumed++;
}
/*
 * number of elements typed in is
 * more than the size of the field
 */
else
OVERFLOW = TRUE;
/*
 * ensure the character is removed
 * after it is read
 */
curbuf[curbuf_read] = '\0';
}/* end of for curbuf is legitimate
 * number */
temp[curbuf_consumed] = '\0';/* terminate temp string */
if (payment_input == TRUE) {/* floating point field */
/* convert the ascii to float */
q = (atof(temp)) ;
bufp += DISPLAY_FLOAT(bufp, 2, (col + size - 4), row, q);
/*
if (curbuf_consumed < 3)
else
bufp += DISPLAY_FLOAT(bufp, 2, (col + size - curbuf_consumed - 1), row, q);
*/
} else {
if (curbuf_consumed < size + 1)
bufp += DISPLAY(bufp, (col + size -
      curbuf_consumed), row,
temp);
return_status = curbuf[curbuf_read];
cur_col++;
}
}
/* if curbuf[] between "0" and "9" */
/*
 * if not integer, then test for movement character
 */
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;
}
/* if curbuf[] a movement character */
/*
 * if not integer of movement, test for DELETE
 */
else if (curbuf[curbuf_read] == DELETE) {
if (payment_input == TRUE) {/* for floating point
 * field */
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++;/* pressed key overflow
 * situations */
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);
}
}
/* end of if DELETE */
 /* could be a ^C */
else if (curbuf[curbuf_read] == QUIT) {
temp[0] = '\0';
return_status = QUIT;
curbuf[curbuf_read] = '\0';
exit_read_function = TRUE;
} else {
/** Any other character entered at the keyboard ... **/
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++;
}
}/** End of the WHILE loop **/
if (OVERFLOW == TRUE && exit_read_function == FALSE) {
/*
 * if number of characters are exceeding the field
 * limit beep and warning message is necessary
 */
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;
}
}
}
/* ensuring unnecessary warning messages are removed */
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;
/* Position cursor at start of field */

if (curbuf_read == read_count || curbuf_read == 0) {
screen_buf[0] = '\0';
bufp += GOTOXY(bufp, col, row);/* Goto input area */
PAINTSCRLEN(screen_buf, bufp - screen_buf);
bufp = screen_buf;
}
if ((*(char *) data) != '\0')
previous_data_exists = TRUE;
while (exit_read_function == FALSE) {
/*
 * Below we read from standard input into the array curbuf.
 * curbuf_read is the pointer to the array curbuf indicating
 * the position upto which the curbuf has been parsed.
 * curbuf_consumed is the number of elements in the buffer
 * temp that holds the array that is to be displayed.
 * Elements of curbuf_consumed is selectively copied from
 * curbuf Note:read_count is the total number of characters
 * in the buffer curbuf. curbuf_read is always less than or
 * equal to read_count.
 */
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] <= '~') {/** if between ASCII
space (040) through ~ (0176) **/
for (; curbuf[curbuf_read] >= ' '
     && curbuf[curbuf_read] <= '~'; curbuf_read++) {
/*
 * ensuring the curbuf_consumed is
 * not more than field size
 */
if (curbuf_consumed < size) {
temp[curbuf_consumed] = curbuf[curbuf_read];
curbuf_consumed++;
}
/* else overflow condition */
else
OVERFLOW = TRUE;
curbuf[curbuf_read] = '\0';/* erasing characters
 * already read from the
 * buffer */
}
temp[curbuf_consumed] = '\0';/* terminate temp string */
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 {/** Any other character entered at the keyboard ... **/
if (message == FALSE) {
bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW, INVALID_MSG);
bufp += GOTOXY(bufp, col, row);
message = TRUE;
}
curbuf_read++;
}
}/** End of the WHILE loop **/
if (OVERFLOW == TRUE && exit_read_function == FALSE)
/** If read enough to fill the size already **/
{
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);
}
voiddisplay_newo();
voiddisplay_paym();
voiddisplay_ords();
voiddisplay_del();
voiddisplay_stock();
void(*p_print_function[]) () = {
&display_newo,
&display_paym,
&display_ords,
&display_del,
&display_stock
};
display_output(int txn_type)
{
charc;

(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);
}
#ifdef DEBUG
Clog("DBG: Screen output chars = %d\n", (bufp - &output_screen[0]));
#endif
}
void
display_paym()
{
char           *bufp, temp[51], tempbuf2[201];
char           *make_phone(char *), *make_zip(char *);
bufp = output_screen;
bufp += SWITCH_TO_NORMAL(bufp);/* jr */
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);
#ifdef DEBUG
Clog("DBG: Screen output chars = %d\n", (bufp - &output_screen[0]));
#endif
}
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);
#ifdef DEBUG
Clog("DBG: Screen output chars = %d\n", (bufp - &output_screen[0]));
#endif
}
void
display_del()
{
char           *bufp;

bufp = output_screen;
/*PAINTSCR(DELIVERY_QUEUED_MSG);*/
bufp += sprintf(bufp,"%s",DELIVERY_QUEUED_MSG);
bufp += DISPLAY(bufp, 23, 75, "**((");
*bufp++ = '\0';
PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
Clog("DBG: Screen output chars = %d\n", (bufp -
 &output_screen[0]));
#endif
}
void
display_stock()
{
char           *bufp;
bufp = output_screen;
bufp += SWITCH_TO_NORMAL(bufp);/* jr */

bufp += DISPLAY_INT(bufp, 3, 12, 6, oSL->low_stock);
bufp += DISPLAY(bufp, 23, 75, "**((");
*bufp++ = '\0';
PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
Clog("DBG: low stock:%d\n", oSL->low_stock);
Clog("DBG: Screen output chars = %d\n", (bufp -
 &output_screen[0]));
#endif
}
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;
}
```

tpcc_forms.h

```
/*****************************************************************************
tpcc_forms.h
*****************************************************************************/

#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'

/** Possible status values returned by read functions **/

#define CANCELLED 3
#define PREVIOUS_FIELD 4

/** 52 Possible key strokes read in by the read functions. Some are also
returned as status from the read functions.  **/

#define BACKTAB 2/** Decided to use the CTRL B for now **/
#define DELETE 8
#define ESCAPE 27
#define LF 10
#define QUIT 3/** CNTRL-C Key stroke to quit for now **/
#define SPACE 32
#define SUBMIT 13/** Done with screen and submit key: CR **/

#define TAB 9
#define UNDERLINE 95
#define LEAVE_SCREEN_MIN 300/** Minimum # of characters
to leave screen **/
#define LEAVE_SCREEN_TIMEOUT 2/** Minimum time to leave
screen, 10=1sec **/
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;/* for suppressing warning messages */
BOOLEAN         payment_input = FALSE;/* for setting money condition in
 * read_int */
static struct termio tbufsave;
extern void     syserr();
void            Init_Screen();
void            display_screen_array(int);
void            Send_Menu();
int             Get_Menu_Input();
/**
The following is the struct type used to define the I/O element
y is the row position on the screen.
x is the column position on the screen.
len is the size of the data field in bytes.
flags is the indicator of manadatory data fields. '1' means required.
dptr is the pointer to the data.
fptr is the pointer to the read funtion for the type of data dptr
points to.
**/
typedef struct {
int             y;
int             x;
int             len;
int             flags;
int            *dptr;
int             (*fptr) ();
}               io_elem;
/*
 * Temporary structures for storing data that needs manipulation before
 */
int             int_h_amount;
/* All the possible messages to print out */
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";/* jr */
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)";

intmenu_buflen = sizeof (menu_buf);
io_elem         neworder_inputs[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ---------- ------------------- */
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[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
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[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
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[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
4, 17, 2, 0, 0, &read_integer,
999
};
io_elem         stocklev_inputs[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
4, 24, 2, 0, 0, &read_integer,
999
};
io_elem         wd_inputs[] = {
/* y x len flags ptr to data ptr to read function */
/* - - --- ----- ----------- ------------------- */
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
};
#ifdefMultiple_blank_form
const char WD_blank_form[SCRBUF_LEN] =
CLRSCN_STR/**/DISPLAY_STR(2,1,'Warehouse:')/**/DISPLAY_STR(2,26,'District:')";
#endif
struct form_info {
const text_elem      *tp;
char            *blank_form;
intblank_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}
};
```

tpcc_tux.c

```
/************************************************** **********************
*************************** monitor.c ************************************
************************************************************ **********************/
/* ** monitor.c -- All functions for Tuxedo call and return */
#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)
{

#if DEBUG
Clog("DBG: In Snd_Txn_To_Monitor\n");
print_input_data(txn_type);
#endif

if (txn_type == DELIVERY) {

if ( tpacall((char *)svc_names[txn_type], tuxibuf, ilen, TPNOREPLY) == -1){

Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
     svc_names[txn_type], tpstrerror(tperrno));

return (-100);
}
return;

} else {

if (tpcall((char *)svc_names[txn_type],
(char *)tuxibuf, ilen,
&tuxobuf, &olen, 0) == -1){
```

```
Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
svc_names[txn_type], tpstrerror(tperrno));

return(-100);
}
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();

#ifdef DEBUG
Clog("terminated Tuxedo connection with status %d\n", status);
#endif
}
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");
}
#ifdef DEBUG
print_input_data(int type)
{
int             i;
time_t          the_time;
the_time = time(&the_time);
Clog("DBG:=============TIME: %s == == == == == == ==\n",ctime(&the_time));
     switch (type) {
case NEWORDER:
Clog("DBG: NEWORDER INPUTS at %s\n", ctime(&the_time));
Clog("DBG: w_id: %d, d_id: %d, c_id: %d o_ol_cnt: %d\n",
     iNO->w_id, iNO->d_id, iNO->c_id, iNO->o_ol_cnt);
for (i = 0; i < iNO->o_ol_cnt; i++)
Clog("DBG: ol_i_id: %d, ol_supply_w_id: %d, ol_quantity: %d \n ",iNO-
>o_ol[i].ol_i_id, iNO->o_ol[i].ol_supply_w_id,iNO->o_ol[i].ol_quantity);
break;
case PAYMENT:
Clog("DBG: PAYMENT INPUTS at %s \n ",ctime(&the_time));
     Clog("DBG: w_id: %d, d_id: %d\n", iPT->w_id, iPT->d_id);
Clog("DBG: c_last: %s ", iPT->c_last);
Clog(" c_id: %d", iPT->c_id);
Clog(" c_w_id: %d, c_d_id: %d\n", iPT->c_w_id, iPT->c_d_id);
Clog("DBG: h_amount: %f\n", iPT->h_amount);
break;
case ORDSTAT:
Clog("DBG: ORDER STATUS INPUTS at %s \n ",ctime(&the_time));
     Clog("DBG: w_id: %d, d_id: %d\n", iOS->w_id, iOS->d_id);
Clog("DBG: c_id: %d, c_last: %s\n",
     iOS->c_id, iOS->c_last);
break;
case DELIVERY:
Clog("DBG: DELIVERY INPUTS at %s\n", ctime(&the_time));
Clog("DBG: w_id: %d, o_carrier_id: %d\n", iDY->w_id, iDY ->o_carrier_id);
break;
case STOCKLEV:
Clog("DBG: STOCK LEVEL INPUTS at %s \n ",ctime(&the_time));
Clog("DBG: w_id: %d, d_id: %d, threshold: %d\n", iSL ->w_id, iSL->d_id,iSL-
>threshold);
break;
other:
Clog("DBG: Txn_type = %d is illegal at %s \n",type,ctime(&the_time));
     }
return;
}
#endif/* ifdef DEBUG */
```

tpcc_tux.h

```
/*****************************************************************************
***************************** monitor.h ***************************************
*****************************************************************************/
/* ** monitor.h -- All Tuxedo definitions and storage ** */
long            ilen;
long            olen;
int tty_in;
int tty_out;

/*
 * Data is returned in tuxibuf. Some #defines are set up here for easy
 * access/naming of the input and output areas
 */
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.sh

```
#!/bin/sh -f
###############################################################################
#
# 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!
#
#
###############################################################################
#

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

CREATE PROC neworder_local (
@w_idsmallint,
@d_idtinyint,
@c_idint,
@o_ol_cntint,

@i_idint = 0, @ol_qtysmallint = 0,
@i_id2int = 0, @ol_qty2smallint = 0,
@i_id3int = 0, @ol_qty3smallint = 0,
@i_id4int = 0, @ol_qty4smallint = 0,
@i_id5int = 0, @ol_qty5smallint = 0,
@i_id6int = 0, @ol_qty6smallint = 0,
@i_id7int = 0, @ol_qty7smallint = 0,
@i_id8int = 0, @ol_qty8smallint = 0,
@i_id9int = 0, @ol_qty9smallint = 0,
@i_id10int = 0, @ol_qty10smallint = 0,
@i_id11int = 0, @ol_qty11smallint = 0,
@i_id12int = 0, @ol_qty12smallint = 0,
@i_id13int = 0, @ol_qty13smallint = 0,
@i_id14int = 0, @ol_qty14smallint = 0,
@i_id15int = 0, @ol_qty15smallint = 0
)
as

declare
@w_taxreal,@d_taxreal,
@c_lastchar(16),@c_creditchar(2),
@c_discountreal,@commit_flagint,
@c_ins_id int,@local_d_idint,

@i_pricefloat,
@i_namechar(24),@i_datachar(50),

@s_quantitysmallint,@ten_smallintsmallint,
@s_ytdint,@s_order_cntint,
@s_distchar(24),@s_datachar(50),
@one_smallintsmallint,@zero_smallint smallint,
@ninenine_smallint smallint,

@ol_numberint,@o_idint,
@o_entry_ddatetime,@b_gchar(1),
@ol_amount float



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
, @local_d_id= @d_id
```

```
, @ten_smallint= 10
, @zero_smallint = 0
, @ninenine_smallint= 99
, @one_smallint= 1
, @o_entry_d= getdate()
WHEREd_w_id= @w_id
 ANDd_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

      /* set i_id, ol_qty for this lineitem */
      /* this is replaced by case statement */

      /* convert c_no_is cursor to a simple select */
      /* get item data (no one update item) */

      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
         /*Otherwise if the item is found */
         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 < @ten_smallint)
                        then @ninenine_smallint else @zero_smallint end,
                    s_quantity = s_quantity - @ol_qty +
                        case when (s_quantity - @ol_qty < @ten_smallint)
                        then @ninenine_smallint  else @zero_smallint end,
           s_order_cnt = s_order_cnt + @one_smallint,
                    @s_data = s_data,
                    @s_dist = case @local_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
       if (@@rowcount = 0)
           begin
             select @commit_flag = 0
             select NULL, NULL, NULL, NULL, NULL
             continue
   end
        /*Otherwise if the Stock is found */

  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, "18000101", @ol_qty,
       @ol_amount, @s_dist)


         /* send line-item data to client */
         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 /* while */

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/* Return to client */
@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_idsmallint,
@d_idtinyint,
@c_idint,
@o_ol_cntint,

@i_idint = 0, @s_w_idsmallint = 0, @ol_qtysmallint = 0,
@i_id2int = 0, @s_w_id2smallint = 0, @ol_qty2smallint = 0,
@i_id3int = 0, @s_w_id3smallint = 0, @ol_qty3smallint = 0,
@i_id4int = 0, @s_w_id4smallint = 0, @ol_qty4smallint = 0,
@i_id5int = 0, @s_w_id5smallint = 0, @ol_qty5smallint = 0,
@i_id6int = 0, @s_w_id6smallint = 0, @ol_qty6smallint = 0,
@i_id7int = 0, @s_w_id7smallint = 0, @ol_qty7smallint = 0,
@i_id8int = 0, @s_w_id8smallint = 0, @ol_qty8smallint = 0,
@i_id9int = 0, @s_w_id9smallint = 0, @ol_qty9smallint = 0,
@i_id10int = 0, @s_w_id10smallint = 0, @ol_qty10smallint = 0,
@i_id11int = 0, @s_w_id11smallint = 0, @ol_qty11smallint = 0,
@i_id12int = 0, @s_w_id12smallint = 0, @ol_qty12smallint = 0,
@i_id13int = 0, @s_w_id13smallint = 0, @ol_qty13smallint = 0,
@i_id14int = 0, @s_w_id14smallint = 0, @ol_qty14smallint = 0,
@i_id15int = 0, @s_w_id15smallint = 0, @ol_qty15smallint = 0
)
as

declare
@w_taxreal,@d_taxreal,
@c_lastchar(16),@c_creditchar(2),
@c_discountreal,@commit_flagtinyint,
@c_ins_id int,@local_d_idint,

@i_pricefloat,
@i_namechar(24),@i_datachar(50),

@s_quantitysmallint,@ten_smallintsmallint,
@s_ytdint,@s_order_cntint,
@s_distchar(24),@s_datachar(50),
@one_smallintsmallint,@zero_smallint smallint,
@ninenine_smallint smallint,

@ol_numberint,@o_idint,
@o_entry_ddatetime,@b_gchar(1),
@ol_amount float


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
, @local_d_id= @d_id
, @ten_smallint= 10
, @zero_smallint = 0
, @ninenine_smallint= 99
, @one_smallint= 1
, @o_entry_d= getdate()
WHEREd_w_id= @w_id
  ANDd_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

 /* convert c_no_is cursor to a simple select */
 /* get item data (no one update item) */
 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
  /* Otherwise if the item is found */
     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 < @ten_smallint)
               then @ninenine_smallint else @zero_smallint end,
          s_quantity = s_quantity - @ol_qty +
               case when (s_quantity - @ol_qty < @ten_smallint)
               then @ninenine_smallint  else @zero_smallint end,
   s_order_cnt = s_order_cnt + @one_smallint,
          @s_data = s_data,
          @s_dist = case @local_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_remote_cnt = s_remote_cnt +
   case when (@s_w_id = @w_id)
                       then 0 else 1 end
             where s_w_id = @s_w_id and
                   s_i_id = @i_id

             if (@@rowcount = 0)
               begin
                 select @commit_flag = 0
                 select NULL, NULL, NULL, NULL, NULL
                 continue
               end

  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,
       @s_w_id, "18000101", @ol_qty,
       @ol_amount, @s_dist)

        /* send line-item to client */
        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 /* while */


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/* Return to client */
@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_idsmallint,@c_w_idsmallint,
@h_amount float,
@d_idtinyint,@c_d_idtinyint,
@c_idint
as
declare@c_lastchar(16)

declare@w_street_1char(20),@w_street_2char(20),
@w_citychar(20),@w_statechar(2),
@w_zipchar(9),@w_namechar(10),
@w_ytd float,@w_id_retrieved smallint

declare@d_street_1char(20),@d_street_2char(20),
@d_citychar(20),@d_statechar(2),
@d_zipchar(9),@d_namechar(10),
@d_ytd float,@commit_flgint

declare@c_firstchar(16),@c_middlechar(2),
@c_street_1char(20),@c_street_2char(20),
@c_citychar(20),@c_statechar(2),
@c_zipchar(9),@c_phonechar(16),
@c_sincedatetime,@c_creditchar(2),
@c_credit_limnumeric(12,0),@c_balancefloat,
@c_discountreal,
@data1char(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
   ,@commit_flg = 1
WHERE d_w_id= @w_id
ANDd_id= @d_id

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

if (@@rowcount = 0)
   begin
select @commit_flg = 0
   end

/* Customer data */
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 (@@rowcount = 0)
   begin
select @commit_flg = 0
   end

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 /* if */

/* Create the history record */
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_retrieved,
@today, @h_amount, (@w_name + "    " + @d_name))

/* COMMIT TRANSACTION PID */

if (@commit_flg = 1)
             COMMIT TRANSACTION PID
          else
      ROLLBACK TRANSACTION PID

select/* Return to client */
@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_idsmallint,@c_w_idsmallint,
@h_amount float,
@d_idtinyint,@c_d_idtinyint,
@c_lastchar(16)
as
declare@nint,@c_idint

declare@w_street_1char(20),@w_street_2char(20),
@w_citychar(20),@w_statechar(2),
@w_zipchar(9),@w_namechar(10),
@w_ytd float,@w_id_retrieved smallint

declare@d_street_1char(20),@d_street_2char(20),
@d_citychar(20),@d_statechar(2),
@d_zipchar(9),@d_namechar(10),
@d_ytd float,@commit_flgint

declare@c_firstchar(16),@c_middlechar(2),
@c_street_1char(20),@c_street_2char(20),
@c_citychar(20),@c_statechar(2),
@c_zipchar(9),@c_phonechar(16),
@c_sincedatetime,@c_creditchar(2),
@c_credit_limnumeric(12,0),@c_balancefloat,
@c_discountreal,
@data1char(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
   ,@commit_flg = 1
WHERE d_w_id= @w_id
ANDd_id= @d_id

if (@@rowcount = 0)
   begin
select @commit_flg = 0
   end

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

/* Customer data */
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 (@@rowcount = 0)
   begin
select @commit_flg = 0
   end

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 /* if */

/* Create the history record */
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_retrieved,
@today, @h_amount, (@w_name + "    " + @d_name))

/* COMMIT TRANSACTION PNM */

if (@commit_flg = 1)
            COMMIT TRANSACTION PNM
          else
      ROLLBACK TRANSACTION PNM

select/* Return to client */
@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_idsmallint,
@d_idtinyint,
@c_idint
as

DECLARE@o_idint,
@o_entry_ddatetime,
@o_carrier_idsmallint

BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
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
ANDo_d_id= @d_id
ANDo_c_id= @c_id
ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
set rowcount 0

/* Select order lines for the current order */
select/* Return multiple rows to client */
ol_supply_w_id,
ol_i_id,
ol_quantity,
ol_amount,
ol_delivery_d
FROMorder_line HOLDLOCK
WHEREol_o_id = @o_id
ANDol_d_id = @d_id
ANDol_w_id = @w_id

select/* Return single row to client */
@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
ANDc_d_id  = @d_id
ANDc_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_idsmallint,
@d_idtinyint,
@c_lastchar(16)
as

DECLARE@o_idint,
@o_entry_ddatetime,
@o_carrier_idsmallint

declare@nint,@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


/* Get the latest order made by the customer */
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
ANDo_d_id= @d_id
ANDo_c_id= @c_id
ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
set rowcount 0

/* Select order lines for the current order */
select/* Return multiple rows to client */
ol_supply_w_id,
ol_i_id,
ol_quantity,
ol_amount,
ol_delivery_d
FROMorder_line HOLDLOCK
WHEREol_o_id = @o_id
ANDol_d_id = @d_id
ANDol_w_id = @w_id

select/* Return single row to client */
@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
ANDc_d_id  = @d_id
ANDc_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,@ten_tinyinttinyint,
@one_tinyinttinyint,@one_smallintsmallint,
        @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

        /*
        ** The only purpose of the index hint in the above is to ensure
        ** that the clustered index is used.  As it turns out, our optimizer
        ** chooses the clustered index anyway -- with or without the hint.
        */
begin
    SELECT @one_tinyint = 1, @ten_tinyint = 10, @one_smallint = 1
    while (@d_id <= @ten_tinyint ) 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 + @one_smallint
                WHERE   c_id    = @o_c_id
                AND     c_d_id  = @d_id
                AND     c_w_id  = @w_id

                COMMIT TRANSACTION DEL

                select      /* Return to client */
                        @no_o_id
        end
SELECT @d_id = @d_id + @one_tinyint
    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 /* Return to client */
FROMdistrict,
order_line (index ol_clu prefetch 2 lru),
stock (index s_clu prefetch 2 lru)
WHEREd_w_id=@w_id
ANDd_id=@d_id
ANDol_w_id=@w_id
ANDol_d_id=@d_id
ANDol_o_idbetween (d_next_o_id - 20) and (d_next_o_id - 1)
ANDs_w_id=ol_w_id
ANDs_i_id=ol_i_id
AND     s_quantity < @threshold
go
EOF

if [ "$?" != "0" ]
then
echo ""
echo " **** WARNING: Could not connect to server to install procs! ****"
echo " ****      SQL SErver must have failed!!! ****"
echo " ****      TPC-C build & run will fail!!! ****"
echo ""
fi
```

SYB_tpcc.h

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#ifndef SYB_TPCC_H
#define SYB_TPCC_H

#pragma ident "@(#)SYB_tpcc.h1.495/09/12SMI"

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

/*
 * XXX: For error handlers
 */
int     err_handler();
intmsg_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

/*
** Structure for each line of an order
*/

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];
intincrement;
} ORDER_LINE;

/*
**  Define the TPC-C functions
*/

voidgen_new_order();voidnew_order_rpc();
voidgen_payment_byid();voidpayment_byid_rpc();
voidgen_payment_byname();voidpayment_byname_rpc();
voidgen_order_status_byid(); voidorder_status_byid_rpc();
voidgen_order_status_byname();voidorder_status_byname_rpc();
voidgen_delivery();voiddelivery_qu_add();
voidgen_stock_level();voidstock_level_rpc();
voiddelivery_qu_del();voiddelivery_rpc();
voiddelivery_qu_connect();
voiddisplay_xction();
voidsleep_before_retry ();
voiddisplay_xction();
voidpick_xact_type();

typedef struct Xction {
charname[30];
} XCTION;

extern XCTION func_array[XCTION_COUNT+1];

extern introllback_pct;
extern intlines_per_call;
extern charb_g[2];
extern DBFLT8total_amount;
extern DBTINYINTcommit_flag;
extern intxact_type, prev_xact_type;
extern intdeadlock;
extern intbad_items;
extern intmax_ware;
extern RETCODEcode;

/* set to global so that wont clash with tpcc_client.h */

extern DBSMALLINTglobal_w_id;
extern DBTINYINTglobal_d_id;
extern ORDER_LINEol[15];  /* XXX: should be 16, or 15 ??? */

/*
** Variables for the customer table
*/

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];

/*
** Variables for warehouse
*/

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;

/*
** Variables for district
*/

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;

/*
** Variables for item table
*/

extern inti_id;
extern DBFLT8i_price;
extern chari_name[25];

/*
** Variables for the stock table
*/

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

/*
** Variables for order table
*/

extern into_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;

/*
** Variables for order_line
*/

extern intol_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;

/*
** Variables for new_order tble
*/

extern intno_o_id;
extern DBTINYINTno_d_id;
extern DBSMALLINTno_w_id;

/*
** Variables for history table
*/

extern DBFLT8h_amount;
extern charh_date[20];

#endif SYB_TPCC_H
```

SYB_driver.h

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#ifndef SYB_DRIVER_H
#define SYB_DRIVER_H

#pragma ident "@(#)SYB_driver.h1.294/12/07SMI"

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

/* XXX: Basically we don't need this file in dbbench.  We only need the
        above 4 lines.   */

/* define argv's *//* XXX: We won't need these */
#defineDBNAME1       /* XXX */
#define FUNC_NAME2       /* XXX */
#defineNUSERS3       /* XXX */
#defineRAMP_UP4       /* XXX */
#defineSTDYSTATE5       /* XXX */
#defineRAMP_DOWN6       /* XXX */
#define MAX_WAREHOUSE7       /* XXX */
#defineROLLBACK_PCT8       /* XXX */
#defineDELTA9       /* XXX */

#defineINTERVAL25/* XXX: Total interval of buckets, in sec */
#defineUNIT.5/* XXX: Time period of each bucket */
#defineHIST_MAX50/* XXX: Num of histogram buckets = INTERVAL/UNIT */
#define BUCKET500/* XXX: Division factor for response time */
#defineMATCH0/* XXX: used as ='s with string ops */
#define MILLI1000/* XXX: Conversion from seconds to milliseconds */

typedef struct  CNTRL           /* XXX: We won't need these */
{
inttran_count;
intdeadlock_cnt;
intres_time;
doubletot_time;
intmin_res;
intmax_res;
 intnot_done;
double  tran_sqr;
inttran_2sec;
} CONTROL;

extern LOGINREC*login;
extern DBPROCESS*dbproc;

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

doubledrand();
voidsel1(); /* Function to run "select 1" test */
/* interr_handler();  XXX: moved to SYB_tpcc.h */
/* intmsg_handler();  XXX: moved to SYB_tpcc.h */
void     init_time();
unsigned long delay();

#endif SYB_DRIVER_H
```

SYB_error.c

```
/*
** Confidential property of Sybase, Inc.
** (c) Copyright Sybase, Inc. 1991
** All rights reserved
**
** Version 94.04.26
** Modified by Keng-Tai Ko [05/19/94]
*/
#include <stdio.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include "SYB_tpcc.h"

/* message numbers that we don't want to deal with */
#defineCONTEXT_SET5701
#defineLANGUAGE_SET5703
#defineCHARACTER_SET5704
#defineABORT_ERROR6104

/* Tuxedo include files */
#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 on any error */
exit(-100);
}



int
msg_handler(dbproc,msgno,msgstate,severity,msgtext,servername,procname,line)
DBPROCESS*dbproc;
intmsgno;
intmsgstate;
intseverity;
char*msgtext;
char            *servername;
char            *procname;
int             line;
{
/* changing database messages */
if (msgno == CONTEXT_SET ||
    msgno == LANGUAGE_SET ||
    msgno == CHARACTER_SET)
return(SUCCEED);

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

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

SYB_rpc.c

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)SYB_rpc.c1.1195/10/18SMI"

#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"

/* Tuxedo include files */
#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"

#if ACID
time_t curtime,
*timep = &curtime;
int returned_time;
char acid_date[25];

#endif

/* Date conversion routines */
DBDATETIME      syb_datetime;
DBDATETIME      syb_date;

/* Added to count invalid transactions for V 3.3 */
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;      /* Pointer to current item */

#ifdef ACID
    DBSMALLINT      com_flag;
    int well_sleep = 0;
#endif

    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 ACID
if ((ol[i].i_id==99) || (ol[i].i_id> 100000) || (ol[i].i_id==98))
well_sleep = 1;
#endif
    }

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

#ifdef ACID
    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date),acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    userlog("ACID NEWORDER Transaction begins: %s\n",acid_date);
#endif

    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')
    {
/*display_xction("Invalid item in");*/
/*bad_items++;*/

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("");
    }
    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)
{
/*
    userlog("NEWORDER failed\n");
*/
return TRUE;
    }

#ifdef ACID

    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date),acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
    if(well_sleep)
    userlog("ACID NEWORDER sleeping before commit/rollback: %s\n",acid_date);

    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc,1,SMALLBIND,0,&com_flag);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    if (com_flag == TRUE)
                userlog("ACID NEWORDER commit time: %s\n",acid_date);
    else
                userlog("ACID NEWORDER rollback time: %s\n",acid_date);

    if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;

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


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

#if ACID
    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date),acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
    userlog("ACID NEWORDER completed at %s Order id %d\n", acid_date,o_id);
    if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
#endif

/*
    userlog("finishing\n");
*/


    /* TPC-C V3.3 Error Checking for invalid input data */
```

```
    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()
{
#ifdef ACID
DBFLT8      pay_amount;
DBFLT8      pay_balance;
DBSMALLINT      pay_cust;
#endif
    if (dbsqlok(dbproc) != SUCCEED)
            return TRUE;
    if (dbresults(dbproc) != SUCCEED || deadlock)
    return TRUE;
    else
    {
#ifdef ACID

    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    userlog("ACID PAYMENT Transaction begins: %s\n",acid_date);
    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc,1,FLT8BIND,0,&pay_balance);
    dbbind(dbproc,  2, SMALLBIND,     0, &pay_cust);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
    if (h_amount == 100.00)
                userlog("ACID PAYMENT rollback : W_ID = %d D_ID = %d C_ID = %d AMOUNT
= %0.2f BALANCE = %0.2f\n",global_w_id,global_d_id,pay_cust,h_amount,pay_balance);
    else
                userlog("ACID PAYMENT commit : W_ID = %d D_ID = %d C_ID = %d AMOUNT =
%0.2f BALANCE = %0.2f\n",global_w_id,global_d_id,pay_cust,h_amount,pay_balance);

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


#endif


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;
#ifdef ACID

    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    userlog("ACID PAYMENT Transaction ends: %s\n",acid_date);

#endif

if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
sybdate2datetime(&syb_datetime, h_date);
sybdate2date(&syb_date, c_since);

/*
    if (*c_last == '\0')
    {
      strcpy(c_data, "   Execution Status :  Invalid Transaction......   ");
              tpreturn(TPFAIL, 0, (char *)paym_rqst->data, paym_rqst->len, 0);

            }
*/

    }

    /* TPC-C V3.3 Error Checking for invalid input data */
    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()
{
    intcount;

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

#ifdef ACID

    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    userlog("ACID ORDER STATUS Transaction begins: %s\n",acid_date);

#endif


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


        /* V 3.3 Error Checking
** checks whether current command returned any rows
*/
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)
{
    /*
    ** Print order_line information on RTE
    */
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;
#if ACID
if ((ol_i_id == 98) || (ol_i_id == 100))
    userlog("ACID ORDER STATUS Item %d Price %f\n",ol_i_id,ol_amount );
#endif
sybdate2date(&syb_date, ol_delivery_d);
                strcpy(ordstat->o_items[count].ol_delivery_d, ol_delivery_d);
                count++;
}
        ordstat->item_cnt = count;

/*
userlog("count= %d \n", count);
*/

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

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

        /* V 3.3 Error Checking
** checks whether current command returned any rows
*/
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;

#if ACID
    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
    userlog("ACID ORDER STATUS completed at %s Order id %d\n", acid_date,o_id);
    if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
#endif

    }

    return FALSE;
}

void
delivery_rpc()
{
/*
  Called by delivery processes.
*/
    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()
{
#ifdef ACID
DBFLT8tot_amount;
DBSMALLINT dl_c_id;
DBSMALLINT      com_flag;
#endif

    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++)
    {
#ifdef ACID

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

```
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    userlog("ACID DELIVERY Transaction begins: %s\n",acid_date);

#endif
    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;

#ifdef ACID

    dbbind(dbproc, 1, SMALLBIND, 0, &dl_c_id);
    dbbind(dbproc, 2, FLT8BIND, 0, &tot_amount);
    dbbind(dbproc, 3, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    userlog("ACID DELIVERY W_ID = %d D_ID = %d C_ID = %d AMOUNT =
%0.2f\n",global_w_id,global_d_id,dl_c_id,tot_amount);

    if(global_d_id == 1)
    userlog("ACID DELIVERY sleeping before commit/rollback: %s\n",acid_date);

    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    if(global_d_id == 1)
        if(o_carrier_id == 9)
                userlog("ACID DELIVERY rollback time: %s\n",acid_date);
        else
                userlog("ACID DELIVERY commit time: %s\n",acid_date);
    else userlog("ACID DELIVERY commit time: %s\n",acid_date);

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


#endif

    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 ACID
    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
    userlog("ACID DELIVERY completed at %s\n", acid_date);
#endif

    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;

#ifdef ACID

    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;

    userlog("ACID STOCKLEVEL Transaction begins: %s\n",acid_date);

#endif

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

    /* sort for distinct(s_i_id) */
    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;

#if ACID

    userlog("ACID STOCKLEVEL w_id=%d d_id=%d threshold=%d low_count=%d\n",
global_w_id, global_d_id, threshold, low_count);
    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(acid_date), acid_date);
    if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
    userlog("ACID STOCKLEVEL completed at %s Order id %d\n", acid_date,o_id);
    if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
#endif

    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:/* new_order */
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:/* payment_byid */
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:/* payment_byname */
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:/* order_status_byid */
userlog("cw=%d, cd=%d, c=%d\n", c_w_id, c_d_id, c_id);
break;

    case XACT_ORDS_NAME:/* order_status_byname */
userlog("cw=%d, cd=%d, l=%s\n", c_w_id, c_d_id, c_last);
break;

    case XACT_DEL:/* delivery_qu */
userlog("w=%d, carrier=%d\n", global_w_id, o_carrier_id);
break;

    case XACT_STOCK:/* stock level */
userlog("w=%d, d=%d, th=%d\n", global_w_id, global_d_id, threshold);
break;

    case XACT_BKEND:/* delivery */
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);
    }
}
```

SYB_rpc_var.c

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)SYB_rpc_var.c1.495/09/12SMI"

/*
**  This file contains the declaration of the shared variables in rpc.c.
**  Shared variables are used in passing arguments to TPC-C transactions.
*/

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

introllback_pct;
intlines_per_call = 15;
charb_g[2];
DBFLT8total_amount;
DBTINYINT commit_flag;
intxact_type, prev_xact_type = -9999;
intdeadlock;
intbad_items;
intmax_ware;
char*db_name = "tpcc";
RETCODEcode;

DBSMALLINTglobal_w_id;
DBTINYINTglobal_d_id;
ORDER_LINEol[15];/* XXX: should be 16, or 15 ?? */

/*
** Variables for the customer table
*/

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

/*
** Variables for warehouse
*/

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

/*
** Variables for district
*/

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;

/*
** Variables for item table
*/

inti_id;
DBFLT8i_price;
chari_name[25];

/*
** Variables for the stock table
*/

DBSMALLINTs_quantity;
DBSMALLINTthreshold;
DBINTlow_count;
chars_dist[25];

/*
** Variables for order table
*/

into_id;
DBTINYINTo_d_id;
DBSMALLINTo_w_id;
DBSMALLINTo_c_id;
charo_entry_d[31];
DBSMALLINTo_carrier_id;
DBSMALLINTo_ol_cnt, o_ol_now, o_ol_done;
DBTINYINTo_all_local;

/*
** Variables for order_line
*/

intol_o_id;
DBTINYINTol_d_id;
DBSMALLINTol_w_id;
DBSMALLINTol_number;
DBINTol_i_id;
DBSMALLINTol_supply_w_id;
charol_delivery_d[31];
DBSMALLINTol_quantity;
DBFLT8ol_amount;

/*
** Variables for new_order tble
*/

intno_o_id;
DBTINYINTno_d_id;
DBSMALLINTno_w_id;

/*
** Variables for history table
*/

DBFLT8h_amount;
charh_date[20];
```

tpcc_tux_forms.h

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#ifndef TPCC_TUXFORMS_H
#define TPCC_TUXFORMS_H

#pragma ident "@(#)tpcc_tux_forms.h1.394/12/07SMI"

/*  This file contains the definition of the data structures used
 *  by the tpcc_srv_xxx.c files to communicate with the Tuxforms.
 *
 *  It should be placed AFTER the following include files:
 * #include "atmi.h"
 * #include "userlog.h"
 *
 */

/* For NEWO:     */

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;
};


/* For BKEND:*/

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

/* Structure used to queue delivery transaction */
struct req_struct {
int w_id;
int o_carrier_id;
time_t qtime;           /* Time transaction was queued */
};


/* For ORDS:*/

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 {  /* This structure is copied from ORDS_VIEW in TPCC.h */
        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];
};


/* For PAYM:    */

/* List of fields in payment */
/* This structure should be EXACTLY identical to the one
 * declared in client.h
 */
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];
};


/* For STOCK:*/

/* List of fields in stock */
/* This structure should be EXACTLY identical to the one declared in client.h *//*
List of fields in stock-level */
struct stock_inf {
int w_id;
int d_id;
int threshold;
int low_stock;
};

#endif TPCC_TUXFORMS_H
```

tpcc_tux_forms_var.c

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcc_tux_forms_var.c1.394/12/07SMI"

/* This file declares the share variables between tpcc_srv_xxx.c and
 * SYB_rpc.c files.
 *
 * The idea is that this file will only be included in SYB_rpc.c placed
 * after the #include "tpcc_tux_forms.h" line.  The tpcc_srv_xxx.c files
 * will declare those that they need as extern.
 *
 */


/* For NEWO:     */

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;


/* For BKEND:   */

struct req_struct *delp;      /* Transaction message */
char              outbuf[1024];  /* Buffer for results file */
int               tx_count = 0;  /* Transaction counter */
FILE              *delfile;
TPSVCINFO         *del_rqst;


/* For ORDS:    */

struct ord_inf ordsp,
                *ordstat = &ordsp;
FBFR *ordsbuf;  /* FML buffer for output carray */
int ordslen;    /* Size of FML buffer */
TPSVCINFO *ords_rqst;


/* For PAYM:*/

struct pay_inf *payment;        /* Input structure to payment_tx */
struct pay_inf paymsp;          /* Payment structure */
TPSVCINFO *paym_rqst;


/* For STOCK:   */

struct stock_inf *stocklevel;   /* Input to stocklevel transaction */
struct stock_inf stocksp;
TPSVCINFO *stock_rqst;
```

tpcc_srv_newordpay.c

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcc_srv_newo.c1.394/12/07SMI"

/*
 * File: newo.c
 * Neworder transaction code for Sybase using Tuxedo
 * Author : Sheahan D
 * Date   : 15/06/94
 *
 */

#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>

/* Sybase header files */
#include <sybfront.h>
#include <sybdb.h>
#include "SYB_tpcc.h"
#include "SYB_driver.h"
/*  #include "SYB_rpc_var.c"  XXX: Don't need this line.  */

/* Tuxedo includes */
#include "atmi.h"
#include "userlog.h"

#include "tpcc_tux_forms.h"
/* Lists of items on an order */
/* These structures should match the struct definitions for item_struct and no_struct
 * defined in client.h exactly.
 * Any change to those, should be reflected here
 */


struct newo_inf *neworder;/* Neworder field structure */
struct ord_inf *ordstat;/* Input structure to ordstat_tx */
char blank_mesg[25] = "                          ";


/* List of fields in payment */
/* This structure should be EXACTLY identical to the one declared in client.h */
#ifdef NOT_REQ
struct pay_inf {
intw_id;
intd_id;
intc_id;
intc_w_id;
intc_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;/* Input structure to payment_tx */


DBPROCESS       *dbproc;
LOGINREC        *login;


#if ACID
#include <sys/types.h>
#include <time.h>
time_t curtime, *timep = &curtime;
#endif

/*
 * Initialize the neworder transaction
 */
int
init_all_tx()
{
        /* Install the error and message handler */
userlog("before dberrhandle \n");
        dberrhandle(err_handler);
        dbmsghandle(msg_handler);

        /* Initialize global variable for error handling */
        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"));

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

        /* Use the the right database */
userlog("before dbuse \n");
        if ( dbuse(dbproc, (char *)DATABASE) != SUCCEED)
        {
                initerr("Fatal dbuse: Could not use DATABASE\n");
                return(-1);
        }

        /* Done with initialization */
userlog("leaving tpsvrinit \n");
        return(0);

}


/*
 * This function executes the neworder transaction
 */
neworder_tx(rqst)
TPSVCINFO *rqst;
{

        int     i;
int rollback = 0;
int linecnt;
int ret;
struct items_inf *cur_ip;      /* Pointer to current item */


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

/*
dtcurrent(&ord_date);
dttofmtasc(&ord_date, neworder->o_entry_d,
sizeof(neworder->o_entry_d), "%d-%m-%Y %H:%M:%S");
*/


/*        datetime(&neworder->o_entry_d); */
strncpy(neworder->status, blank_mesg, 24);
#if ACID
time(timep);
userlog("ACID NEWORDER Transaction begun at %s\n", ctime(timep));
#endif
/* read warehouse, customer */
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;        /* Assume all local, then check.
                                   Ideally, this flag can be passed from
                                   the FORMs package */
/*
userlog("after o_all_local ... \n");
*/

        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;        /* non-local order */
        }

/*
userlog("before calling new_order_rpc \n");
userlog("w_id=%d, d_id=%d, c_id=%d, o_ol_cnt=%d \n", global_w_id, global_d_id, c_id,
o_ol_cnt);
*/

        new_order_rpc();


        /* pick up total amount */
        neworder->total = total_amount;




#if ACID
time(timep);
userlog("ACID NEWORDER completed at %s\n", ctime(timep));
#endif
/*
userlog("after calling new_order_rpc \n");
userlog("w_id=%d, d_id=%d, c_id=%d, o_ol_cnt=%d \n", neworder->w_id, neworder->d_id,
neworder->c_id, neworder->o_ol_cnt);
*/
tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct newo_inf), 0);
}

/* Start of Tuxedo code */
int
tpsvrinit(argc, argv)
char **argv;
{
return(init_all_tx());/* Prepare transaction */
}

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 ACID
time(timep);
userlog("ACID ORDSTAT Transaction begun at %s\n", ctime(timep));
#endif
        if (ordstat->c_id == 0) {       /* Customer selected by name */
                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);
        }


#if ACID
time(timep);
userlog("ACID ORDSTAT Transaction completed at %s\n", ctime(timep));
#endif
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);

#if ACID
time(timep);
userlog("ACID PAYMENT Transaction Begun at %s\n", ctime(timep));
#endif

        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) {              /* Customer selected by name */
                byid = FALSE;
                xact_type = XACT_PAYM_NAME;
        }
        else {
                byid = TRUE;
                xact_type = XACT_PAYM_ID;
        }

        if (byid) {               /* Customer selected by id */
                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);
        }

#ifdef ACID
time(timep);
userlog("ACID PAYMENT Transaction completed at %s\n", ctime(timep));
#endif
tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct pay_inf), 0);
}

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

tpcc_srv_stockdel.c

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */


/*
#pragma ident "@(#)tpcc_srv_del.c1.394/12/07SMI"
 * File: tpcc_srv_stockdel.ec
 * Delivery and Stock transaction code for Sybase Tuxedo
 * This program is different from the other servers, in that it
 * records transaction info in a results file.
 * created by sudhas
 *
 */

#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"

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

/* Sybase header files */
#include <sybfront.h>
#include <sybdb.h>
#include "SYB_tpcc.h"
#include "SYB_driver.h"
/*  #include "SYB_rpc_var.c"  XXX: Don't need this line.  */


static struct req_struct *delp;/* Transaction message */
extern char outbuf[];
extern int tx_count;/* Transaction counter */
extern FILE *delfile;


/* List of fields in stock */
/* This structure should be EXACTLY identical to the one declared in client.h */
/* List of fields in stock-level */
/* struct stock_inf {
int     w_id;
int d_id;
int threshold;
int low_stock;
}; */

struct stock_inf *stocklevel;   /* Input to stocklevel transaction */


DBPROCESS       *dbproc;
LOGINREC        *login;



#if ACID
#include <sys/types.h>
#include <time.h>
time_t curtime, *timep = &curtime;
#endif

cleanup()
{
        fclose(delfile);
}


int
init_stockdel_tx()
{
        /* Prepare delivery transaction */

        /* Install the error and message handler */
    userlog("before dberrhandle \n");
        dberrhandle(err_handler);
        dbmsghandle(msg_handler);

        /* initialize global variable for deadlock and error handling */
        deadlock = 0;


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

        /*
         *      always use large packet ...
                cp = getenv("CHARSET");
            userlog("result of getenv CHARSET. result= %s \n", cp);
                cp = getenv("PACKET");
            userlog("result of getenv PACKET. result= %s \n", cp);
                if (strcmp(cp,"LARGE") == 0) {
         */

    userlog("before DBSETLPACKET \n");

        DBSETLPACKET(login, 4096);

        /*
                }
         */

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

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

        /* Use the the right database */
    userlog("before dbuse \n");
        if ( dbuse(dbproc, (char *)DATABASE) != SUCCEED)
        {
            initerr("Fatal dbuse: Could not use DATABASE\n");
            return(-1);
        }

        /* Done with initialization */
    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);
#if ACID
time(timep);
userlog("ACID DELIVERY Transaction begun at %s\n", ctime(timep));
#endif

    delivery_rpc();     /* XXX: use Sybase's SYB_rpc.c version  */


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);
}


/* If errors occur during initialization, exit */
initerr(str)
char *str;
{
        userlog("init_stockdel_tx ERROR %\n", str);
}


/* Tuxedo */
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 = atoi(argv[optind]); */  /* Needs argument which is the proc_no */

proc_no = (int)getpid();
/* Get hostname of our machine  and create results file */
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());/* Prepare transaction */
}

void
tpsvrdone()
{
        cleanup();      /* Close results file */
        dbexit();
}

DEL(rqst)
TPSVCINFO *rqst;
{
        xact_type = XACT_BKEND;  /* ??? is is OK to set it here ???? */
delivery_tx(rqst);
}



/*
 * Function: do stocklevel transaction
 * Input is the stocklevel structure. Output is low_stock field
 */
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;  /* ??? is is OK to set it here ???? */
        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.

bld_system

```
#!/bin/ksh  -x
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)bld_system1.196/01/09SMI"
#
DB_SCALE=16500
CONFIG_FILE=$SQL_RELEASE/A4.cfg
DEVICE_FILE=/misc/veritas/tmp/devices
export DEVICE_FILE

#Cleanup added by blarson 12/8/99
# May as well dump old archived admin-logfile since we will run new buildmaster
rm $HOME/err.${USER_SUF}.log

echo `date` "Started bld_system"

`$HOME/vendors/sybase/scripts/devcreate.sh buildmaster \
$SQL_RELEASE/bin/buildmaster \
< $DEVICE_FILE` > $$_bm.log

# Boot server, run installmaster, reconfigure server, and shutdown
$HOME/vendors/sybase/scripts/run_s_server - -c$CONFIG_FILE

sleep 120

# 11.9.3 way:
isql -Usa -P < $SYBASE/$SYBASE_ASE/scripts/installmaster > $$_im.log


# Build devices, database, and segments
echo `date` "Creating devices, databases and segments"
#$HOME/vendors/sybase/scripts/devcreate.sh sql System11 \
#< $DEVICE_FILE | \
#isql -e -Usa -P > $$_create_db.log
#echo `date` " Finished building database"

$HOME/vendors/sybase/scripts/devcreate.sh sql System11 \
< $DEVICE_FILE > cr_db.sql

cp cr_db.sql disk_init.sql

ed disk_init.sql <<EOT
/create/,\$d
w
q
EOT

ed cr_db.sql <<EOT
/create/
1,.-1d
w
q
EOT

isql -e -Usa -P -i disk_init.sql > $$_disk_init.log
$HOME/vendors/sybase/scripts/shutdown_server.sh >> $$_trunc_log.log 2>&1
sleep 49

$HOME/vendors/sybase/scripts/run_s_server - -c${CONFIG_FILE}
sleep 300
# create tpcc database
# drop system and default from new devices
isql -e -Usa -P -i cr_db.sql > $$_create_db.log

# Create tables, some indexes, and administrative procs.
$HOME/vendors/sybase/TPCSO/scripts/tpcc_tables.sh > $$_tables_procs.log 2>&1
$HOME/vendors/sybase/TPCSO/scripts/tpcc_admin.sh >> $$_tables_procs.log 2>&1
$HOME/vendors/sybase/scripts/generic_procs.sh >> $$_tables_procs.log 2>&1

# Truncate log, checkpoint, and shutdown
$HOME/vendors/sybase/scripts/dumptran_server.sh master > $$_trunc_log.log 2>&1
$HOME/vendors/sybase/scripts/dumptran_server.sh tpcc >> $$_trunc_log.log 2>&1
$HOME/vendors/sybase/scripts/shutdown_server.sh >> $$_trunc_log.log 2>&1
sleep 180

# Start server with no logging and 4 engines for load.
# Load the data; shutdown again.
$HOME/vendors/sybase/scripts/run_server - -T699 -c${CONFIG_FILE}
sleep 360

echo `date` " Started loading data"
$HOME/vendors/sybase/TPCSO/scripts/tpcc_load.sh 1 $DB_SCALE > $$_load.log 2>&1
echo `date` " Finished loading data"

$HOME/vendors/sybase/scripts/shutdown_server.sh

sleep 180
date

# Reboot, create indexes, config trips, load stored procs,
# and create named caches. Shutdown.

$HOME/vendors/sybase/scripts/run_server - -c${CONFIG_FILE}
sleep 360
echo `date` -- Started building indexes
$HOME/vendors/sybase/TPCSO/scripts/tpcc_indexes.sh > $$_indexes.log 2>&1
echo `date` -- Finished building indexes

$HOME/vendors/sybase/TPCSO/scripts/tpcc_stats.sh >> $$_indexes.log 2>&1
$HOME/vendors/sybase/TPCSO/scripts/tpcc_proc_tax.sh >> $$_indexes.log 2>&1

$HOME/vendors/sybase/scripts/shutdown_server.sh

echo `date` "Finished bld_system"
```

bulk_sybase.c

```
/*****************************************************************
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)bulk_sybase.c1.396/03/01SMI"

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

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];
intcount[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 8192 */
DBSETLPACKET(login, 8192);

/* 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);
}
```

customer.sh

```
:
PROG=`basename $0 .sh`
DATE=`date +%y%m%d:%H%M`
LOG=$PROG.$DATE.log
(
date
TABLE=customer
SEGMENT=Scustomer
INDEX=c_clu
FIELDS="c_w_id, c_id, c_d_id"
isql -w144 -Usa -P -e -n <<EOF
use tpcc
go
select getdate()
go
sp_spaceused $TABLE
go
drop index customer.c_non1
go
sp_spaceused $TABLE
go
sp_spaceused $TABLE
go
drop index customer.c_clu
go
sp_spaceused $TABLE
go
select getdate()
go
set statistics io on
go
create unique clustered index $INDEX
on $TABLE($FIELDS)
with sorted_data
on $SEGMENT
go
set statistics io off
go
select getdate()
go
sp_spaceused $TABLE
go
select getdate()
go
set statistics io on
go
create unique nonclustered index c_non1
on customer(c_w_id, c_d_id, c_last, c_first, c_id)
       on Scidx
go
set statistics io off
go
select getdate()
go
sp_spaceused $TABLE
go
select getdate()
go
EOF
) >> $LOG 2>&1
exit 0
```

des_tune.sh

```
:
echo "-- $0 --"
echo "............."
echo "SYBASE=$SYBASE"
echo "DSLISTEN=$DSLISTEN DSQUERY=$DSQUERY"
echo "............."
isql -w144 -n -e -Usa -P << EOF
dbcc tune(des_bind, 4, "orders")
go
dbcc tune(des_bind, 4, "customer")
go
dbcc tune(des_bind, 4, "district")
go
dbcc tune(des_bind, 4, "history")
go
dbcc tune(des_bind, 4, "item")
go
dbcc tune(des_bind, 4, "new_order")
go
dbcc tune(des_bind, 4, "stock")
go
dbcc tune(des_bind, 4, "order_line")
go
dbcc tune(des_bind, 4, "warehouse")
go
dbcc tune(des_bind, 4, "delivery")
go
dbcc tune(des_bind, 4, "neworder_local")
go
dbcc tune(des_bind, 4, "neworder_remote")
go
dbcc tune(des_bind, 4, "order_status_byid")
go
dbcc tune(des_bind, 4, "order_status_byname")
go
dbcc tune(des_bind, 4, "payment_byid")
go
dbcc tune(des_bind, 4, "payment_byname")
go
dbcc tune(des_bind, 4, "stock_level")
go
select getdate()
go
EOF
```

devcreate.sh

```
#!/bin/ksh
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)devcreate.sh1.196/01/09SMI"
#
# @(#) devcreate.sh 1.1 6/7/95
#
#scripts/devcreate.sh
#
#Read a device file from stdin in the format given in format/devices
#and ouput on stdout the SQL statements to create the devices,
#databases and segments defined by the input device file.
#
#The SQL is output in the following order
#
#1) Disk inits and disk mirrors
#2) Create databases
#3) sp_addsegments and sp_extendsegments
#
#
#  Sreekanth Setty 2/18/97
#    - with bug fixes
#    - version works with System 11 and above
#
#

if [ "$1" = "buildmaster" ]
then
bm=y
bmbinary=$2
elif [ "$1" = "sql" ]
then
bm=n
release=$2
else
```

```
echo "Usage : $0 [builmaster buildmaster_binary |sql] < device_file" >&2
echo "buildmaster - generate buildmaster command" >&2
echo "sql [release] - generate SQL commands" >&2
exit 1
fi

in_device=n
in_db=n

logical_name=
physical_name=
device_size=

vdevno=0

sql_file=/tmp/dvsql$$
db_file=/tmp/dvdb$$
db_s_file=/tmp/dvdb_s$$
seg_file=/tmp/dvseg$$
tmp_seg_file=/tmp/tmpdvseg$$
export sql_file db_file seg_file tmp_seg_file

grep -v '^#' | tr -s '\011 ' '\012\012' | while read token garbage
do
case $token in
DEVICE)# A new device
# clear the fields for the next device
logical_name=
physical_name=
device_size=
vstart_offset=
mirror=

in_device=y
in_db=n
;;

db=*)  # database name
if [ "$in_db" = "y" ]
then
# Store info about db
echo
fi

# Start the new database
db_name=`echo $token | sed 's/db=//'`
db_log=
db_size=0

in_db=y
;;


log)# This disk is the log disk for the current db
if [ "$in_db" = "y" ]
then
db_log="log on"
fi
;;

vstart=*)
vstart_offset=`echo $token | sed 's/vstart=//'`
;;

mirror=*)
# store info about log-mirror
mirror=`echo $token | sed 's/mirror=//'`
;;

size=*) # the size of the current db on this disk
if [ "$in_db" = "y" ]
then
# Store info about size
db_size=`echo $token | sed 's/size=//'`

fi
;;

segment=*)
if [ "$in_db" = "y" ]
then
segment=`echo $token | sed 's/segment=//'`
echo "$db_name $segment $logical_name" >> $seg_file
fi
;;

DEVICE_END)# Complete the device

# Save any database fragment information
if [ "$in_db" = "y" ]
then
echo "$db_name $logical_name $db_size $db_log" >> $db_file
fi

if [ "$bm" = "y" -a "$in_device" = "y" -a "$logical_name" = "master" ]
then
# Convert Mb to 2k pages.
page_size=`expr $device_size \* 512`

echo "$bmbinary -d$physical_name -s$page_size"
fi

# The disk init SQL, but not for the master device
#
if [ "$bm" = "n" -a "$in_device" = "y" -a "$logical_name" != "master" ]
then
# Convert Mb to 2k pages.
page_size=`expr $device_size \* 512`

# echo SQL to create the device
echo "disk init"
echo "  name = '$logical_name',"
echo "  physname = '$physical_name',"
echo "  vdevno = $vdevno,"
echo "  size = $page_size"
if [ "$vstart_offset" != "" ]
then
echo "  , vstart = $vstart_offset"
fi
echo go
fi


# The disk mirror SQL (including master)
if [ "$mirror" = "" -o "$bm" = "y" ]
then
false;
else
# Echo SQL to create the disk mirror
echo "disk mirror"
echo "  name = '$logical_name',"
echo "  mirror = '$mirror',"
echo "  writes = noserial"
echo go
fi

;;

*)# could be one of several
if [ "$in_device" = "y" ]
then
if [ "$logical_name" = "" ]
then
logical_name=$token
if [ $logical_name != "master" ]
then
vdevno=`expr $vdevno + 1`
fi
elif [ "$physical_name" = "" ]
then
physical_name=$token
elif [ "$device_size" = "" ]
then
device_size=$token
else
echo
fi
else
echo
fi
;;
esac
done

# If we are in buildmaster mode we can just stop here.
if [ "$bm" = "y" ]
then
rm $db_file $seg_file
exit 0
fi

#
# Now we have generated the disk init commands, create the
# create database commands.
#
# The file $db_file will have been created with the following format
#
# dbname device size [log on]

#sort $db_file > $db_s_file
cat $db_file > $db_s_file
rm $db_file

# Add a dummy line end to the database file
echo "__$$" >> $db_s_file

cat $db_s_file > /export/home/dbbench/vendors/sybase/tmp_file
cat $seg_file > /export/home/dbbench/vendors/sybase/tmp_seg_file

current_db=
in_db=n
logdbinfo=
export in_db current_db logdbinfo
#
#
echo "set statistics time on"
echo "go"
echo "set statistics io on"
echo "go"

#
#
LINE=1
WORD=1
cat $db_s_file | while read dbname device size log
do
if [ "$dbname" = "$current_db" ]
then
if [ -z "$log" ]
then
WORD=`expr $WORD + 1`
if [ -z "${dbinfo[$LINE]}" ]
then
dbinfo[$LINE]="on $device = $size"
else
if [ $WORD -eq 4 ]; then
    LINE=`expr $LINE + 1`
    WORD=0
fi
dbinfo[$LINE]="${dbinfo[$LINE]}, $device = $size"
WORD=`expr $WORD + 1`
fi
else
#added by sree
                        if [ -z "$logdbinfo" ]
then
        logdbinfo="$log $device = $size"
else
```

```
logdbinfo="$logdbinfo, $device = $size"
fi
fi
# Test here to see if we spot a new dbname...
elif  [ "$in_db" = "y" ]
then
# Here when we enter a second db, we make another create db...
echo "create database $current_db"
LINE_CNT=1
while [ $LINE_CNT -lt $LINE ]; do
    echo ${dbinfo[$LINE_CNT]}
    LINE_CNT=`expr $LINE_CNT + 1`
done
if [ -n "$logdbinfo" ]
then
echo $logdbinfo
fi
echo go
logdbinfo=

current_db=$dbname
dbinfo="on $device = $size"
in_db=y
else
# Here when we hit the first new dbname
current_db=$dbname
if [ -z "$log" ]
then
LINE_CNT=1
dbinfo[1]="on $device = $size"
else
logdbinfo="$log $device = $size"
fi
in_db=y
fi
done
#rm $db_s_file

#
# Now we have the create database commands, create the segment commands
#
# The file $seg_file will have been created with the following format
#
# dbname device segment
#

current_db=
current_seg=
seg_db=
export current_seg current_db seg_db

sort $seg_file | while read dbname segment device garbage
do
if [ "$dbname" = "$current_db" ]
then
false
else
echo "use $dbname"
echo go
# In System 10 segment procs now takes db as 2nd arg
#if [ "$release" = "System10" ]    changed by sree
#then
seg_db="$dbname ,"
#fi
fi

if [ "$segment" = "system" -o "$segment" = "default" ]
then
false # do nothing
elif [ "$segment" = "$current_seg" ]
then
echo "sp_extendsegment $segment , $seg_db  $device"
echo go
else
echo "sp_addsegment $segment , $seg_db $device"
echo go
fi
current_seg=$segment
current_db=$dbname
done

# now sort the segment file in database, device order
# to enable us to drop the unwanted system and default segments

in_device=no
export in_device

# added by sree
sort +0 -1 +2 -3 $seg_file > $tmp_seg_file
# Add a dummy line end to the database file
echo "dummy line $$" >> $tmp_seg_file

#sort +0 -1 +2 -3  $seg_file > /export/home/dbbench/vendors/sybase/tmp_seg_file1
cat $tmp_seg_file | while read dbname segment device garbage
do
if [ "$device" = "$current_dev" ]
then
false
else
if [ "$in_device" = "yes" ]
then
if [ "$drop_segs" = "yes" ]
then
echo "sp_dropsegment 'default', $seg_db $current_dev"
echo go
echo "sp_dropsegment 'system', $seg_db $current_dev"
echo go
fi
fi

in_device=yes
drop_segs=yes
fi
# 'and' condition added by sree

if [ "$dbname" = "$current_db"  -o  "$dbname" = "dummy" ]
then
false
else
echo "use $dbname"
echo go
# In System 10 segment procs now takes db as 2nd arg
#if [ "$release" = "System10" ]
#then
seg_db="$dbname ,"
#fi
fi

if [ "$segment" = "system" -o "$segment" = "default" ]
then
drop_segs=no
fi

current_dev=$device
current_db=$dbname
done

rm $seg_file $tmp_seg_file

echo"use master"
echogo
echo"checkpoint"
echogo
```

devices

```
DEVICE master /sybase_dev/Amaster1 250
    db=tpcc size=250
  segment=system
  segment=default
DEVICE_END

DEVICE customer1 /sybase_dev/Acust1 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer2 /sybase_dev/Acust2 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer3 /sybase_dev/Acust3 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer4 /sybase_dev/Acust4 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer5 /sybase_dev/Acust5 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer6 /sybase_dev/Acust6 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer7 /sybase_dev/Acust7 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer8 /sybase_dev/Acust8 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer9 /sybase_dev/Acust9 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer10 /sybase_dev/Acust10 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer11 /sybase_dev/Acust11 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer12 /sybase_dev/Acust12 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END
```

```
DEVICE customer13 /sybase_dev/Acust13 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer14 /sybase_dev/Acust14 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer15 /sybase_dev/Acust15 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer16 /sybase_dev/Acust16 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer17 /sybase_dev/Acust17 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer18 /sybase_dev/Acust18 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer19 /sybase_dev/Acust19 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer20 /sybase_dev/Acust20 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer21 /sybase_dev/Acust21 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer22 /sybase_dev/Acust22 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer23 /sybase_dev/Acust23 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer24 /sybase_dev/Acust24 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer25 /sybase_dev/Acust25 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer26 /sybase_dev/Acust26 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer27 /sybase_dev/Acust27 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer28 /sybase_dev/Acust28 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer29 /sybase_dev/Acust29 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer30 /sybase_dev/Acust30 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer31 /sybase_dev/Acust31 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer32 /sybase_dev/Acust32 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer33 /sybase_dev/Acust33 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer34 /sybase_dev/Acust34 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer35 /sybase_dev/Acust35 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer36 /sybase_dev/Acust36 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer37 /sybase_dev/Acust37 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer38 /sybase_dev/Acust38 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer39 /sybase_dev/Acust39 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer40 /sybase_dev/Acust40 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer41 /sybase_dev/Acust41 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer42 /sybase_dev/Acust42 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer43 /sybase_dev/Acust43 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer44 /sybase_dev/Acust44 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer45 /sybase_dev/Acust45 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer46 /sybase_dev/Acust46 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer47 /sybase_dev/Acust47 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer48 /sybase_dev/Acust48 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer49 /sybase_dev/Acust49 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer50 /sybase_dev/Acust50 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END
```

```
DEVICE customer51 /sybase_dev/Acust51 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer52 /sybase_dev/Acust52 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer53 /sybase_dev/Acust53 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer54 /sybase_dev/Acust54 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer55 /sybase_dev/Acust55 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer56 /sybase_dev/Acust56 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer57 /sybase_dev/Acust57 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer58 /sybase_dev/Acust58 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer59 /sybase_dev/Acust59 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer60 /sybase_dev/Acust60 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer61 /sybase_dev/Acust61 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer62 /sybase_dev/Acust62 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer63 /sybase_dev/Acust63 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer64 /sybase_dev/Acust64 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer65 /sybase_dev/Acust65 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer66 /sybase_dev/Acust66 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer67 /sybase_dev/Acust67 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer68 /sybase_dev/Acust68 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer69 /sybase_dev/Acust69 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer70 /sybase_dev/Acust70 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer71 /sybase_dev/Acust71 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE customer72 /sybase_dev/Acust72 5300
    db=tpcc size=5300
  segment=Scustomer
  segment=Scidx
DEVICE_END

DEVICE stock1 /sybase_dev/Astock1 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock2 /sybase_dev/Astock2 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock3 /sybase_dev/Astock3 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock4 /sybase_dev/Astock4 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock5 /sybase_dev/Astock5 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock6 /sybase_dev/Astock6 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock7 /sybase_dev/Astock7 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock8 /sybase_dev/Astock8 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock9 /sybase_dev/Astock9 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock10 /sybase_dev/Astock10 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock11 /sybase_dev/Astock11 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock12 /sybase_dev/Astock12 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock13 /sybase_dev/Astock13 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock14 /sybase_dev/Astock14 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock15 /sybase_dev/Astock15 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock16 /sybase_dev/Astock16 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock17 /sybase_dev/Astock17 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock18 /sybase_dev/Astock18 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock19 /sybase_dev/Astock19 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END
```

```
DEVICE stock20 /sybase_dev/Astock20 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock21 /sybase_dev/Astock21 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock22 /sybase_dev/Astock22 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock23 /sybase_dev/Astock23 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock24 /sybase_dev/Astock24 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock25 /sybase_dev/Astock25 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock26 /sybase_dev/Astock26 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock27 /sybase_dev/Astock27 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock28 /sybase_dev/Astock28 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock29 /sybase_dev/Astock29 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock30 /sybase_dev/Astock30 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock31 /sybase_dev/Astock31 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock32 /sybase_dev/Astock32 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock33 /sybase_dev/Astock33 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock34 /sybase_dev/Astock34 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock35 /sybase_dev/Astock35 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock36 /sybase_dev/Astock36 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock37 /sybase_dev/Astock37 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock38 /sybase_dev/Astock38 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock39 /sybase_dev/Astock39 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock40 /sybase_dev/Astock40 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock41 /sybase_dev/Astock41 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock42 /sybase_dev/Astock42 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock43 /sybase_dev/Astock43 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock44 /sybase_dev/Astock44 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock45 /sybase_dev/Astock45 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock46 /sybase_dev/Astock46 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock47 /sybase_dev/Astock47 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock48 /sybase_dev/Astock48 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock49 /sybase_dev/Astock49 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock50 /sybase_dev/Astock50 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock51 /sybase_dev/Astock51 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock52 /sybase_dev/Astock52 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock53 /sybase_dev/Astock53 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock54 /sybase_dev/Astock54 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock55 /sybase_dev/Astock55 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock56 /sybase_dev/Astock56 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock57 /sybase_dev/Astock57 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock58 /sybase_dev/Astock58 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock59 /sybase_dev/Astock59 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock60 /sybase_dev/Astock60 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock61 /sybase_dev/Astock61 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock62 /sybase_dev/Astock62 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock63 /sybase_dev/Astock63 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock64 /sybase_dev/Astock64 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock65 /sybase_dev/Astock65 7150
    db=tpcc size=7150
  segment=Sstock
```

```
DEVICE_END

DEVICE stock66 /sybase_dev/Astock66 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock67 /sybase_dev/Astock67 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock68 /sybase_dev/Astock68 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock69 /sybase_dev/Astock69 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock70 /sybase_dev/Astock70 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock71 /sybase_dev/Astock71 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock72 /sybase_dev/Astock72 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock73 /sybase_dev/Astock73 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock74 /sybase_dev/Astock74 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock75 /sybase_dev/Astock75 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock76 /sybase_dev/Astock76 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock77 /sybase_dev/Astock77 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock78 /sybase_dev/Astock78 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock79 /sybase_dev/Astock79 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock80 /sybase_dev/Astock80 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock81 /sybase_dev/Astock81 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock82 /sybase_dev/Astock82 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock83 /sybase_dev/Astock83 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock84 /sybase_dev/Astock84 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock85 /sybase_dev/Astock85 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock86 /sybase_dev/Astock86 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock87 /sybase_dev/Astock87 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock88 /sybase_dev/Astock88 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock89 /sybase_dev/Astock89 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock90 /sybase_dev/Astock90 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock91 /sybase_dev/Astock91 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock92 /sybase_dev/Astock92 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock93 /sybase_dev/Astock93 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock94 /sybase_dev/Astock94 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock95 /sybase_dev/Astock95 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock96 /sybase_dev/Astock96 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock97 /sybase_dev/Astock97 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock98 /sybase_dev/Astock98 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock99 /sybase_dev/Astock99 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE stock100 /sybase_dev/Astock100 7150
    db=tpcc size=7150
  segment=Sstock
DEVICE_END

DEVICE order_line1 /sybase_dev/Aoline1 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line2 /sybase_dev/Aoline2 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line3 /sybase_dev/Aoline3 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line4 /sybase_dev/Aoline4 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line5 /sybase_dev/Aoline5 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line6 /sybase_dev/Aoline6 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line7 /sybase_dev/Aoline7 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line8 /sybase_dev/Aoline8 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line9 /sybase_dev/Aoline9 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line10 /sybase_dev/Aoline10 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line11 /sybase_dev/Aoline11 9150
```

```
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line12 /sybase_dev/Aoline12 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line13 /sybase_dev/Aoline13 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line14 /sybase_dev/Aoline14 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line15 /sybase_dev/Aoline15 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line16 /sybase_dev/Aoline16 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line17 /sybase_dev/Aoline17 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line18 /sybase_dev/Aoline18 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line19 /sybase_dev/Aoline19 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line20 /sybase_dev/Aoline20 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line21 /sybase_dev/Aoline21 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line22 /sybase_dev/Aoline22 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line23 /sybase_dev/Aoline23 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line24 /sybase_dev/Aoline24 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line25 /sybase_dev/Aoline25 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line26 /sybase_dev/Aoline26 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line27 /sybase_dev/Aoline27 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line28 /sybase_dev/Aoline28 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line29 /sybase_dev/Aoline29 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line30 /sybase_dev/Aoline30 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line31 /sybase_dev/Aoline31 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line32 /sybase_dev/Aoline32 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line33 /sybase_dev/Aoline33 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line34 /sybase_dev/Aoline34 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line35 /sybase_dev/Aoline35 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line36 /sybase_dev/Aoline36 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line37 /sybase_dev/Aoline37 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line38 /sybase_dev/Aoline38 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line39 /sybase_dev/Aoline39 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE order_line40 /sybase_dev/Aoline40 9150
    db=tpcc size=9150
  segment=Sorder_line
DEVICE_END

DEVICE orders1 /sybase_dev/Aorder1 2100
    db=tpcc size=2100
  segment=Sorders
DEVICE_END

DEVICE orders2 /sybase_dev/Aorder2 2100
    db=tpcc size=2100
  segment=Sorders
DEVICE_END

DEVICE orders3 /sybase_dev/Aorder3 2100
    db=tpcc size=2100
  segment=Sorders
DEVICE_END

DEVICE orders4 /sybase_dev/Aorder4 2100
    db=tpcc size=2100
  segment=Sorders
DEVICE_END

DEVICE orders5 /sybase_dev/Aorder5 2100
    db=tpcc size=2100
  segment=Sorders
DEVICE_END

DEVICE orders6 /sybase_dev/Aorder6 2100
    db=tpcc size=2100
  segment=Sorders
DEVICE_END

DEVICE orders7 /sybase_dev/Aorder7 2100
    db=tpcc size=2100
  segment=Sorders
DEVICE_END

DEVICE orders8 /sybase_dev/Aorder8 2100
    db=tpcc size=2100
  segment=Sorders
DEVICE_END

DEVICE wdino1 /sybase_dev/Awdino1 750
    db=tpcc size=750
  segment=Scache
DEVICE_END

DEVICE wdino2 /sybase_dev/Awdino2 750
    db=tpcc size=750
  segment=Scache
DEVICE_END

DEVICE wdino3 /sybase_dev/Awdino3 750
    db=tpcc size=750
  segment=Scache
DEVICE_END

DEVICE history1 /sybase_dev/Ahist1 4750
    db=tpcc size=4750
  segment=Shistory
DEVICE_END

DEVICE history2 /sybase_dev/Ahist2 4750
    db=tpcc size=4750
  segment=Shistory
DEVICE_END

DEVICE history3 /sybase_dev/Ahist3 4750
    db=tpcc size=4750
  segment=Shistory
DEVICE_END

DEVICE history4 /sybase_dev/Ahist4 4750
    db=tpcc size=4750
  segment=Shistory
DEVICE_END

DEVICE history5 /sybase_dev/Ahist5 4750
    db=tpcc size=4750
  segment=Shistory
DEVICE_END
```

```
DEVICE history6 /sybase_dev/Ahist6 4750
    db=tpcc size=4750
  segment=Shistory
DEVICE_END

DEVICE history7 /sybase_dev/Ahist7 4750
    db=tpcc size=4750
  segment=Shistory
DEVICE_END

DEVICE tpcc_log1 /sybase_dev/Alogs1 30000
    db=tpcc size=30000
  log
DEVICE_END

DEVICE tpcc_log2 /sybase_dev/Alogs2 30000
    db=tpcc size=30000
  log
DEVICE_END
```

dumptran_server.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)dumptran_server.sh1.299/10/13SMI"
#
isql -Usa -P$PASSWORD << EOF
dbcc tune(maxwritedes, 2000)
go
dump tran $1 with truncate_only
go
use $1
go
checkpoint
go
dbcc tune(maxwritedes, 10)
go
EOF
```

error.c

```
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)error.c1.296/01/09SMI"

#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
*/

/*
** error.c:1.14/30/9119:47:32
**Standard error handler for RungenII and supporting code
**
**HMS [04/30/91]
*/

/* Required standard include files */
#include <stdio.h>

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

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

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 %d: \n\t%s\n",errno, dberrstr(errno));

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

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



int
msg_handler(dbproc,msgno,msgstate,severity,msgtext,servername,procname,line)
DBPROCESS*dbproc;
intmsgno;
intmsgstate;
intseverity;
char*msgtext;
char            *servername;
char            *procname;
int             line;
{

/* changing database messages */
if (msgno == DUMB_MESSAGE || msgno == ABORT_ERROR || msgno == 5703 || msgno == 5704)
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 */
sleep((unsigned) 2);
return(SUCCEED);

}

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

/* exit on any error */
exit(-101);
}
```

fdr_tune.sh

```
:
echo "-- $0 --"
echo "............."
echo "SYBASE=$SYBASE"
echo "DSLISTEN=$DSLISTEN DSQUERY=$DSQUERY"
echo "............."
isql -w1044 -n -e -Usa -P << EOF
select getdate()
go
use tpcc
go
dbcc tune (cleanup, 0)
go
dbcc tune (doneinproc, 0)
go
dbcc tune (maxwritedes, 50)
go
sp_setpglockpromote "table", "tpcc..item", 2, 2, 0
go
select getdate()
go
EOF
```

finish_indexes.sh

```
:
:
PROG=`basename $0 .sh`
DATE=`date +%y%m%d:%H%M`
LOG=$PROG.$DATE.log
(
date
isql -w144 -Usa -P -e -n <<EOF
use tpcc
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips, 100, new_order)
go
dbcc tune(ascinserts, 1, orders)
go
dbcc tune(oamtrips, 100, orders)
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go
dbcc tune(indextrips, 10, item)
go
dbcc tune(indextrips, 10, stock)
go
dbcc tune(indextrips, 100, warehouse)
go
dbcc tune(indextrips, 100, district)
go
checkpoint
go
use master
go
dump tran tpcc with no_log
go
use tpcc
go
checkpoint
go
use master
go
dump tran tpcc with truncate_only
go
checkpoint
go
EOF
) >> $LOG 2>&1
```

generic_procs.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
#
# ident "@(#)generic_procs.sh1.296/06/13SMI"
#
isql -Usa -P$PASSWORD <<EOF
use ${1:-tpcc}
go

if exists (select * from sysobjects where name = "indexes" and type = 'P')
drop proc indexes
go
if exists (select * from sysobjects where name = "rowsize" and type = 'P')
drop proc rowsize
go
if exists (select * from sysobjects where name = "spaceused" and type = 'P')
drop proc spaceused
go
if exists (select * from sysobjects where name = "columns" and type = 'P')
drop proc columns
go

create proc indexes as
selectname, indid, ipgtrips, status2
fromsysindexes
whereid > 1000
and indid > 0
go

create proc rowsize @table_name varchar(30) = NULL as
select
o.name,
bytes = sum(c.length),
rowsPerPage = 2016/(sum(c.length)+4)
from
sysobjects o, syscolumns c
where
(@table_name = NULL OR o.name = @table_name) AND
o.type = 'U' AND
o.id>999 AND
o.id = c.id
group by
o.id
go

create proc spaceused as
declare@table_namechar(30)
declare@eot_soint
declare c_msu_so cursor for
select name
from sysobjects
whereid>9999 and
type = 'U'
begin
open c_msu_so
select @eot_so = 0
while (@eot_so = 0) begin
fetch c_msu_so into @table_name
if (@@sqlstatus > 0) begin
select @eot_so = 1
break
end
exec sp_spaceused @table_name
end
end
go

create proc columns @table_name varchar(30) as
select
c.name, c.length, c.type, c.prec, c.scale
from
sysobjects o, syscolumns c
where
o.name = @table_name AND
o.type = 'U' AND
o.id = c.id
go
EOF
```

hist_part.sh

```
:
echo "-- $0 --"
echo "............."
echo "SYBASE=$SYBASE"
echo "DSLISTEN=$DSLISTEN DSQUERY=$DSQUERY"
echo "............."
isql -w144 -n -e -Usa -P << EOF
use tpcc
go
alter table history unpartition
go
alter table history partition 768
go
select getdate()
go
EOF
```

load.c

```
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)load.c1.196/01/09SMI"

typedefunsigned longBitVector;
#define WSZ(sizeof(BitVector)*8)
/* For axposf use WAREBATCH 0f 150 */
#ifndef WAREBATCH
#defineWAREBATCH200
#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;
{

/* Setup to use  the dblib version 10 for numeric datatypes */
       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(10, 20) / 100.0;
w_ytd = 300000.00;

warehouse_load();

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


begin_warehouse_load()
{
inti = 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(10,20) / 100.0;

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



begin_district_load()
{
inti = 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()
{
inti = 1;

bulk_i = bulk_open("tpcc", "item", password);

/* bind the variables to the sybase columns */
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()
{
inti = 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;/* is this money or long or float? */
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];
IDlen;

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_last);
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, 50) / 100.0;
c_balance = -10.0;
customer_load();
}
}
}

begin_customer_load()
{
inti = 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_ol_cnt = RandomNumber(5, 15); */
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;
ol_delivery_d = o_entry_d;
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()
{
inti = 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()
{
inti = 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()
{
inti = 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()
{
inti = 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 || w2 > 1000000 || 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"; */
  "abcedfghijklmnopqrstuvwxyz";
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;
}
```

rand.sh

```
isql -Usa -P << EOF
dbcc tune(log_prealloc, 4, "on")
go
dbcc traceon(1131)
go
dbcc tune(des_greedyalloc, 4, "delivery", "on")
go
dbcc tune(des_greedyalloc, 4, "stock", "on")
go
dbcc tune(des_greedyalloc, 4, "customer", "on")
go
dbcc tune(des_greedyalloc, 4, "new_order", "on")
go
dbcc tune(des_greedyalloc, 4, "orders", "on")
go
dbcc tune(des_greedyalloc, 4, "order_line", "on")
go
EOF
```

shutdown_server.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)shutdown_server.sh1.196/01/09SMI"
#
# Now shut the server down
isql -Usa -P$PASSWORD <<EOF
shutdown
go
EOF
```

stock.sh

```
:
PROG=`basename $0 .sh`
DATE=`date +%y%m%d:%H%M`
LOG=$PROG.$DATE.log
(
date
TABLE=stock
SEGMENT=Sstock
INDEX=s_clu
FIELDS="s_i_id, s_w_id"
isql -w144 -Usa -P -e -n <<EOF
use tpcc
go
sp_spaceused $TABLE
go
drop index $TABLE.$INDEX
go
sp_spaceused $TABLE
go
select getdate()
go
set statistics io on
go
create unique clustered index $INDEX
on $TABLE($FIELDS)
with sorted_data
on $SEGMENT
go
set statistics io off
go
select getdate()
go
sp_spaceused $TABLE
go
EOF
) >> $LOG 2>&1
```

tpcc_admin.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)tpcc_admin.sh1.196/01/09SMI"
#

dbname=${1:-tpcc}

#cat << EOF
isql -Usa -P$PASSWORD <<EOF
use $dbname
go


if exists (select * from sysobjects where name = 'tc_infotab' and type = 'U')
        drop table tc_infotab
go
create table tc_infotab
(
        clients         int     not null,
        duration        int     not null,
        engines         int     not null,
        warehouses      int     not null,
        arrived         int     not null,
        deliveryRatio   int     not null
)
go

if exists (select * from sysobjects where name = 'tc_stattab' and type = 'U')
        drop table tc_stattab
go
create table tc_stattab
(
        name            char(30)                not null,
        total           int      default 0      not null,
        limit           int      default 0      not null,
        overlimit       int      default 0      not null,
        deadlock        int      default 0      not null,
        sumResp         float    default 0      not null,
        sumRespSq       float    default 0      not null,
        filler1         char(255) default ""    not null,
        filler2         char(255) default ""    not null,
        filler3         char(255) default ""    not null,
        filler4         char(255) default ""    not null,

        unique(name)
)
go

if exists (select * from sysobjects where name = 'tc_synctab' and type = 'U')
drop table tc_synctab
go
create table tc_synctab
(
junkint default 0not null,
)
go

if exists (select * from sysobjects where name = 'tc_initialize' and type = 'P')
drop proc tc_initialize
go
create proc tc_initialize
@clientsint,
@durationint = 600,
@enginesint = 0,
@warehousesint = 0,
@deliveryRatioint
as
declare@availwareint
declare@onlineint

select @availware = count(*) from warehouse

if (@availware < @warehouses or @warehouses = 0)
select @warehouses = @availware

select @online = count(*)
from master..sysengines
where status = "online"
while (@engines > 0 and @engines > @online)
begin
dbcc engine("online")
select @online = @online + 1
end
while (@engines > 0 and @engines < @online)
begin
dbcc engine("offline")
select @online = @online + 1
end

begin transaction
delete from tc_infotab
insert into tc_infotab( clients, arrived,  duration,
                                engines,  warehouses, deliveryRatio)
values(@clients,       1, @duration, @online, @warehouses, @deliveryRatio)
commit transaction

print "Using %1! warehouses (out of %2!), %3! engines, and %4! users for %5!
seconds",
@warehouses, @availware, @online, @clients, @duration

begin transaction
delete from tc_synctab
insert into tc_synctab(junk) values (0)
commit transaction

begin transaction
delete from tc_stattab
commit transaction

select engines, warehouses from tc_infotab
go

if exists (select * from sysobjects where name = 'tc_initstat' and type = 'P')
drop proc tc_initstat
go
create proc tc_initstat
@namechar(30),
@limitint
as
begin
begin transaction
insert into tc_stattab(name,limit) values (@name, @limit)
commit transaction
end
go

if exists (select * from sysobjects where name = 'tc_startup' and type = 'P')
drop proc tc_startup
go
create proc tc_startup
as
declare@junkint
begin
update tc_infotab set arrived = arrived+1
select @junk=junk from tc_synctab
select engines, warehouses from tc_infotab
end
go

if exists (select * from sysobjects where name = 'tc_addstats' and type = 'P')
drop proc tc_addstats
go
create proc tc_addstats
@namechar(30),
@totalint,
@overlimitint,
@deadlockint,
@sumRespfloat,
@sumRespSqfloat
as
begin
begin transaction
update tc_stattab set
total  = total+ @total,
overlimit = overlimit+ @overlimit,
deadlock  = deadlock+ @deadlock,
sumResp  = sumResp+ @sumResp,
sumRespSq = sumRespSq+ @sumRespSq
where name = @name
commit transaction
end
go

if exists (select * from sysobjects where name = 'showstats' and type = 'P')
drop proc showstats
go
create proc showstats
as
declare@alltotalfloat,
@minutesfloat
begin
select @alltotal = sum(total)
from tc_stattab
where not name = "delivery"
select @minutes = convert(float,duration)/60.0 from tc_infotab
select
convert(char(20), name) as Name,
convert(decimal(6,0),total/@minutes) as PerMin,
convert(decimal(6,3),sumResp/(total*1000.0)) as Average,
convert(decimal(6,3),
```

```
(sumRespSq-sumResp*sumResp/total)/(total*1000000.0))
as Variance,
convert(decimal(6,3), convert(float, limit)/1000.0) as Limit,
convert(decimal(6,3),
(convert(float, overlimit)*100.0)/total) as PercOver,
convert(decimal(4,0),deadlock) as Deadlocks,
convert(decimal(5,2),total/@alltotal*100.0) as PercMix
    fromtc_stattab
    wheretotal>0
end
go

if exists (select * from sysobjects where name = 'spread')
drop view spread
go
create view spread as
selectd_w_id ware,
d_id dist,
(select min(no_o_id) from new_order
whereno_w_id = d.d_w_id
andno_d_id = d.d_id) first,
d_next_o_id last
from district d
go
EOF
```

tpcc_cache_bind.sh

```
#!/bin/sh -f

# assumes single user mode, so make sure no one else is logged on

isql -Usa -P$PASSWORD -e << EOF

use master
go
sp_dboption tpcc, "single user", true
go

use tpcc
go
checkpoint
go

/*
** Cache c_log
*/

sp_bindcache "c_log", "tpcc", "syslogs"
go

/*
** Cache c_tinyhot
*/

sp_bindcache "c_tinyhot", "tpcc", "sysgams"
go
sp_bindcache "c_tinyhot", "tpcc", "sysgams", "sysgams"
go
sp_bindcache "c_tinyhot", "tpcc", "sysindexes"
go
sp_bindcache "c_tinyhot", "tpcc", "sysindexes","sysindexes"
go

/*
** Cache c_tinyhot (continued)
*/

sp_bindcache "c_tinyhot", "tpcc", "warehouse"
go
sp_bindcache "c_tinyhot", "tpcc", "warehouse","w_clu"
go
sp_bindcache "c_tinyhot", "tpcc", "district"
go
sp_bindcache "c_tinyhot", "tpcc", "district","d_clu"
go
sp_bindcache "c_tinyhot", "tpcc", "item"
go
sp_bindcache "c_tinyhot", "tpcc", "item","i_clu"
go

/*
** Cache c_reserved
*/

sp_bindcache "c_history", "tpcc", "history"
go

/*
** Cache c_new_order
*/
sp_bindcache "c_new_order", "tpcc", "new_order", "no_clu"
go
sp_bindcache "c_new_order", "tpcc", "new_order"
go

/*
** Cache c_order_line
*/

sp_bindcache "c_order_line", "tpcc", "order_line"
go
sp_bindcache "c_order_line", "tpcc", "order_line", "ol_clu"
go

/*
** Cache c_orders
*/

sp_bindcache "c_orders", "tpcc", "orders"
go
sp_bindcache "c_orders", "tpcc", "orders","o_clu"
go

/*
** Cache c_stock_index
*/

sp_bindcache "c_stock_index", "tpcc", "stock","s_clu"
go

/*
** Cache c_stock
*/

sp_bindcache "c_stock", "tpcc", "stock"
go

/*
** Cache c_customer
*/

sp_bindcache "c_customer", "tpcc", "customer"
go
sp_bindcache "c_index", "tpcc", "customer","c_clu"
go
sp_bindcache "c_index_nc", "tpcc", "customer","c_non1"
go

use master
go
sp_dboption tpcc, "single user", false
go

use tpcc
go
checkpoint
go


EOF
```

tpcc_indexes.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)tpcc_indexes.sh1.196/01/09SMI"
#

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

create unique clustered index w_clu
on warehouse(w_id)
with fillfactor = 1
on Scache
go
dbcc tune(indextrips, 100, warehouse)
go

create unique clustered index d_clu
on district(d_w_id, d_id)
with fillfactor = 1
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 Scidx
go
select getdate()
go

checkpoint
go
EOF
```

tpcc_load.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)tpcc_load.sh1.196/01/09SMI"
#
if [ $# -ne 2 ];then
   echo "Udage: $0 start_warehouse_number endware_house_number"
   exit
fi

LD_LIBRARY_PATH=/export/home/sybase/lib:/usr/lib:/export/home/sybase/11.9/lib

export LD_LIBRARY_PATH

echo load started at `date` ........
echo Lib path is    $LD_LIBRARY_PATH
echo
```

```
#LOAD=/export/home/sybase/tpcc11/loader/load
LOAD=/export/home/dbbench/vendors/sybase/TPCSO/bin/load

export LOAD

ulimit -s 2097148

ulimit -a

echo
echo

# Only have to do this once
if [ $1 -eq 1 ]; then
    $LOAD item 1 1 $PASSWORD
fi

for i in \
        new_order \
        warehouse \
        district
do
        ( echo `date` "Started background LOAD for $i warehouse $1 to $2 --- " >
load_${i}.$$.log
        $LOAD $i $1 $2 $PASSWORD >> load_${i}.$$.log
        echo `date` "Finished background LOAD for $i --- "  >> load_${i}.$$.log ) &
done

# loading customer, history, stock, orders, and order-line
/export/home/dbbench/vendors/sybase/TPCSO/scripts/tpcc_load_history.sh $1 $2 &
/export/home/dbbench/vendors/sybase/TPCSO/scripts/tpcc_load_customer.sh $2 $2 &
/export/home/dbbench/vendors/sybase/TPCSO/scripts/tpcc_load_stock.sh $1 $2 &
/export/home/dbbench/vendors/sybase/TPCSO/scripts/tpcc_load_orders.sh $1 $2 &

wait

date
```

tpcc_load_customer.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)tpcc_load.sh1.196/01/09SMI"
#
if [ $# -ne 2 ];then
   echo "Udage: $0 start_warehouse_number endware_house_number"
   exit
fi

LD_LIBRARY_PATH=/export/home/sybase/lib:/usr/lib:/export/home/sybase/11.9/lib

export LD_LIBRARY_PATH

echo load started at `date` ........
echo Lib path is    $LD_LIBRARY_PATH
echo

LOAD=/export/home/dbbench/vendors/sybase/TPCSO/bin/load

export LOAD

ulimit -s 2097148

ulimit -a
#echo `date` "Started background LOAD for $i --- "
#$LOAD customer $1 $2 $PASSWORD
#echo `date` "Finished background LOAD for $i --- "

START=$1
END=$2
INCR=1000
LAST=`expr $INCR - 1`
II=`expr $START + $LAST`
echo `date` "Started background LOAD for customer warehouse $START to $END --- " >
load_customer.$$.log
while   [ $START -lt $END ]; do
    if [ $II -gt $END ]; then II=$END; fi
    $LOAD customer $START $II $PASSWORD >> load_customer.$$.log &
    wait
    echo `date` "Finished background LOAD for customer from $START through $II" >>
load_customer.$$.log
    START=`expr $START + $INCR`
    II=`expr $II + $INCR`
done

wait


date
```

tpcc_load_history.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)tpcc_load.sh1.196/01/09SMI"
#
if [ $# -ne 2 ];then
   echo "Udage: $0 start_warehouse_number endware_house_number"
   exit
fi

LD_LIBRARY_PATH=/export/home/sybase/lib:/usr/lib:/export/home/sybase/11.9/lib

export LD_LIBRARY_PATH

echo load started at `date` ........
echo Lib path is    $LD_LIBRARY_PATH
echo

LOAD=/export/home/dbbench/vendors/sybase/TPCSO/bin/load

export LOAD

ulimit -s 2097148

ulimit -a
#echo `date` "Started background LOAD for $i --- "
#$LOAD history $1 $2 $PASSWORD
#echo `date` "Finished background LOAD for $i --- "

START=$1
END=$2
INCR=1000
LAST=`expr $INCR - 1`
II=`expr $START + $LAST`
echo `date` "Started background LOAD for history warehouse $START to $END --- " >
load_history.$$.log
while   [ $START -lt $END ]; do
    if [ $II -gt $END ]; then II=$END; fi
    $LOAD history $START $II $PASSWORD >> load_history.$$.log &
    wait
    echo `date` "Finished background LOAD for history from $START through $II" >>
load_history.$$.log
    START=`expr $START + $INCR`
    II=`expr $II + $INCR`
done

wait


date
```

tpcc_load_orders.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)tpcc_load.sh1.196/01/09SMI"
#
if [ $# -ne 2 ];then
   echo "Udage: $0 start_warehouse_number endware_house_number"
   exit
fi

LD_LIBRARY_PATH=/export/home/sybase/lib:/usr/lib:/export/home/sybase/11.9/lib

export LD_LIBRARY_PATH

echo load started at `date` ........
echo Lib path is    $LD_LIBRARY_PATH
echo

LOAD=/export/home/dbbench/vendors/sybase/TPCSO/bin/load

export LOAD

ulimit -s 2097148

ulimit -a
#echo `date` "Started background LOAD for $i --- "
#$LOAD orders $1 $2 $PASSWORD
#echo `date` "Finished background LOAD for $i --- "

START=$1
END=$2
INCR=1000
LAST=`expr $INCR - 1`
II=`expr $START + $LAST`
echo `date` "Started background LOAD for orders/order-line warehouse $START to $END -
-- " > load_orders.$$.log
while   [ $START -lt $END ]; do
    if [ $II -gt $END ]; then II=$END; fi
    $LOAD orders $START $II $PASSWORD >> load_orders.$$.log &
    wait
    echo `date` "Finished background LOAD for orders/order-line from $START through
$II" >> load_orders.$$.log
    START=`expr $START + $INCR`
    II=`expr $II + $INCR`
done

wait


date
```

tpcc_load_stock.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)tpcc_load.sh1.196/01/09SMI"
#
if [ $# -ne 2 ];then
   echo "Udage: $0 start_warehouse_number endware_house_number"
   exit
fi
```

```
LD_LIBRARY_PATH=/export/home/sybase/lib:/usr/lib:/export/home/sybase/11.9/lib

export LD_LIBRARY_PATH

echo load started at `date` ........
echo Lib path is    $LD_LIBRARY_PATH
echo

LOAD=/export/home/dbbench/vendors/sybase/TPCSO/bin/load

export LOAD

ulimit -s 2097148

ulimit -a
#echo `date` "Started background LOAD for $i --- "
#$LOAD stock $1 $2 $PASSWORD
#echo `date` "Finished background LOAD for $i --- "

START=$1
END=$2
INCR=200
LAST=`expr $INCR - 1`
II=`expr $START + $LAST`
echo `date` "Started background LOAD for stock warehouse $START to $END --- " >
load_stock.$$.log
while   [ $START -lt $END ]; do
    if [ $II -gt $END ]; then II=$END; fi
    $LOAD stock $START $II $PASSWORD >> load_stock.$$.log &
    wait
    echo `date` "Finished background LOAD for stock from $START through $II" >>
load_stock.$$.log
    START=`expr $START + $INCR`
    II=`expr $II + $INCR`
done

wait


date
```

tpcc_proc.sh

```
#!/bin/sh -f
################################################################################
#
# 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!
#
#
################################################################################
#

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

CREATE PROC neworder_local (
@w_idsmallint,
@d_idtinyint,
@c_idint,
@o_ol_cntint,

@i_idint = 0, @ol_qtysmallint = 0,
@i_id2int = 0, @ol_qty2smallint = 0,
@i_id3int = 0, @ol_qty3smallint = 0,
@i_id4int = 0, @ol_qty4smallint = 0,
@i_id5int = 0, @ol_qty5smallint = 0,
@i_id6int = 0, @ol_qty6smallint = 0,
@i_id7int = 0, @ol_qty7smallint = 0,
@i_id8int = 0, @ol_qty8smallint = 0,
@i_id9int = 0, @ol_qty9smallint = 0,
@i_id10int = 0, @ol_qty10smallint = 0,
@i_id11int = 0, @ol_qty11smallint = 0,
@i_id12int = 0, @ol_qty12smallint = 0,
@i_id13int = 0, @ol_qty13smallint = 0,
@i_id14int = 0, @ol_qty14smallint = 0,
@i_id15int = 0, @ol_qty15smallint = 0
)
as

declare
@w_taxreal,@d_taxreal,
@c_lastchar(16),@c_creditchar(2),
@c_discountreal,@commit_flagint,
@c_ins_id int,@local_d_idint,
@i_pricefloat,
@i_namechar(24),@i_datachar(50),

@s_quantitysmallint,@ten_smallintsmallint,
@s_ytdint,@s_order_cntint,
@s_distchar(24),@s_datachar(50),
@one_smallintsmallint,@zero_smallint smallint,
@ninenine_smallint smallint,

@ol_numberint,@o_idint,
@o_entry_ddatetime,@b_gchar(1),
@ol_amount float



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
, @local_d_id= @d_id
, @ten_smallint= 10
, @zero_smallint = 0
, @ninenine_smallint= 99
, @one_smallint= 1
, @o_entry_d= getdate()
WHERED_w_id= @w_id
 ANDd_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

      /* set i_id, ol_qty for this lineitem */
      /* this is replaced by case statement */

      /* convert c_no_is cursor to a simple select */
      /* get item data (no one update item) */

      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
         /*Otherwise if the item is found */
         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 < @ten_smallint)
                        then @ninenine_smallint else @zero_smallint end,
                    s_quantity = s_quantity - @ol_qty +
                         case when (s_quantity - @ol_qty < @ten_smallint)
                         then @ninenine_smallint  else @zero_smallint end,
           s_order_cnt = s_order_cnt + @one_smallint,
                  @s_data = s_data,
                  @s_dist = case @local_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
       if (@@rowcount = 0)
           begin
             select @commit_flag = 0
             select NULL, NULL, NULL, NULL, NULL
             continue
    end
         /*Otherwise if the Stock is found */

  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, "18000101", @ol_qty,
       @ol_amount, @s_dist)


          /* send line-item data to client */
          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 /* while */

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/* Return to client */
@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_idsmallint,
@d_idtinyint,
@c_idint,
@o_ol_cntint,

@i_idint = 0, @s_w_idsmallint = 0, @ol_qtysmallint = 0,
@i_id2int = 0, @s_w_id2smallint = 0, @ol_qty2smallint = 0,
@i_id3int = 0, @s_w_id3smallint = 0, @ol_qty3smallint = 0,
@i_id4int = 0, @s_w_id4smallint = 0, @ol_qty4smallint = 0,
@i_id5int = 0, @s_w_id5smallint = 0, @ol_qty5smallint = 0,
@i_id6int = 0, @s_w_id6smallint = 0, @ol_qty6smallint = 0,
@i_id7int = 0, @s_w_id7smallint = 0, @ol_qty7smallint = 0,
@i_id8int = 0, @s_w_id8smallint = 0, @ol_qty8smallint = 0,
@i_id9int = 0, @s_w_id9smallint = 0, @ol_qty9smallint = 0,
@i_id10int = 0, @s_w_id10smallint = 0, @ol_qty10smallint = 0,
@i_id11int = 0, @s_w_id11smallint = 0, @ol_qty11smallint = 0,
@i_id12int = 0, @s_w_id12smallint = 0, @ol_qty12smallint = 0,
@i_id13int = 0, @s_w_id13smallint = 0, @ol_qty13smallint = 0,
@i_id14int = 0, @s_w_id14smallint = 0, @ol_qty14smallint = 0,
@i_id15int = 0, @s_w_id15smallint = 0, @ol_qty15smallint = 0
)
as

declare
@w_taxreal,@d_taxreal,
@c_lastchar(16),@c_creditchar(2),
@c_discountreal,@commit_flagtinyint,
@c_ins_id int,@local_d_idint,

@i_pricefloat,
@i_namechar(24),@i_datachar(50),

@s_quantitysmallint,@ten_smallintsmallint,
@s_ytdint,@s_order_cntint,
@s_distchar(24),@s_datachar(50),
@one_smallintsmallint,@zero_smallint smallint,
@ninenine_smallint smallint,

@ol_numberint,@o_idint,
@o_entry_ddatetime,@b_gchar(1),
@ol_amount float


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
, @local_d_id= @d_id
, @ten_smallint= 10
, @zero_smallint = 0
, @ninenine_smallint= 99
, @one_smallint= 1
, @o_entry_d= getdate()
WHERED_w_id= @w_id
  ANDd_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

 /* convert c_no_is cursor to a simple select */
 /* get item data (no one update item) */
 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
  /* Otherwise if the item is found */
     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 < @ten_smallint)
                then @ninenine_smallint else @zero_smallint end,
           s_quantity = s_quantity - @ol_qty +
                case when (s_quantity - @ol_qty < @ten_smallint)
                then @ninenine_smallint  else @zero_smallint end,
   s_order_cnt = s_order_cnt + @one_smallint,
         @s_data = s_data,
         @s_dist = case @local_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_remote_cnt = s_remote_cnt +
   case when (@s_w_id = @w_id)
                       then 0 else 1 end
              where s_w_id = @s_w_id and
                    s_i_id = @i_id

              if (@@rowcount = 0)
                begin
                  select @commit_flag = 0
                  select NULL, NULL, NULL, NULL, NULL
                  continue
                end

  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,
       @s_w_id, "18000101", @ol_qty,
       @ol_amount, @s_dist)

        /* send line-item to client */
        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 /* while */


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/* Return to client */
@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_idsmallint,@c_w_idsmallint,
@h_amount float,
@d_idtinyint,@c_d_idtinyint,
@c_idint
as
declare@c_lastchar(16)

declare@w_street_1char(20),@w_street_2char(20),
@w_citychar(20),@w_statechar(2),
@w_zipchar(9),@w_namechar(10),
@w_ytd float,@w_id_retrieved smallint

declare@d_street_1char(20),@d_street_2char(20),
@d_citychar(20),@d_statechar(2),
@d_zipchar(9),@d_namechar(10),
@d_ytd float,@commit_flgint

declare@c_firstchar(16),@c_middlechar(2),
@c_street_1char(20),@c_street_2char(20),
@c_citychar(20),@c_statechar(2),
@c_zipchar(9),@c_phonechar(16),
@c_sincedatetime,@c_creditchar(2),
@c_credit_limnumeric(12,0),@c_balancefloat,
@c_discountreal,
@data1char(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
   ,@commit_flg = 1
WHERE d_w_id= @w_id
ANDd_id= @d_id

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

if (@@rowcount = 0)
   begin
select @commit_flg = 0
   end

/* Customer data */
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 (@@rowcount = 0)
   begin
select @commit_flg = 0
   end

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 /* if */

/* Create the history record */
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_retrieved,
@today, @h_amount, (@w_name + "    " + @d_name))

/* COMMIT TRANSACTION PID */

if (@commit_flg = 1)
            COMMIT TRANSACTION PID
          else
      ROLLBACK TRANSACTION PID

select/* Return to client */
@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_idsmallint,@c_w_idsmallint,
@h_amount float,
@d_idtinyint,@c_d_idtinyint,
@c_lastchar(16)
as
declare@nint,@c_idint

declare@w_street_1char(20),@w_street_2char(20),
@w_citychar(20),@w_statechar(2),
@w_zipchar(9),@w_namechar(10),
@w_ytd float,@w_id_retrieved smallint

declare@d_street_1char(20),@d_street_2char(20),
@d_citychar(20),@d_statechar(2),
@d_zipchar(9),@d_namechar(10),
@d_ytd float,@commit_flgint

declare@c_firstchar(16),@c_middlechar(2),
@c_street_1char(20),@c_street_2char(20),
@c_citychar(20),@c_statechar(2),
@c_zipchar(9),@c_phonechar(16),
@c_sincedatetime,@c_creditchar(2),
@c_credit_limnumeric(12,0),@c_balancefloat,
@c_discountreal,
@data1char(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
   ,@commit_flg = 1
WHERE d_w_id= @w_id
ANDd_id= @d_id

if (@@rowcount = 0)
   begin
select @commit_flg = 0
   end

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

/* Customer data */
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 (@@rowcount = 0)
   begin
select @commit_flg = 0
   end

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 /* if */

/* Create the history record */
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_retrieved,
@today, @h_amount, (@w_name + "    " + @d_name))

/* COMMIT TRANSACTION PNM */

if (@commit_flg = 1)
            COMMIT TRANSACTION PNM
         else
      ROLLBACK TRANSACTION PNM

select/* Return to client */
@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_idsmallint,
@d_idtinyint,
@c_idint
as

DECLARE@o_idint,
@o_entry_ddatetime,
@o_carrier_idsmallint

BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
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
ANDo_d_id= @d_id
ANDo_c_id= @c_id
ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
set rowcount 0

/* Select order lines for the current order */
select/* Return multiple rows to client */
ol_supply_w_id,
ol_i_id,
ol_quantity,
ol_amount,
```

```
ol_delivery_d
FROMorder_line HOLDLOCK
WHEREol_o_id = @o_id
ANDol_d_id = @d_id
ANDol_w_id = @w_id

select/* Return single row to client */
@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
ANDc_d_id  = @d_id
ANDc_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_idsmallint,
@d_idtinyint,
@c_lastchar(16)
as

DECLARE@o_idint,
@o_entry_ddatetime,
@o_carrier_idsmallint

declare@nint,@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


/* Get the latest order made by the customer */
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
ANDo_d_id= @d_id
ANDo_c_id= @c_id
ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
set rowcount 0

/* Select order lines for the current order */
select/* Return multiple rows to client */
ol_supply_w_id,
ol_i_id,
ol_quantity,
ol_amount,
ol_delivery_d
FROMorder_line HOLDLOCK
WHEREol_o_id = @o_id
ANDol_d_id = @d_id
ANDol_w_id = @w_id

select/* Return single row to client */
@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
ANDc_d_id  = @d_id
ANDc_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,@ten_tinyinttinyint,
@one_tinyinttinyint,@one_smallintsmallint,
        @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

        /*
        ** The only purpose of the index hint in the above is to ensure
        ** that the clustered index is used.  As it turns out, our optimizer
        ** chooses the clustered index anyway -- with or without the hint.
        */
begin
    SELECT @one_tinyint = 1, @ten_tinyint = 10, @one_smallint = 1
    while (@d_id <= @ten_tinyint ) 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 + @one_smallint
                WHERE   c_id    = @o_c_id
                AND     c_d_id  = @d_id
                AND     c_w_id  = @w_id

                COMMIT TRANSACTION DEL

                select      /* Return to client */
                        @no_o_id
        end
SELECT @d_id = @d_id + @one_tinyint
    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 /* Return to client */
FROMdistrict,
order_line (index ol_clu prefetch 2 lru),
stock (index s_clu prefetch 2 lru)
WHERED_w_id=@w_id
ANDd_id=@d_id
ANDol_w_id=@w_id
ANDol_d_id=@d_id
ANDol_o_idbetween (d_next_o_id - 20) and (d_next_o_id - 1)
ANDs_w_id=ol_w_id
ANDs_i_id=ol_i_id
AND     s_quantity < @threshold
go
EOF

if [ "$?" != "0" ]
then
echo ""
echo " **** WARNING: Could not connect to server to install procs! ****"
echo " ****      SQL SErver must have failed!!! ****"
echo " ****      TPC-C build & run will fail!!! ****"
echo ""
fi
```

tpcc_proc_tax.sh

```
################################################################################
#
# tpcc_proc_case.sh
#
################################################################################
#
# 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
#
isql -Usa -P$PASSWORD -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_idsmallint,
@d_idtinyint,
@c_idint,
@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_taxreal,@d_taxreal,
@c_lastchar(16),@c_creditchar(2),
@c_discountreal,@commit_flagtinyint,

@i_pricereal,
@i_namechar(24),@i_datachar(50),

@s_quantitysmallint,
@s_ytdint,@s_order_cntint,
@s_distchar(24),@s_datachar(50),

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

declare c_no_wdc CURSOR FOR
SELECTw_tax, d_tax, d_next_o_id,
c_last, c_discount, c_credit
,1,0
,getdate()
FROMdistrict HOLDLOCK,
warehouse HOLDLOCK,
customer (index c_clu prefetch 2 lru) HOLDLOCK
WHEREd_w_id= @w_id
ANDd_id= @d_id
ANDw_id= d_w_id
ANDc_w_id= d_w_id
ANDc_d_id= d_id
ANDc_id= @c_id
FOR UPDATE OF d_next_o_id

begin

begin transaction NO

OPEN c_no_wdc
FETCH c_no_wdc INTO
@w_tax, @d_tax, @o_id,
@c_last, @c_discount, @c_credit
,@commit_flag, @ol_number
,@o_entry_d
UPDATEdistrict
SETd_next_o_id = @o_id + 1
WHERE CURRENT OF c_no_wdc
CLOSE c_no_wdc

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

      /* set i_id, ol_qty for this lineitem */
      /* this is replaced by case statement */

      /* convert c_no_is cursor to a simple select */
      /* get item data (no one update item) */

      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

         /*Otherwise if the item is found */
         update stock
            set s_ytd = s_ytd + @ol_qty,
          @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

  select @ol_amount = @ol_qty * @i_price

  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)


          /* send line-item data to client */
          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 /* while */

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_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)

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

select/* Return to client */
@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_idsmallint,
@d_idtinyint,
@c_idint,
@o_ol_cnttinyint,

@i_idint = 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_taxreal,@d_taxreal,
@c_lastchar(16),@c_creditchar(2),
@c_discountreal,@commit_flagtinyint,

@i_pricereal,
@i_namechar(24),@i_datachar(50),

@s_quantitysmallint,
@s_ytdint,@s_order_cntint,
@s_distchar(24),@s_datachar(50),
@s_remote_cntint,@remoteint,

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

declare c_no_wdc CURSOR FOR
SELECTw_tax, d_tax, d_next_o_id,
c_last, c_discount, c_credit
,1,0
,getdate()
FROMdistrict HOLDLOCK,
warehouse HOLDLOCK,
customer (index c_clu prefetch 2 lru) HOLDLOCK
WHEREd_w_id= @w_id
ANDd_id= @d_id
ANDw_id= d_w_id
ANDc_w_id= d_w_id
ANDc_d_id= d_id
ANDc_id= @c_id
FOR UPDATE OF d_next_o_id

begin

begin transaction NO

OPEN c_no_wdc
FETCH c_no_wdc INTO
@w_tax, @d_tax, @o_id,
@c_last, @c_discount, @c_credit
,@commit_flag, @ol_number
,@o_entry_d
UPDATEdistrict
SETd_next_o_id = @o_id + 1
WHERE CURRENT OF c_no_wdc
CLOSE c_no_wdc

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

 /* convert c_no_is cursor to a simple select */
 /* get item data (no one update item) */
 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
  /* Otherwise if the item is found */
     update stock
       set s_ytd = s_ytd + @ol_qty,
           @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 = @s_w_id and
                     s_i_id = @i_id

  select @ol_amount = @ol_qty * @i_price

  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)

        /* send line-item to client */
        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 /* while */

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_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)

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

select/* Return to client */
@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_idsmallint,@c_w_idsmallint,
@h_amount float,
@d_idtinyint,@c_d_idtinyint,
@c_idint
as
declare@c_lastchar(16)

declare@w_street_1char(20),@w_street_2char(20),
@w_citychar(20),@w_statechar(2),
@w_zipchar(9),@w_namechar(10),
@w_ytd float

declare@d_street_1char(20),@d_street_2char(20),
@d_citychar(20),@d_statechar(2),
@d_zipchar(9),@d_namechar(10),
@d_ytd float

declare@c_firstchar(16),@c_middlechar(2),
@c_street_1char(20),@c_street_2char(20),
@c_citychar(20),@c_statechar(2),
@c_zipchar(9),@c_phonechar(16),
@c_sincedatetime,@c_creditchar(2),
@c_credit_limnumeric(12,0),@c_balancefloat,
```

```
@c_discountreal,
@data1char(250),@data2char(250),
@c_data_1char(250),@c_data_2char(250)

declare @screen_datachar(200),@today datetime


declare @w_id_new smallint

declare c_pay_wd CURSOR FOR
SELECTw_id, w_street_1, w_street_2, w_city,
w_state, w_zip, w_name, w_ytd,
d_street_1, d_street_2, d_city,
d_state, d_zip, d_name, d_ytd
FROMdistrict HOLDLOCK,
warehouse HOLDLOCK
WHEREd_w_id= @w_id
ANDd_id= @d_id
ANDw_id= d_w_id
FOR UPDATE OF w_ytd, d_ytd

BEGIN TRANSACTION PID
OPEN c_pay_wd
FETCH c_pay_wd INTO
@w_id_new, @w_street_1, @w_street_2, @w_city,
@w_state, @w_zip, @w_name, @w_ytd,
@d_street_1, @d_street_2, @d_city,
@d_state, @d_zip, @d_name, @d_ytd
UPDATE district
SET     d_ytd  = @d_ytd + @h_amount
WHERE CURRENT OF c_pay_wd
UPDATE warehouse
SET w_ytd = @w_ytd + @h_amount
WHERE CURRENT OF c_pay_wd
CLOSE c_pay_wd

/* Customer data */
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 /* if */

/* Create the history record */
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_new,
@today, @h_amount, (@w_name + "    " + @d_name))

COMMIT TRANSACTION PID

select/* Return to client */
@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_idsmallint,@c_w_idsmallint,
@h_amount float,
@d_idtinyint,@c_d_idtinyint,
@c_lastchar(16)
as
declare@nint,@c_idint

declare@w_street_1char(20),@w_street_2char(20),
@w_citychar(20),@w_statechar(2),
@w_zipchar(9),@w_namechar(10),
@w_ytd float

declare@d_street_1char(20),@d_street_2char(20),
@d_citychar(20),@d_statechar(2),
@d_zipchar(9),@d_namechar(10),
@d_ytd float

declare@c_firstchar(16),@c_middlechar(2),
@c_street_1char(20),@c_street_2char(20),
@c_citychar(20),@c_statechar(2),
@c_zipchar(9),@c_phonechar(16),
@c_sincedatetime,@c_creditchar(2),
@c_credit_limnumeric(12,0),@c_balancefloat,
@c_discountreal,
@data1char(250),@data2char(250),
@c_data_1char(250),@c_data_2char(250)

declare @screen_datachar(200),@today datetime

declare @w_id_new smallint

declare c_pay_wd CURSOR FOR
SELECTw_id, w_street_1, w_street_2, w_city,
w_state, w_zip, w_name, w_ytd,
d_street_1, d_street_2, d_city,
d_state, d_zip, d_name, d_ytd
FROMdistrict HOLDLOCK,
warehouse HOLDLOCK
WHEREd_w_id= @w_id
ANDd_id= @d_id
ANDw_id= d_w_id
FOR UPDATE OF w_ytd, d_ytd

declare c_find CURSOR FOR
SELECT  c_id
FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
WHERE   c_w_id  =       @c_w_id
AND     c_d_id  =       @c_d_id
AND     c_last  =       @c_last
ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
FOR READ ONLY

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
OPEN c_find
while (@n>0) begin
FETCH c_find INTO @c_id
SELECT @n = @n-1
end
CLOSE c_find

OPEN c_pay_wd
FETCH c_pay_wd INTO
@w_id_new, @w_street_1, @w_street_2, @w_city,
@w_state, @w_zip, @w_name, @w_ytd,
@d_street_1, @d_street_2, @d_city,
@d_state, @d_zip, @d_name, @d_ytd
UPDATE district
SET     d_ytd  = @d_ytd + @h_amount
WHERE CURRENT OF c_pay_wd
UPDATE warehouse
SET w_ytd = @w_ytd + @h_amount
WHERE CURRENT OF c_pay_wd
CLOSE c_pay_wd

/* Customer data */
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 /* if */

/* Create the history record */
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_new,
@today, @h_amount, (@w_name + "    " + @d_name))

COMMIT TRANSACTION PNM

select/* Return to client */
@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_idsmallint,
@d_idtinyint,
@c_idint
as

DECLARE@o_idint,
@o_entry_ddatetime,
@o_carrier_idsmallint

BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
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
ANDo_d_id= @d_id
ANDo_c_id= @c_id
/* ORDER BY o_w_id, o_d_id, o_id */

/* Select order lines for the current order */
select/* Return multiple rows to client */
ol_supply_w_id,
ol_i_id,
ol_quantity,
ol_amount,
ol_delivery_d
FROMorder_line HOLDLOCK
WHEREol_o_id = @o_id
ANDol_d_id = @d_id
ANDol_w_id = @w_id

select/* Return single row to client */
@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
ANDc_d_id  = @d_id
ANDc_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_idsmallint,
@d_idtinyint,
@c_lastchar(16)
as

DECLARE@o_idint,
@o_entry_ddatetime,
@o_carrier_idsmallint

declare@nint,@c_idint
declare c_find CURSOR FOR
SELECT  c_id
FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
WHERE   c_w_id  =       @w_id
AND     c_d_id  =       @d_id
AND     c_last  =       @c_last
ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
FOR READ ONLY

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
OPEN c_find
while (@n>0) begin
FETCH c_find INTO @c_id
SELECT @n = @n-1
end
CLOSE c_find

/* Get the latest order made by the customer */
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
ANDo_d_id= @d_id
ANDo_c_id= @c_id
/* ORDER BY o_w_id, o_d_id, o_id */

/* Select order lines for the current order */
select/* Return multiple rows to client */
ol_supply_w_id,
ol_i_id,
ol_quantity,
ol_amount,
ol_delivery_d
FROMorder_line HOLDLOCK
WHEREol_o_id = @o_id
ANDol_d_id = @d_id
ANDol_w_id = @w_id

select/* Return single row to client */
@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
ANDc_d_id  = @d_id
ANDc_w_id  = @w_id

COMMIT TRANSACTION OSNM
go
if exists (select * from sysobjects where name = 'delivery')
drop proc delivery
go
CREATE PROC delivery
@w_idsmallint,
@o_carrier_idsmallint,
@d_idtinyint = 1
as

declare@no_o_id int,@o_c_idsmallint,
@ol_total float,@ol_amount float,
@junk_idsmallint

declare c_del_no CURSOR FOR
SELECTno_o_id
FROMnew_order (index no_clu) HOLDLOCK
WHEREno_d_id = @d_id
ANDno_w_id = @w_id
FOR UPDATE
/*
** The only purpose of the index hint in the above is to ensure
** that the clustered index is used.  As it turns out, our optimizer
** chooses the clustered index anyway -- with or without the hint.
*/

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
select @d_id = @d_id + 1
continue
end

DELETEFROM new_order
WHERECURRENT OF c_del_no
CLOSEc_del_no

/* Using the 'update' enhancement */
UPDATEorders
SETo_carrier_id = @o_carrier_id,
@o_c_id     = o_c_id,
@ol_total = 0.0
        WHEREo_id= @no_o_id
        ANDo_d_id= @d_id
        ANDo_w_id= @w_id

UPDATEorder_line
SETol_delivery_d = getdate(),
@ol_total = ol_amount + @ol_total
WHEREol_o_id= @no_o_id
ANDol_d_id= @d_id
ANDol_w_id= @w_id
UPDATEcustomer
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    /* Return to client */
@no_o_id
select @d_id = @d_id + 1
    end /* while @d_id... */
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
ANDd_id=@d_id
ANDol_w_id=@w_id
ANDol_d_id=@d_id
ANDol_o_idbetween (d_next_o_id - 20) and (d_next_o_id - 1)
ANDs_w_id=ol_w_id
ANDs_i_id=ol_i_id
AND     s_quantity < @threshold
go
EOF
```

tpcc_stats.sh

```
#!/bin/sh -f

isql -e -Usa -P << EOF
use tpcc
go


select getdate()
go
update statistics warehouse
go

select getdate()
go
update statistics district
go

select getdate()
go
update statistics item
go

select getdate()
go
update statistics stock
go

select getdate()
go

EOF

#select getdate()
#go
#update statistics customer
#go
#
#select getdate()
#go
#update statistics orders
#go
#
#select getdate()
#go
#update statistics new_order
#go
#
#select getdate()
#go
#update statistics order_line
#go
#
```

tpcc_tables.sh

```
#!/bin/sh -f
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)tpcc_tables.sh1.196/01/09SMI"
#

isql -Usa -P$PASSWORD << 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_idsmallint,
w_namechar(10),
w_street_1char(20),
w_street_2char(20),
w_citychar(20),
w_statechar(2),
w_zipchar(9),
w_taxreal,
w_ytdfloat/*- Updated by PID, PNM */
) on Scache
go

if exists ( select name from sysobjects where name = 'district' )
drop table district
go
create table district (
d_idtinyint,
d_w_idsmallint,
d_namechar(10),
d_street_1char(20),
d_street_2char(20),
d_citychar(20),
d_statechar(2),
d_zipchar(9),
d_taxreal,
d_ytdfloat,/*- Updated by PID, PNM */
d_next_o_idint/*- Updated by NO */
) on Scache
go

if exists ( select name from sysobjects where name = 'customer' )
drop table customer
go
create table customer (
c_idint,
c_d_idtinyint,
c_w_idsmallint,
c_firstchar(16),
c_middlechar(2),
c_lastchar(16),
c_street_1char(20),
c_street_2char(20),
c_citychar(20),
c_statechar(2),
c_zipchar(9),
c_phonechar(16),
c_sincedatetime,
c_creditchar(2),
c_credit_limnumeric(12,2),
c_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_data1char(250),/*- Updated (?) by PNM, PID */
c_data2char(250)/*- Updated (?) by PNM, PID */
) on Scustomer
go
create unique clustered index c_clu
on customer(c_w_id, c_id, c_d_id)
on Scustomer
go

if exists ( select name from sysobjects where name = 'history' )
drop table history
go
create table history (
h_c_idint,
h_c_d_idtinyint,
h_c_w_idsmallint,
h_d_idtinyint,
h_w_idsmallint,
h_datedatetime,
h_amountfloat,
h_datachar(24)
```

```
) on Shistory
go
alter table history partition 8
go

if exists ( select name from sysobjects where name = 'new_order' )
drop table new_order
go
create table new_order (
no_o_idint,
no_d_idtinyint,
no_w_idsmallint,
) on 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

if exists ( select name from sysobjects where name = 'orders' )
drop table orders
go
create table orders (
o_idint,
o_c_idint,
o_d_idtinyint,
o_w_idsmallint,
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

if exists ( select name from sysobjects where name = 'order_line' )
drop table order_line
go
create table order_line (
ol_o_idint,
ol_d_idtinyint,
ol_w_idsmallint,
ol_numbertinyint,
ol_i_idint,
ol_supply_w_idsmallint,
ol_delivery_ddatetime,/*- Updated by D */
ol_quantitysmallint,
ol_amountfloat,
ol_dist_infochar(24)
) on Sorder_line
go
create unique clustered index ol_clu
on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
on Sorder_line
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go

if exists ( select name from sysobjects where name = 'item' )
drop table item
go
create table item (
i_idint,
i_im_idint,
i_namechar(24),
i_pricefloat,
i_datachar(50)
) on Scache
go
create unique clustered index i_clu
on item(i_id)
on Scache
go
dbcc tune(indextrips, 10, item)
go

if exists ( select name from sysobjects where name = 'stock' )
drop table stock
go
create table stock (
s_i_idint,
s_w_idsmallint,
s_quantitysmallint,/*- Updated by NO */
s_ytdint,/*- 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_datachar(50)
) 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
```

Appendix C. Tunable Parameters

This Appendix contains the configuration information for the operating system, the RDBMS and Tuxedo.

Operating System Configuration Values

The Solaris kernel configuration parameters set in the file /etc/system are given below.

Solaris 7 Configuration File for the Sun Starfire Enterprise 10000 server

```
set msgsys:msginfo_msgmax=1048576
set msgsys:msginfo_msgmnb=4194304
set msgsys:msginfo_msgmni=4400
set msgsys:msginfo_msgtql=32768
set msgsys:msginfo_msgseg=32767
set msgsys:msginfo_msgssz=128
set msgsys:msginfo_msgmap=3002
set shmsys:shminfo_shmseg=200
set semsys:seminfo_semmap=100
set semsys:seminfo_semmni=8000
set semsys:seminfo_semmns=8000
set semsys:seminfo_semmnu=8000
set semsys:seminfo_semmsl=512
set semsys:seminfo_semume=100

set shmsys:shminfo_shmmax=0xffffffffffffffff
set pln:pln_enable_detach_suspend=1
set soc:soc_enable_detach_suspend=1

set autofs:autofs_unmount_thread_timer=360000
set pt_cnt=4096
set maxpgio=131072
set maxphys=4194304
set autoup=900
set bufhwm=8000
set segspt_minfree=16000

set priority_paging=1

* vxvm_START (do not remove)
forceload: drv/atf
forceload: drv/pln
forceload: drv/ses
forceload: drv/vxio
forceload: drv/vxspec
set vxio:vol_maxio=8192
set vxio:vol_maxkiocount=8192
set vxio:voliomem_chunk_size=1048576
set vxio:voliomem_kvmap_size=15728640
set vxio:voliomem_maxioctl=131072
set vxio:voliomem_maxspecialio=10240
set vxio:voliomem_maxpool_sz=134217728
* vxvm_END (do not remove)
set rlim_fd_max=10240
set rlim_fd_cur = 256
set report_ce_console=1

set cachefree=512
set lotsfree=256
set desfree=128
set minfree=64

set rechoose_interval=300
set disable_memscrub=1

* BEGIN RAID Manager addition
* DO NOT EDIT from BEGIN above to END below...
forceload: drv/ssd
forceload: drv/rdnexus
forceload: drv/rdriver
* END RAID Manager addition
```

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 = 100
global cache partition number  = 4

[Named Cache:c_customer]
cache size = 64M
cache status = mixed cache
cache replacement policy = DEFAULT
local cache partition number = 32

[2K I/O Buffer Pool]
pool size = 64M
wash size = 512 K
local async prefetch limit = 10

[Named Cache:c_history]
cache size = 32M
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 = 32M
wash size = 512 K
local async prefetch limit = 0

[Named Cache:c_index]
cache size = 2000M
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 = 2000M
wash size = 512 K
local async prefetch limit = 0

[Named Cache:c_index_nc]
cache size = 3200M
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 = 3200M
wash size = 512 K
local async prefetch limit = 0

[Named Cache:c_log]
cache size = 100M
cache status = mixed cache
cache status = HK ignore cache
cache replacement policy = DEFAULT
local cache partition number = 1

[8K I/O Buffer Pool]
pool size = 90M
wash size = 50M
local async prefetch limit = DEFAULT

[Named Cache:c_new_order]
cache size = 500 M
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 = 500 M
```

```
wash size = 20 M
local async prefetch limit = 0

[Named Cache:c_order_line]
cache size = 2000M
cache status = mixed cache
cache status = HK ignore cache
cache replacement policy = DEFAULT
local cache partition number = 32

[2K I/O Buffer Pool]
pool size = 2000M
wash size = DEFAULT
local async prefetch limit = 0

[Named Cache:c_orders]
cache size = 2400M
cache status = mixed cache
cache status = HK ignore cache
cache replacement policy = DEFAULT
local cache partition number = 32

[2K I/O Buffer Pool]
pool size = 1200M
wash size = DEFAULT
local async prefetch limit = 10

[16K I/O Buffer Pool]
pool size = 1200M
wash size = DEFAULT
local async prefetch limit = 10

[Named Cache:c_stock]
cache size = 40000 M
cache status = mixed cache
cache replacement policy = DEFAULT
local cache partition number = 64

[2K I/O Buffer Pool]
pool size = 40000 M
wash size = 4G
local async prefetch limit = 0

[Named Cache:c_stock_index]
cache size = 4500 M
cache status = mixed cache
cache status = HK ignore cache
cache replacement policy = relaxed LRU replacement
local cache partition number = 64

[2K I/O Buffer Pool]
pool size = 4500 M
wash size = DEFAULT
local async prefetch limit = 0

[Named Cache:c_tinyhot]
cache size = 400M
cache status = mixed cache
cache status = HK ignore cache
cache replacement policy = relaxed LRU replacement
local cache partition number = 32

[2K I/O Buffer Pool]
pool size = 400M
wash size = 256 K
local async prefetch limit = 0

[Named Cache:default data cache]
cache size = 3M
cache status = default data cache
cache status = HK ignore cache
cache replacement policy = DEFAULT
local cache partition number = 2

[2K I/O Buffer Pool]
pool size = 3M
wash size = DEFAULT
local async prefetch limit = 0

[Meta-Data Caches]
number of open databases = 5
number of open objects = DEFAULT
open object spinlock ratio = DEFAULT
number of open indexes = DEFAULT
open index hash spinlock ratio = DEFAULT
open index spinlock ratio = DEFAULT

[Disk I/O]
allow sql server async i/o = DEFAULT
disk i/o structures = 49152
page utilization percent = DEFAULT
number of devices = 240
disable disk mirroring = 1
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 = 1500 #1200
number of remote logins = DEFAULT
number of remote sites = DEFAULT
max number network listeners = DEFAULT
tcp no delay = DEFAULT
allow sendmsg = DEFAULT
syb_sendmsg port number = DEFAULT

[O/S Resources]
max async i/os per engine = 10000
max async i/os per server = 10000

[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 = 31850496  # 60.75GB
additional network memory = 18432000  # 4096*3*1500
lock shared memory = DEFAULT
shared memory starting address = DEFAULT
max SQL text monitored = DEFAULT

[Processors]
max online engines = 64
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 = DEFAULT
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 = 100
number of pre-allocated extents = DEFAULT
event buffers per engine = 3
cpu accounting flush interval = DEFAULT
i/o accounting flush interval = DEFAULT
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 = DEFAULT
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
enable sort-merge join and JTC = DEFAULT
abstract plan load = DEFAULT
abstract plan dump = DEFAULT
abstract plan replace = DEFAULT
abstract plan cache = DEFAULT
text prefetch size = DEFAULT

[User Environment]
number of user connections = 1500 #1200
stack size = 102400
stack guard size = DEFAULT
permission cache entries = DEFAULT
user log cache size = 4096
user log cache spinlock ratio = DEFAULT
enable HA = DEFAULT
enable DTM = DEFAULT
allow remote access = DEFAULT
lock shared memory = DEFAULT
allow sql server async i/o = DEFAULT

[Lock Manager]
number of locks = 30000
deadlock checking period = 3000
freelock transfer block size = DEFAULT
max engine freelocks = DEFAULT
lock spinlock ratio = 10
lock address spinlock ratio = 1
lock table spinlock ratio = 1
lock hashtable size = DEFAULT
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
check password for digit = DEFAULT
minimum password length = DEFAULT
maximum failed logins = DEFAULT
unified login required = DEFAULT
use security services = DEFAULT
msg confidentiality reqd = DEFAULT
msg integrity reqd = DEFAULT
dtm detach timeout period = DEFAULT
secure default login = 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

[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

[Java Services]
enable java = DEFAULT
size of process object heap = DEFAULT
size of shared class heap = DEFAULT
size of global fixed heap = DEFAULT

[DTM Administration]
enable xact coordination = 0
xact coordination interval = DEFAULT
number of dtx participants = DEFAULT
strict dtm enforcement = DEFAULT
txn to pss ratio = DEFAULT
dtm lock timeout period = DEFAULT

[Diagnostics]
dump on conditions = DEFAULT
maximum dump conditions = DEFAULT
number of ccbs = DEFAULT
caps per ccb = DEFAULT
average cap size = DEFAULT

[Monitoring]
Q diagnostics active = DEFAULT
sql text pipe active = DEFAULT
sql text pipe max messages = DEFAULT
plan text pipe active = DEFAULT
plan text pipe max messages = DEFAULT
statement pipe active = DEFAULT
statement pipe max messages = DEFAULT
errorlog pipe active = DEFAULT
errorlog pipe max messages = DEFAULT
deadlock pipe active = DEFAULT
deadlock pipe max messages = DEFAULT
wait event timing = DEFAULT
process wait events = DEFAULT
object lockwait timing = DEFAULT
SQL batch capture = DEFAULT
statement statistics active = DEFAULT
per object statistics active = DEFAULT
```

Tuxedo Configuration values

```
*RESOURCES
IPCKEY          40001# IPC KEY from 32,768 to 16,777,215
MASTER          c3# machine on which master copy is found
UID             30# user id as displayed by command "id"
GID             30# 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      200# maximum number of servers
MAXSERVICES     200# maximum number of services
MAXCONV         1
MODEL           SHM# SHM=single processor, MP=multi processor
LDBAL           N# load balancing, Y=yes, N=no
CMTRET          COMPLETE
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
"c3"LMID="c3"
TUXCONFIG="/export/home/dbbench/tuxedo/tuxconfig"
ROOTDIR="/export/home/tuxedo"
APPDIR="/export/home/dbbench/tuxedo"
ULOGPFX="/export/home/dbbench/tuxedo/ULOG"
ENVFILE="/export/home/dbbench/tuxedo/tuxedo.env"

*GROUPS
"group1"LMID="c3"GRPNO=1

*SERVERS

"tpcc_srv_newordpay"SRVGRP="group1" SRVID=1 CLOPT="-A" RQADDR="newopayq1"REPLYQ=Y
MIN=24 RESTART=N

"tpcc_srv_stockdel"SRVGRP="group1" SRVID=40 CLOPT="-A" RQADDR="stockdelq1" REPLYQ=Y
MIN=6 RESTART=N

*SERVICES
"DEL"
"NEWO"
"ORDS"
"PAYM"
"STOCK"

*ROUTING
```

Compilation flags

```
GENRAW = $(HOME)/generic/TPCC/c/tuxraw
TPCCSRC = $(HOME)/generic/TPCC/c
GENSRC = $(HOME)/generic/c
ROOTDIR = /export/home/tuxedo
CFLAGS= -O -I. -I$(GENRAW) -I$(TPCCSRC) -I$(GENSRC) -I$(ROOTDIR)/include \
        -D_TMSTHREADS -DTELNET -DPTY -DWEIGHTED_DISTRIBUTION
SYB_LIBDIR  =   $(SYBASE)/lib
SYB_LDFLAGS =   -L$(SYB_LIBDIR) -lsybdb -lm -lc -lnsl
```

Appendix D. Disk Storage

This Appendix contains the 180-day space calculations.

The calculations used to determine the storage requirements for the 8 hours logical log and 180-day space calculations are contained in this appendix.

The calculations for the 8 hours recovery log was determined as follows:

The number of the log pages used during the measurement run was determined by using the Sybase stored procedure sp_helpsegment logsegment before and after the run. We found the ammount of log spaceused by the DBMS during the benchmark run was 22218152 pages (44436304 KB). The total new-order transactions were done during the measurement was 11613458. The amount of log disk used per new-order transaction be 3.826 KB, therefore the 8 hours log space needed is 274.75GB.

See the spread sheet for more information.

Note: Numbers are in Kbytes unless otherwise specified						
# of warehouse	12880	TPM-C	156873.03	tpm/w	12.17958307	
Table Name	Rows	Data	Index	5% Space	8Hr. Space	Total Space
Warehouse	12,880	25,760.00	92.00	1,292.60		27,144.60
District	165,000	330,000.00	1,324.00	16,566.20		347,890.20
Item	100,000	9,524.00	86.00	192.20		9,802.20
New-Order	148,500,000	1,622,952.00	19,552.00		259,240.11	1,901,744.11
History	495,000,000	27,723,072.00	0.00		4,375,595.00	32,098,667.00
Orders	495,000,000	13,378,380.00	161,186.00		2,136,980.25	15,676,546.25
Customer	495,000,000	330,000,000.00	25,617,092.00	7,112,341.84		362,729,433.84
Order-Line	4,950,000,000	300,000,000.00	3,947,366.00		47,972,698.48	351,920,064.48
Stock	1,650,000,000	550,200,022.00	3,039,784.00	11,064,796.12		564,304,602.12
Total	8,233,777,880	1,223,289,710	32,786,482	18,195,188.96	54,744,513.83	1,329,015,894.79
Segment Name	# of devices	Segment Size	Needed Space	Overhead		Not Needed
Scache	4	3,072,000.00	2,309,446.92	23,094.47		739,458.61
Shistory	7	34,048,000.00	32,419,653.67	324,196.54		1,304,149.80
Sorders	8	17,203,200.00	15,833,311.71	158,333.12		1,211,555.17
Scustomer	72	390,758,400.00	366,356,728.18	3,663,567.28		20,738,104.54
Sorder_line	40	374,784,000.00	355,439,265.12	3,554,392.65		15,790,342.23
Sstock	100	732,160,000.00	569,947,648.14	5,699,476.48		156,512,875.38
Total	231.00	1,552,025,600.00	1,342,306,053.74	13,423,060.54		196,296,485.72
Dynamic Space	330,527,306.99		Sum of Data for Orders, Order-Line and History (excluding free extants)			
Static Space	957,167,134.51		Data + Index + 5% Space + Overhead - Dynamic Space			
Free Space	68,034,672.78		Total Seg. Size - Dynamic Space - Static Space - Not Needed			
Daily Growth	64,410,956.70		(Dynamic Space / W * 62.5) * TPM-C			
Daily Spread	0.00		Free Space - 1.5 * Daily Growth (0 if negative)			
180 Day	12,551,139,340.83		Static Space + 180 * (Daily Growth + Daily Spread)			
180 Day (GB)	11,969.70	(Excludes OS, Paging and RDBMS Logs)				
Log/No trans	3.83		ew-Order transaction			
8 hrs log (GB)	274.75	(not mirrored)				
8 hrs log (GB)	549.50	(mirrored)				
Disk name	Size(MB)	# of disks	Total MB	Total GB		
18GB	17,267.00	792.00	13,675,464.00	13,354.95		
9x18GB(RAID0)	153,648.00	8.00	1,229,184.00	1,200.38		

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

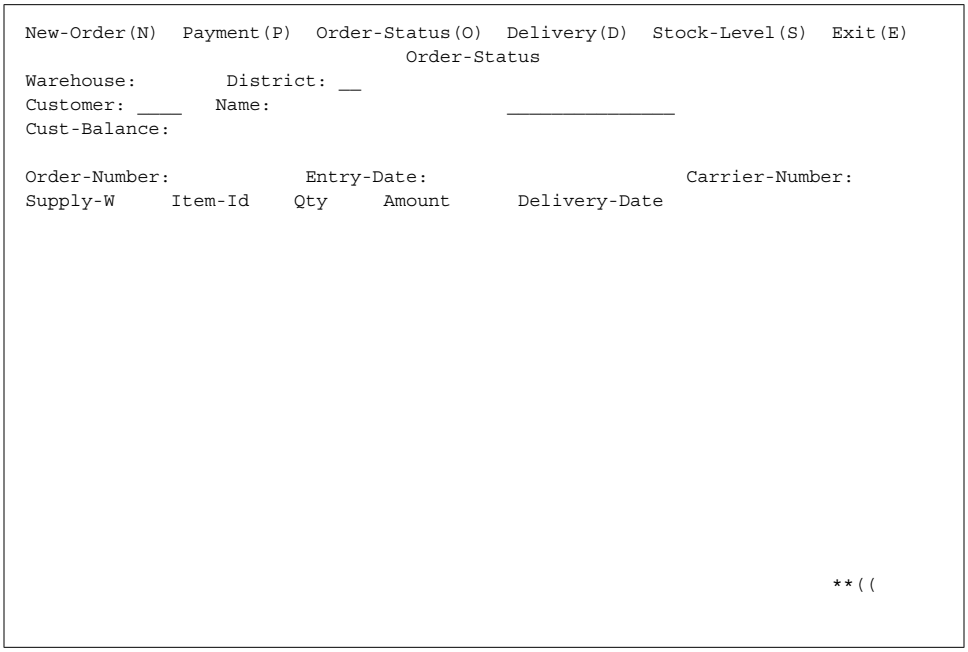
```
/* run for ramp up without capturing the stats */
i=0;
in_ramp = 1;
while (1)
{
tx_type = do_menu();/* Select transaction */
switch (tx_type) {
case NEWORDER:
do_neworder();
break;
case PAYMENT:
do_payment();
break;
case DELIVERY:
do_delivery();
break;
case ORDSTAT:
do_ordstat();
break;
case STOCKLEVEL:
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 */
}
}
```

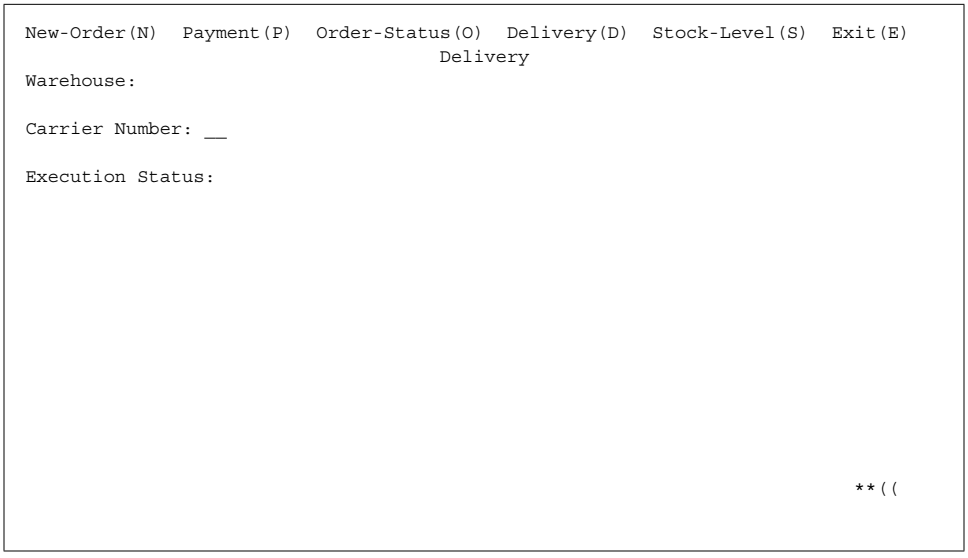
Appendix F. Screen Layout

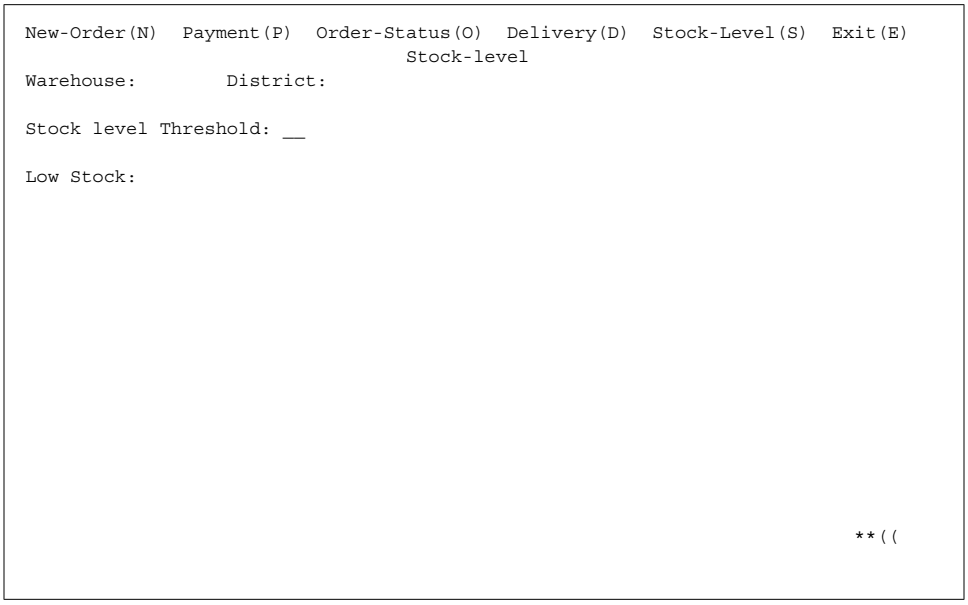
This Appendix contains the screen form layouts for the 5 transactions.

```
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

The following pages contain the pricing quotes for the hardware and software included in this FDR.

August 21, 2000

Mr. Daryl Madura
Strategic Applications Engineering
Sun Microsystems
503-520-7667
503-520-7724 Fax

Dear Mr. Madura:

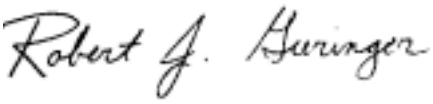
Per your request I am enclosing the pricing information regarding TUXEDO that you requested. This pricing applies to Tuxedo 6.1, 6.2, 6.3, 6.4,6.5, and 7.1. Please note that Tuxedo 7.1 is our most recent version of Tuxedo but that all 6.x releases are generally available. Core functionality services pricing is appropriate for your activities. As per the table below Sun Ultra Sparc systems are classified as either a Tier 1, 2, 3, 4 or 5 systems depending on the performance and CPU capacity of the system. This quote is valid for 90 days from the date of this letter.

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

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five-tier computer classification as TUXEDO 6.x and 7.1. Prices range from $3,000 for Tier 1 to $250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

Very Truly Yours,



Robert J. Gieringer
Worldwide Pricing Manager

BEA Tux/CFS Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 8) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers	Unlimited	$3,000.00	$480.00	$690.00
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations	Unlimited	$12,000.00	$1,920.00	$2,760.00
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	Unlimited	$30,000.00	$4,800.00	$6,900.00
Tier 4 - Large (more than 8, less than 32 CPUs)	Unlimited	$100,000.00	$16,000.00	$23,000.00
Tier 5 - Massively Parallel Systems, > 32 processors	Unlimited	$250,000.00	$40,000.00	$57,500.00

	Tier 1	Tier 2	Tier 3	Tier 3	Tier 4	Tier 5
Platform						
SUN SPARC	X- term-inal 1 Station 5/85 Station 4 Station 20/50 Station 20/51 Station 20/61 Ultra 1 140/170 Server 470 Server 5/70 Server 20/50 Server 20/51 Server 20/61 Ultra 2 Desktop Ultra 5, 5S Ultra 10, 10S Enterprise 1 &150	Server 5/85 Station 20/71 Server 20/71 Enterprise 220R Enterprise 250 Enterprise 2 - 2100,2200 Ultra 60	Station 20/502 MP 20/612 MP 20/514 MP 20/HS11 20/712 MP Server 1000 Server 1000E Server 20/502 Server 20/712 Server 20/612 Server 20/514 Enterprise 2 -2300 Enterprise 420R	Enterprise 450 SparcCenter 1000 Enterprise 3000 & 3500 Enterprise 4000, 4500, 5000 & 5500 < 8 proc.	SparcCenter 2000 SparcCenter 2000E Enterprise 4000, 4500, 5000 & 5500 Between 8 and 32 proc. Enterprise 6000 & 6500 Between 8 and 32 proc. CRS6400 (< 32 proc.)	CRS6400 (≥32 proc.) Enterprise 6000 & 6500 (≥32 proc.) Enterprise 10000

SALES QUOTE

August 23, 2000

PO #	Quote #
	3385

131 Albright Way
Los Gatos, CA 95032-1801

Vinny Nguyen
Phone: 408-341-1743
Fax: 408-341-1696

Account Manager
Vinny Nguyen

Quote To:

Daryl Madura
Sun Micro Systems
8305 S.W. Creekside Place
Beaverton, OR 97008

Phone: (503) 520-7667
Fax: (503) 520-7724

Ship To:

Daryl Madura
Sun Micro Systems
8305 S.W. Creekside Place
Beaverton, OR 97008

Phone: (503) 520-7667
Fax: (503) 520-7724

FOB: Terms:

Here is the quote you requested. Quote Valid for 60 days

Item	Mfr.	Part No.	Qty	Description	List Price	Unit Price	Ext. Price
1	SUN	E10000-4	1	Enterprise 10000 Server base cabinet, cage for 16 system boards, peripheral space w/power sequencer, Solaris license for unlimited users, license for SSP software		129,600.00	129,600.00
2	SUN	2761A	16	E10000 System Board		39,600.00	633,600.00
3	SUN	2580A	64	400-MHz UltraSPARC module with 8-MB external cache		10,800.00	691,200.00
4	SUN	7025A	16	E10000 memory board, no SIMMs		7,200.00	115,200.00
5	SUN	7023A	64	1-GB memory expansion (8x128-MB SIMMs)		6,120.00	391,680.00
6	SUN	2730A	16	E10000 dual SBus I/O card		5,400.00	86,400.00
7	SUN	2754A	1	Sun Enterprise 10000 System service Processor, Ultra 5 Workstation, 128 MB of memory, 8.4-GB internal disk, 19-inch color monitor, 32X CD-ROM, and Sun Quad FastEthernet PCI adapter		7,560.00	7,560.00
8	SUN	3875A	4	E10000 AC Breaker/Filter assembly		2,160.00	8,640.00
9	SUN	9685A	8	E10000 48 volt power supply		2,160.00	17,280.00
10	SUN	9681A	1	E10000 Power Control Module, includes cables		720.00	720.00
11	SUN	9671A	16	E10000 fan Tray		1,080.00	17,280.00

Item	Mfr.	Part No.	Qty	Description	List Price	Unit Price	Ext. Price
12	SUN	2722A	1	Sun Enterprise 10000 control board with rack-mounted Ethernet hub		10,800.00	10,800.00
13	SUN	3865A	4	Sun Enterprise 10000 power cord, 14 feet (U.S. version)			
14	SUN	SG-ARY154A-72GR5	1	72-GB Sun StorEdge D1000 rack mount (4 x 18.2-GB / 10000 rpm)		8,406.05	8,406.05
15	SUN	6730A	26	FC-100 FC-AL SBus host adapter with one GBIC		1,944.00	50,544.00
16	SUN	1049A	3	Sun Quad FastEthernet with qfe naming		1,436.40	4,309.20
17	SUN	1059A	1	SunFastEthernet 10/100 SBus Adapter 2.0, Network SBus Cards		572.40	572.40
18	SUN	979A	1	12M diff SCSI EXP cable		265.00	265.00
19	Sun	SG-ARY543A-2400G	6	2400-Gbyte Sun StorEdge A5200 Cabinet with six 400-GByte arrays (22 18.2-GByte low profile 10,000 rpm drives, 4 FC-AL hubs w/ 4 GBICs each, four 15 meter fiber optic cables mounted in 72-inch Sun StorEdge expansion cabinet. (Factory Configured for E10000)		376,909.20	2,261,455.20
20	Sun	XT3ES-RK-88-1310	1	1310GB Sun StorEdge T3ES, includes 8 x T3 arrays configured in 4 partner groups, 1144-GB usable RAID 5 storage preconfigured as eight RAID 5 LUN's (8+1), 72 x 18.2GB 10K RPM FC-AL drives, 8 copper-to-optic media interface adapter, 8x15 meter fibre optic cables, 8 Unit Interconnect cables, installed in one 72 in StorEdge Expansion cabinet, fans and door included, installation and configuration 2 year Sun Spectrum Gold included, SRS Ready		263,376.00	263,376.00
21	SUN	X1065A	1	SBus Ultra differential Fast/Wide intelligent SCSI-2 host adapter (UDWIS/S)		932.40	932.40
22	Sun	978A	24	15-meter Fibre Optic Cable		175.00	4,200.00
23	SUN	6283A	1	12-24GB 4MM DDS3 INT/E3,4,5,6		972.00	972.00
24	Sun	SCMMS-210-9P99	1	Sun StorEdge Component Manager V 2.1 Software Media, Docs			
25	Sun	VVMGS-304-9999	1	Veritas Volume Manager 3.0.4 for A5x00s - Media, Documentation & RTU			
26	SUN	E10000-SW-SSP-3.1	1	Starfire System Service Processor 3.1			
27	Sun	3508A	1	North America/universal/Canadian Type 6 Country Kits with Sun I/O for E10000			
28	SUN	3858A	12	U.S. POWER CORD FOR STOREDGE			
29	SUN	3800A	1	Power Cord for Data Center Cabinet, U.S. Version			

Item	Mfr.	Part No.	Qty	Description	List Price	Unit Price	Ext. Price
30				SubTotal		878,188.45	4,704,992.25
31	SUN	A22UKC1A9P-C512CI	46	Ultra 10 Model 440, 440 MHz, 2mb Ecache, Onboard PGX24 Graphics, 512mb DRAM, 9gb/7200rpm hard disk drive, 32X CD-ROM, 1.44mb floppy, Solaris 7 3/99 preinstalled, North American UNIX		4,356.55	200,401.30
32	SUN	X7039A	46	OPT 512MB DRAM, 50NS, U10 only		1,597.55	73,487.30
33	SUN	1034A	46	QUAD FASTETHERNET CONTROLLER PCI ADAPTER		1,292.40	59,450.40
34	SUN	X7143A	46	17-Inch color Monitor, 15.7-Inch diagonal viewable, .28 dot pitch, 2-meter non detachable video, signal cable, DDC1/2B, MPR-II, TCO 95, CE Mark, Energy Star Compliant, 80W power consumption, 1152x900-66Hz, 76Hz, 1024x768-60Hz		289.25	13,305.50
35				SubTotal		7,535.75	346,644.50

Subtotal		$5,051,636.75
Sales Tax @	8.25%	
Total		**$5,051,636.75**

Please contact me if I can be of further assistance. Discount is Based on total size of deal.

ACCEPTED BY- X Date:

Purchase Order Number:
#

Remit To Address:
CAT TECHNOLOGY
PO Box 45124
San Francisco, CA 94145

Software House International Pricing Proposal	Quotation #MD-200823-57096 08/23/2000

SHI Account Exec: Matthew Martin

Sun Microsystems
Daryl Madura
Quote good for 90 days.

Phone: 503-520-7667
Fax: 503-520-7724

Telephone : 408-922-1106
Fax : 408-526-1222

Reference:

Description	Part #	Qty	List$	Customer$	Extended
8port+1 Hub 5yr Ret. to Man War.	Z222146	17710	$29.00	$23.00	$407,330.00
8port+1 10/100 Hub 5yr Rtn to Man War.	Z22147	3	$99.00	$79.00	$237.00
				Total	**$407,567.00**