digital™

TPC Benchmark C®
Full Disclosure Report

Digital AlphaServer™1000A 5/300 1 CPU
Client/Server System
Using
Sybase SQL Server 11.0®,
BEA TUXEDO®,
and
Digital UNIX® V3.2F-2

Company Name	System Name	Database Software	Operating System Software
Digital Equipment Corporation Sybase, Inc.	Digital AlphaServer 1000A 5/300 1 CPU	Sybase SQL Server 11.0 BEA TUXEDO System/T v4.2	Digital UNIX V3.2F-2

Availability Date: August, 1996

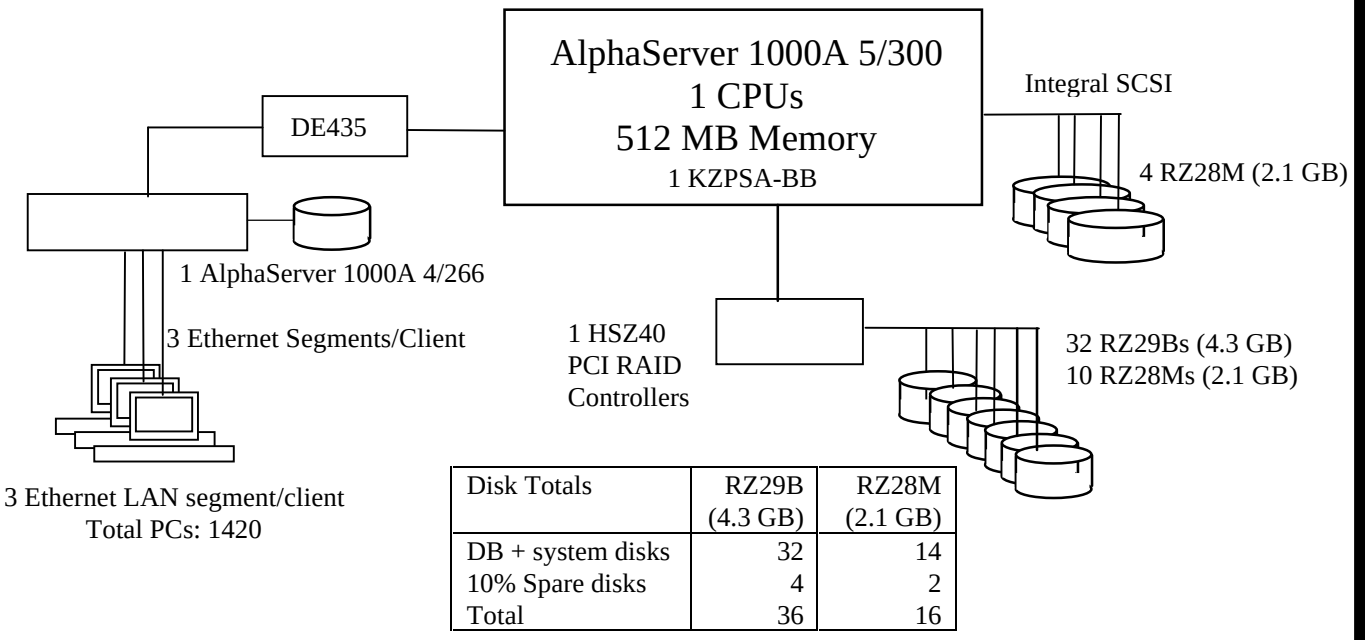
Total System Cost	TPC-C Throughput	Price Performance
- Hardware - Software - 5-Years Maintenance	Sustained maximum throughput of system running TPC Benchmark C expressed in transactions per minute	Total system cost/ TPC-C® throughput $158.24/1691.04
$267,586	**1691.04 tpmC®**	**$158.24/per tpmC**

First Printing July 1996

Digital Equipment Corporation Sybase, Inc.	**Digital AlphaServer 1000A 5/300 1 CPU C/S and 1 AlphaServer 1000A 4/266 Front-End System**	**TPC-C Rev 3.2** **Report Date: July 96**

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$267,586	**1691.04 tpmC**	**$158.24 /tpmC**	**August 1996**

Processors	Database Manager	Operating System	Other Software	Number of Users
1 300 MHz DECchip 21164	**Sybase SQL Server 11.0**	**Digital UNIX V3.2F-2**	**BEA TUXEDO System/T v4.2**	**1420**



Disk Totals	RZ29B (4.3 GB)	RZ28M (2.1 GB)
DB + system disks	32	14
10% Spare disks	4	2
Total	36	16

	Server		Client	
System Components	**Qty**	**Type**	**Qty**	**Type**
Database Nodes	1	AlphaServer 1000A 5/300	1	AlphaServer 1000A 4/266
Processors	1	300 MHz DECchip 21164	1	266 MHz DECchip 21064A
Cache Memory	2	MB		512 KB
Memory	1	512 MB	4	128 MB
Disk Controller	1	PCI		
Disks	32	4.3 GB Disks	2	2.1 GB Disks
	14	2.1 GB Disks		
Total Disk Storage	167.0	GB		

Digital Equipment Corporation	Digital AlphaServer 1000A 5/300	TPC-C REV 3.2 EXECUTIVE SUMMARY PAGE 2 OF 2
		Report Date: 29-July -1996

Description	Part Number	Third Party Brand	Pricing	Unit Price	Qty	Extended Price	5 yr. Maint. Price
Server Hardware							
Digital AlphaServer 1000A 5/300 UNIX SBB	PB77C-FA	Digital	1	9,505	1	9,505	0
AS1000A 5/300 128 MB Memory	PB7MA-CD	Digital	1	4,525	4	18,100	0
SW 6-CH SCSI 32mb R/CA 42 Dev	HSZ40-CF	Digital	1	10,620	1	10,620	0
SWxxx Cntrlr Basic Shelf	BA350-MA	Digital	1	1,085	1	1,085	0
Stor Shelf/Metric w/ps SCSI	BA350-JA	Digital	1	1,395	6	8,370	0
PCI-Host Bus Adapter (FWD)	KZPSA-BB	Digital	1	860	1	860	0
PCI Bus, 32 Bit, PC NIC	DE435-AA	Digital	1	140	1	140	0
8GB DAT w/snap-in carrier	TLZ07-VA	Digital	1	1,565	1	1,565	0
VT510, White, North Amer, No Key	VT510-AA	Digital	1	330	1	330	0
1.0M Cable, SCSI-2 "A"	BN21H-01	Digital	1	65	3	195	0
2.0M Cable, SCSI-2 "A"	BN21H-02	Digital	1	75	3	225	0
3.0M Cable, SCSI-3 "P: 1S/1R	BN21K-03	Digital	1	125	1	125	0
2.1 GB Disk w/snap-in carrier	RZ28M-VA	Digital	1	1,000	14	14,000	0
4.3 GB Disk/ w/snap-in carier	RZ29B-VA	Digital	1	1,725	32	55,200	0
					Subtotal	**120,320**	0
Client Hardware							
AlphaServer 1000A 4/266 Unix PKG	PB74B-FA	Digital	1	15,700	1	15,700	0
Dataram Memory 128mb Memory Module	DRD 70/128	BayState	2	3,850	4	15,400	0
2.1 Gbf 3.5" DK in Mod Carrier	RZ28M-VA	Digital	1	1,060	2	2,120	0
PCI Bus, 32 Bit, PC NIC	DE435-AA	Digital	1	140	4	560	0
VT510, White, North Amer, No Key	VT510-AA	Digital	1	330	1	330	0
					Subtotal	**34,110**	0
Client Software							
Digital UNIX Unld IA User License, Media & I	QL-MT7AE-AA	Digital	1	4,680	1	4,680	0
Digital UNIX Media & Doc CD	QA-MT4AA-H8	Digital	1	280	1	280	0
BEA Tuxedo R4.2.2	BEA TP Mo	BEA Systems	3	7,500	1	7,500	0
1YR BEA Warranty for Tuxedo	BEA TP Mo	BEA Systems	3	1,650	5	0	8,250
Sybase Open Client/DB Lib	Sybase	Sybase	4	795	1	795	0
Sybase SQL Server RDBMS Software	Sybase	Sybase	4	27,996	1	27,996	0
1YR Sybase Maintenance	Sybase	Sybase	4	6,999	5	0	34,995
					Subtotal	**41,251**	**43,245**
Warranties							
Extended Warranty 13-24 Months	FM-WRNTY-24	Bay State	2	890	1		890
Extended Warranty 13-36 Months	FM-WRNTY-36	Bay State	2	1,775	1		1,775
Extended Warranty 13-48 Months	FM-WRNTY-48	Bay State	2	2,725	1		2,725
Extended Warranty 13-60 Months	FM-WRNTY-60	Bay State	2	2,810	1		2,810
					Subtotal		**8,200**
User Connectivity							
Allied Tel 24 port HUB	DEH1487	MacWhse	5	310	66	20,460	
					Subtotal	20,460	0
					Total	**$216,141**	**$51,445**

Notes:

1=Digital 2=Bay State Comp Grp, 3=BEA Systems, 4= Sybase, 5=MacWarehouse

Audited by Information Paradigm, Inc.

Five-Year Cost of Ownership:	**$267,586**
tpmC Rating:	**1691.00**
$ / tpmC:	**158.24**

Abstract

This report documents the compliance of Digital Equipment Corporation's and Sybase, Inc.'s TPC Benchmark C tests on the AlphaServer 1000A 5/300 1 CPU client/server system with Version 3.2 of the TPC Benchmark C Standard Specification. One AlphaServer 1000A 4/266 computer was used as the front-end client. This system was also used as a development system.

The tests were performed using the BEA TUXEDO V4.2 transaction processing monitor, which is based on BEA TUXEDO System/T Transaction Processing Monitor Version 4.2, and Sybase SQL Server 11.0 relational database, running under Digital UNIX.

Two standard metrics, transactions-per-minute-C (tpmC) and price per tpmC ($/tpmC) are reported, in accordance with the TPC Benchmark C Standard. The independent auditor's report by Information Paradigm is appended at the end of this report.

Numeric Quantities Summary
AlphaServer 1000A 5/300 1 CPU C/S System

MQTH

MQTH, computed Maximum Qualified Throughput	1691.04 tpmC
% throughput difference, reported and reproducibility runs	1 %

Response Times (seconds)

Transaction	90th percentile	Average	Maximum
New-Order	1.7	1.1	11.1
Payment	1.3	0.8	11.3
Order-Status	1.5	1.0	6.8
Delivery (interactive)	0.3	0.3	0.4
Delivery (deferred)	3.0	1.0	6.0
Stock-Level	8.3	4.2	24.5
Menu	0.3	0.3	1.2

Transaction Mix, in percent of total transactions

Transaction	Total Occurrences	Percentage
New-Order	50970	44.29
Payment	49807	43.28
Order-Status	4773	4.15
Delivery	4669	4.06
Stock-Level	4862	4.22

Keying/Think Times (seconds)

Transaction	Min.	Average	Max.
New-Order	18.0/0.0	18.0/12.1	18.1/159.7
Payment	3.0/0.0	3.0/12.2	3.1/135.9
Order-Status	2.0/0.0	2.0/10.1	2.1/98.1
Delivery	2.0/0.0	2.0/5.1	2.1/45.4
Stock-Level	2.0/0.0	2.0/5.0	2.1/42.1

Emulation Delay (in seconds)

Transaction	Resp_Time	Menu
New order	None	N/A
Payment	None	N/A
Order status	None	N/A
Delivery (interactive)	None	N/A
Stock level	None	N/A

Test Duration

Ramp up time	30 min.
Measurement interval	30 min.
Number of checkpoints	1
Checkpoint interval	30 min.
Number of New order transactions	50970
Number of transactions (all types) completed in measurement interval	115081

Preface

This report documents the compliance of Digital Equipment Corporation's and Sybase, Inc., and TPC Benchmark C (TPC-C) testing on the AlphaServer 1000A 5/300 1 CPU client/server system with Version 3.2 of the *TPC Benchmark C Standard Specification*[1]. The TPC-C Standard represents an effort by Digital Equipment Corporation and other members of the Transaction Processing Performance Council (TPC) to create industry-wide benchmarks for evaluating the performance and price/performance of transaction processing systems.

These tests were run using the TUXEDO transaction processing monitor and Sybase SQL Server relational database under the Digital UNIX operating system. One AlphaServer 1000A 4/266 computer was used as the front-end client. This system was also used as a development system.

About the TPC-C Benchmark

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

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

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

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

---

[1] *TPC Benchmark C Standard Specification*, Transaction Processing Performance Council, Version 3.2.

Document Structure

This *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-C Standard and explains how each requirement is satisfied.
- Appendix A contains the source code of the TPC-C application code used to implement the TPC-C transactions.
- Appendix B contains the database definition and population code used in the tests.
- Appendix C contains the Remote Terminal Emulator (RTE) code used to generate and record transactions.
- Appendix D contains the tunable parameters.
- Appendix E contains the independent auditor's report on the compliance of this disclosure with the benchmark specifications.
- Appendix F contains third-party price quotations.

Additional Copies

To request additional copies of this report, please contact:

Administrator, TPC Benchmark Reports
Computer Systems Division Performance Group
Digital Equipment Corporation
110 Spit Brook Road (ZK02-3/M31)
Nashua, NH 03062
U.S.A.

FAX number: 603-881-6082

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 with each item listed in Clause 8 of the *TPC Benchmark C Standard Specification*.

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 Order and Titles

The order and titles of sections in the Test Sponsor's Full Disclosure Report must correspond with the order and titles for the TPC-C standard specification. The intent is to make it as easy as possible for readers to compare and contrast material in different Full Disclosure reports.

The order and titles of sections in this report correspond with that of the TPC-C standard specification.

1.2 Summary Statement

The TPC Executive Summary Statement must be included near the beginning of the Full Disclosure report.

The TPC Executive Summary Statement is included at the beginning of this report.

1.3 Numerical Quantities Summary

The numerical quantities listed below must be summarized near the beginning of the Full Disclosure Report.

- *measurement interval in minutes,*
- *number of checkpoints in the measurement interval,*
- *computed maximum Qualified Throughput in tpmC,*
- *percentage difference between reported throughput and throughput obtained in reproducibility run,*
- *ninetieth percentile, average, and maximum response times for the New-Order, Payment, Order-Status, Stock-Level, Delivery (deferred and interactive) and Menu transactions,*
- *time in seconds added to response time to compensate for delays associated with emulated components, and*
- *percentage of transaction mix for each transaction type.*

These numerical quantities are summarized at the beginning of this report.

1.4 Application Program

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

Appendix A contains the DEC C application code and the Sybase stored procedures.

1.5 Sponsor

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

This benchmark test was sponsored by Digital Equipment Corporation and Sybase, Inc., and attested to by Information Paradigm.

1.6 Parameters and Options

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

- *Database tuning options*
- *Recovery/locking options*
- *Operating system and application configuration parameters*

The Test Sponsor has elected to provide a listing of all parameters and options. Appendix D contains the tunable parameters used in the TPC-C tests.

1.7 Configuration Diagrams

Provide diagrams of both the measured and priced configurations, accompanied by a description of the differences. This includes, but is not limited to:

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

The TPC-C terminal users were emulated by Digital's Remote Terminal Emulator (RTE) software, which ran on one VAX 3100. Each emulated terminal was connected through LAT to a client node.

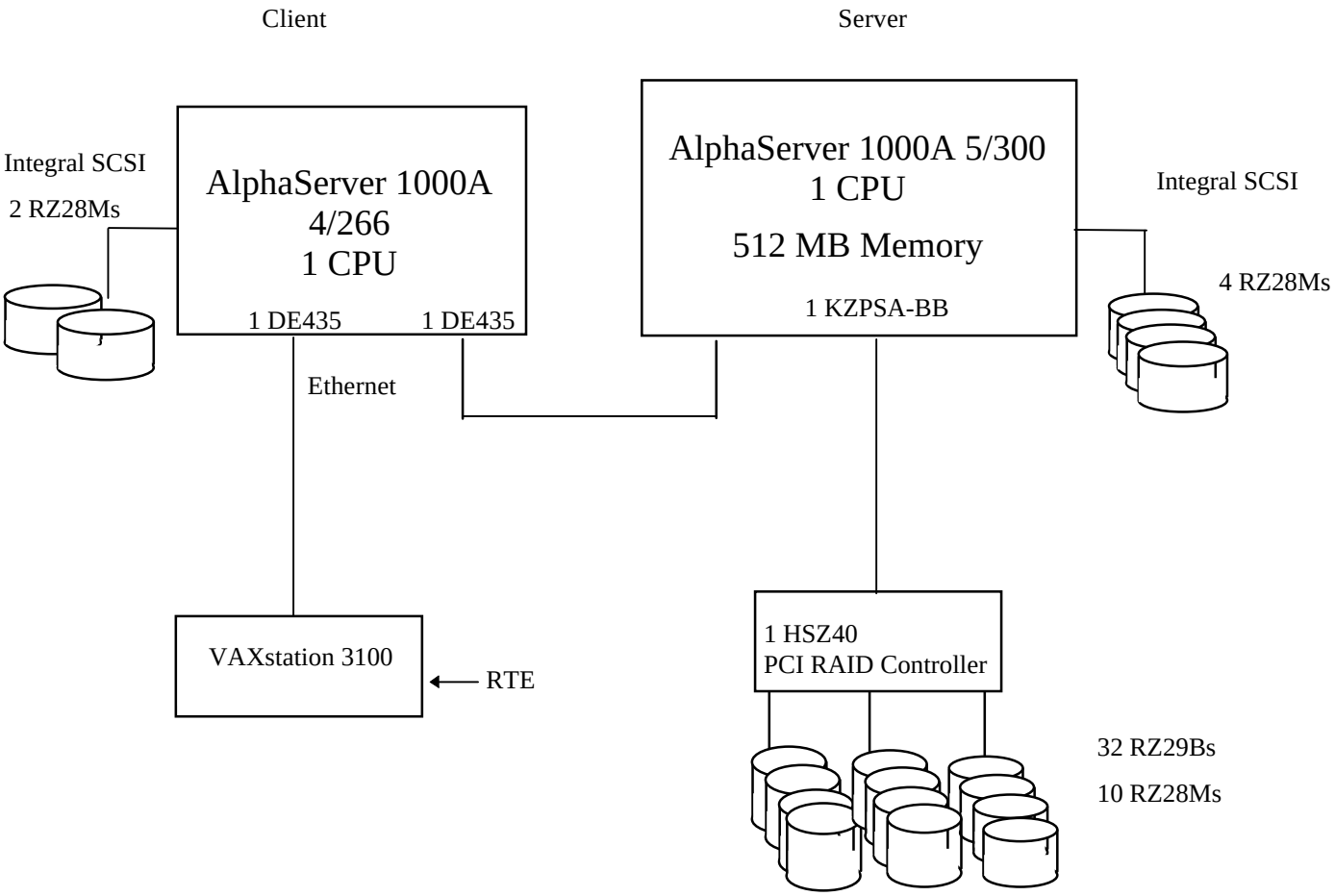
The clients consisted of one AlphaServer 1000A 4/266 computer running the TUXEDO transaction monitor. Each emulated terminal ran a client application which made a request to the TUXEDO services for each transaction. The TUXEDO services used dblib calls to complete the transaction in the database.

The server was an AlphaServer 1000A 5/300 1 CPU configuration running Sybase SQL Server 11.0. The measured server was configured with 42 disks. The priced configuration has 46 disks, which includes the additional 10% spareable disks with a minimum of 2 for each disk type.

The priced server is an AlphaServer 1000A 5/300 with 1 CPU and 512 MB of memory. The benchmarked server was actually an AlphaServer 1000A 5/300.

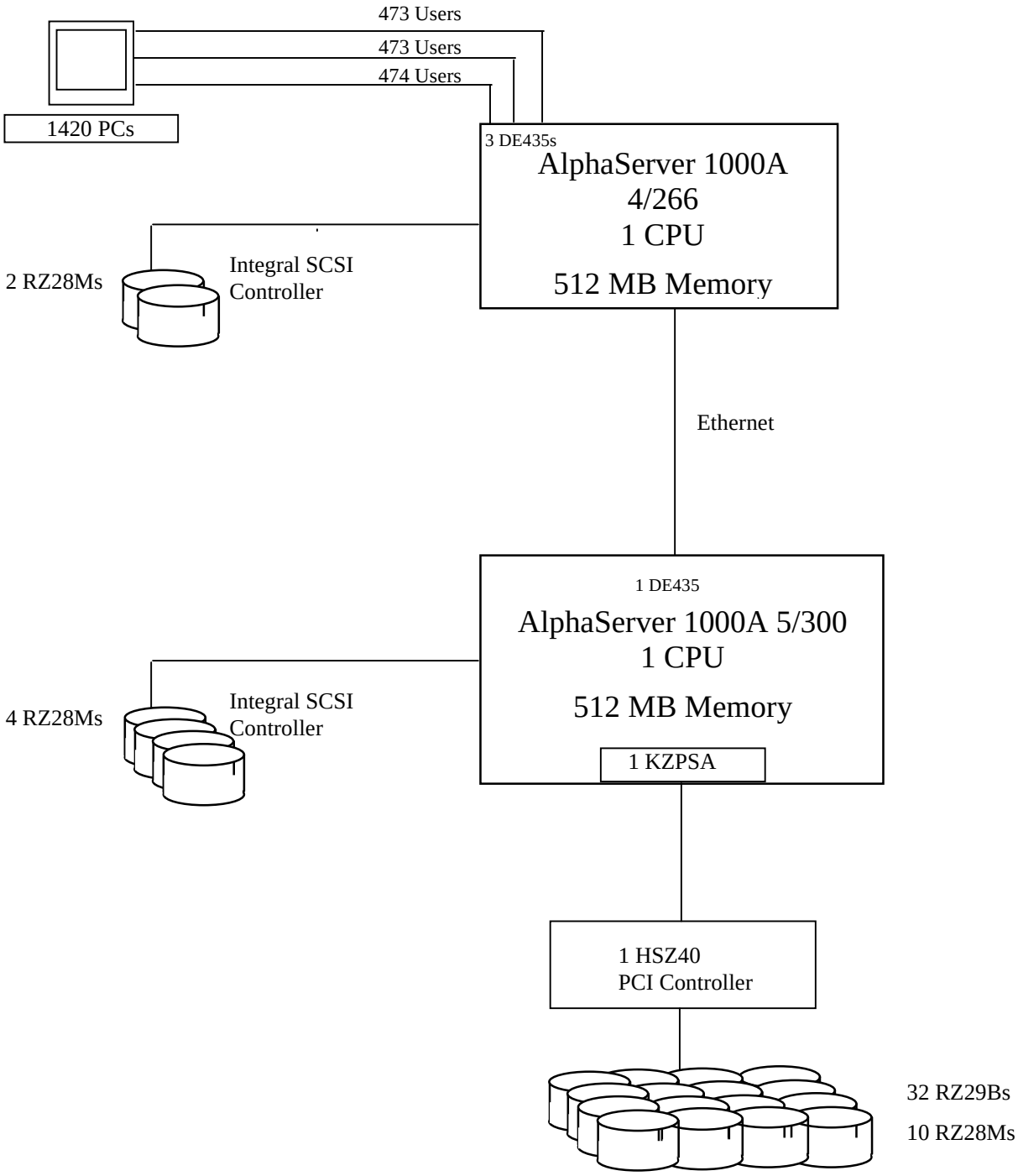
Measured Configuration

The following figure represents the measured configuration. The benchmark system used a remote terminal emulator (RTE) to initiate transactions and measure response times of transactions, as well as record various data for each transaction.



Priced System Configuration

The following figure depicts the priced system, whose cost determines the normalized price per tpmC reported for the test.



2. Logical Database Design Related Items

2.1 Table Definitions

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

Appendix B contains the database definition files that were used to set up the database.

2.2 Table Organization

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

Physical space was allocated to Sybase SQL Server on the server disks according to the details provided in Appendix B. The size of the space segments on each disk was calculated to provide even distribution of data across the disk subsystem. A fill factor was used on the Warehouse and District tables to minimize the amount of data per page. The indices were defined at table definition and were built at the initial table load by executing the database build script in of Appendix B.

2.3 Insert/Delete Operations

It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C 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/or delete operations to any of the tables. The space required for an additional 5% of the initial table cardinality was allocated to Sybase SQL Server and priced as static space.

2.4 Disclosure of Partitioning

While there are 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.

Horizontal partioning was used on the HISTORY table using functionality provided by Sybase SQL Server.

2.5 Replication of Tables

Replication of tables, if used, must be disclosed.

No tables were replicated in this benchmark test.

2.6 Additional and/or Duplicated Attributes in any Table

Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.

No attributes were added or replicated in this benchmark test.

3. Transaction and Terminal Profiles Related Items

3.1 Random Number Generation

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

Random numbers were generated using the drand48() and lrand48() UNIX calls. These functions generate pseudo random numbers using the linear congruential algorithm and 48-bit integer arithmetic. The random number generators are initially seeded using the srand48() call.

3.2 Terminal Input/Output Screen Layouts

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

The screen layouts match the *TPC Benchmark C Standard Specification.*

3.3 Features in Priced Terminals

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

Each of the five transaction types was tested using a VT510 terminal to verify that all the features described in Clause 2.2.2.4 were provided.

Using a VT510 terminal, the independent auditor tested each of the five transactions to verify that all the features described in Clause 2.2.2.4 were provided.

3.4 Presentation Managers

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

Custom client code was written to provide the TPC-C user interface. Screen positioning and underlining on the emulated VT510 was done using VT100-mode escape sequences.

3.5 Home and Remote Order-Lines

The percentage of home and remote order-lines in the New-Order transactions must be provided.

The table in Section 3.10 shows the percentage of home and remote transactions that occurred during the measurement period for the New-Order transactions.

3.6 New Order Roll Back Transition

The percentage of New-Order transactions that were rolled back as a result of an invalid item number musts be provided.

The table in Section 3.10 shows the percentage of home and remote transactions that occurred during the measurement period for the New-Order transaction.

3.7 Number of Order-Lines

The thin-wire between the front-ends and back-ends has a bandwidth of 10 Mbps.

3.8 Home and Remote Payment Transactions

The percentage of home and remote Payment transactions must be provided.

The table in Section 3.10 shows the percentage of home and remote transactions that occurred during the measurement period for the Payment transactions.

3.9 Non-Primary Key Access in Payment and Order-Status

The percentage of Payment and Order-Status transactions that used non-primary key (C_LAST) access to the database must be disclosed.

The table in Section 3.10 shows the percentage of non-primary key accesses to the data base by the Payment and Order-Status transactions.

3.10 Skipped Deliveries

The percentage of Delivery transactions that skipped as a result of insufficient number of rows in the NEW-ORDER table must be disclosed.

The following table summarizes the data required for disclosure from Sections 3.5 through 3.10. The range of acceptable and the measured results are listed.

Section	Description	Acceptable Requirement	Measured Result
3.5	% home order lines in New-Order	(98.95 - 99.05)	99.01
3.5	% remote order lines in New-Order	(0.95 - 1.05)	0.99
3.6	% New-Order transactions rolled back	(0.9 - 1.1)	0.99
3.7	Average number of items per orders	(9.5 - 10.5)	10.00
3.8	% home Payment transactions	(84.0 - 86.0)	84.71
3.8	% remote Payment transactions	(14.0 - 16.0)	15.29
3.9	% non-primary key access, Payment	(57.0 - 63.0)	59.98
3.9	% non-primary key access, Order-Status	(57.0 - 63.0)	59.42
3.10	% skipped deliveries, Delivery	(0.0 - 1.0)	0

3.11 Transaction Mix

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

The following table summarizes the transaction mix.

Transaction Mix (percent)

Transaction Type	New-Order	Payment	Order-Status	Delivery	Stock-Level
Min. Requirement	NA	43.00%	4.00%	4.00%	4.00%
Measured Result	44.29	43.28	4.15	4.06	4.22

3.12 Queuing Mechanism

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

The delivery transactions were submitted using the asynchronous TUXEDO service request call. The request is placed in the service queue, and control is immediately returned to the requester without waiting for the transaction to complete.

4. Transaction and System Properties Related Items

4.1 ACID Properties

The results of the ACID test 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.

Clause 3 of the TPC Benchmark C Standard Specification lists specific tests to ensure the Atomicity, Consistency, Isolation, and Durability (ACID) properties of the SUT. All of the tests, except for the failure of memory durability were previously run and accepted by the auditor.

4.2 Atomicity Tests

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

This test was previously performed and was waived by the auditor.

4.3 Consistency Tests

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.

This test was previously performed and was waived by the auditor.

4.4 Isolation Tests

Operations of concurrent database transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed (i.e. in isolation).

This test was previously performed and was waived by the auditor.

4.5 Durability Tests

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

- *Permanent irrecoverable failure of any single durable medium containing database, ABTH files/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).*

The tests for failure of durable medium containing database tables and durable medium contain database logs were previously performed and were waived by the auditor.

4.5.1 Failure of Memory and Instantaneous Interruption of System

This test was conducted on the fully scaled 142 warehouse database using 1420 emulated terminals.

1. The current number of orders in the database was counted, giving ORDER_COUNT_BEFORE.

2. A test was started and allowed to run at steady state for 2 minutes.
3. The system was shut down.
4. The test was aborted on the RTE.
5. The system was powered back on and rebooted.
6. Sybase was restarted and recovered the database from the transaction log.
7. The current number of orders in the database was counted giving ORDER_COUNT_AFTER. It was verified that ORDER_COUNT_AFTER - ORDER_COUNT_BEFORE was greater than or equal to the number of committed orders recorded by the RTE.
8. Several orders recorded by the RTE were checked in the database to make sure they existed.

5. Scaling and Database Population Related Items

5.1 Cardinalities of the Database Tables

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

The initial cardinalities of the database tables are shown in the following table.

Table Name	Cardinality
WAREHOUSE	142
DISTRICT	1,420
CUSTOMER	4,260,000
HISTORY	4,260,000
ORDER	4,260,000
NEW-ORDER	1,278,000
ORDERLINE	42,600,000
STOCK	14,200,000
ITEM	100,000

5.2 Distribution of Tables and Logs

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

The following benchmark configuration table indicates how the database and system files were allocated on the tested system to meet the 8-hour steady state requirement.

Allocation of System Files on the Tested System Table

Disk Name	File_Name	Table_Name_%	Contents per Disk
/dev/rrzg18b	stock01	Stock	4.7
/dev/rrza20b	stock02	Stock	4.7
/dev/rrzf18b	stock03	Stock	4.7
/dev/rrzg19b	stock04	Stock	4.7
/dev/rrze18b	stock05	Stock	4.7
/dev/rrzf19b	stock06	Stock	4.7
/dev/rrzd18b	stock07	Stock	4.7
/dev/rrzg18d	stock08	Stock	4.7
/dev/rrza20d	stock09	Stock	4.7
/dev/rrzf18d	stock10	Stock	4.7
/dev/rrzg19d	stock11	Stock	4.7
/dev/rrze18d	stock12	Stock	4.7
/dev/rrzf19d	stock13	Stock	4.7
/dev/rrzd18d	stock14	Stock	4.7
/dev/rrzg18e	stock15	Stock	4.7
/dev/rrza20e	stock16	Stock	4.7
/dev/rrzf18e	stock17	Stock	4.7
/dev/rrzg19e	stock18	Stock	4.7
/dev/rrze18e	stock19	Stock	4.7
/dev/rrzf19e	stock20	Stock	4.7
/dev/rrzd18e	stock21	Stock	4.7
/dev/rrze19b	customer01	Customer	8.3
/dev/rrzc18b	customer02	Customer	8.3
/dev/rrzd19b	customer03	Customer	8.3
/dev/rrzb18b	customer04	Customer	8.3
/dev/rrze19d	customer05	Customer	8.3
/dev/rrzc18d	customer06	Customer	8.3
/dev/rrzd19d	customer07	Customer	8.3
/dev/rrzb18d	customer08	Customer	8.3
/dev/rrze19e	customer09	Customer	8.3
/dev/rrzc18e	customer10	Customer	8.3
/dev/rrzd19e	customer11	Customer	8.3
/dev/rrzb18e	customer12	Customer	8.3
/dev/rrzb20b	tpcc_log	Log	100
/dev/rrza18d	orders01	Orders	25
/dev/rrza18e	orders02	Orders	25
/dev/rrza18f	orders03	Orders	25
/dev/rrza18g	orders04	Orders	25
/dev/rrza19b	order_line01	Order_line	25
/dev/rrza19d	order_line02	Order_line	25
/dev/rrza19e	order_line03	Order_line	25
/dev/rrza19f	order_line04	Order_line	25
/dev/rrzc19b	history01	History	100
/dev/rrzb19b	cidx01	Cust_index	100
/dev/rrza18b	master	Master	100

The distribution of the database tables over all the disks of the priced system is an extension of the distribution in the tested system. One hundred eighty (180) day storage requirements are satisfied with the unused space on the tested system disks.

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, all 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.*

The database used for this testing was Sybase SQL Server 11.0 from Sybase Inc. Sybase SQL Server 11.0 is a relational DBMS.

5.4 Database Partitions/Replications

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

Horizontal partioning was used on the History Table. The functionality for this was provided by Sybase SQL Server. For further details of the partitioning of the database, see Appendix B.

5.5 180 Days Space Requirement

The calculations for arriving at the 180-day space computations, as defined in Clause 4.2.3 must be disclosed.

Note : Numbers are in KBytes unless otherwise specified						
Warehouses	142	**tpmC**	1691.04	**tpmC/W**	11.91	
Table	**Rows**	**Data**	**Index**	**5% Space**	**8H Space**	**Total Space**
Warehouse	142	284	4	14		302
District	1,420	2,840	16	143		2,999
Item	100,000	9,524	86	192		9,802
New-order	1,278,000	13,968	168		2,840	16,976
History	4,260,000	238,592	0		36,656	275,248
Orders	4,260,000	115,136	1,388		17,902	134,426
Customer	4,260,000	2,840,000	237,578	61,552		3,139,130
Order-line	42,600,000	2,581,820	33,970		401,872	3,017,662
Stock	14,200,000	4,733,334	52,592	95,719		4,881,645
Totals		10,535,498	325,802	157,619	459,269	11,478,189
Segment	**LogDev Cnt.**	**Seg. Size**	**Needed**	**Overhead**		**Not Needed**
wdino	1	512,000	30,380	304		481,316
history	1	716,800	278,000	2,780		436,020
order	4	409,600	135,770	1,358		272,472
customer	12	3,112,960	3,170,521	31,705		(89,266)
order_line	4	3,276,800	3,047,838	30,478		198,483
stock	21	5,058,560	4,930,461	49,305		78,794
Totals		13,086,720	11,592,971	115,930		1,377,820
Dynamic space	2,844,546	Sum of Data for Order, Order-Line & History (excl. free extents)				
Static space	8,290,303	Data + Index + 5% Space + Overhead - Dynamic space				
Free space	574,051	Total Seg. Size - Dynamic Space - Static Space - Not Needed				
Daily growth	541,999	(Dynamic space/W * 62.5)* tpmC				
Daily spread	(238,947)	Free space - 1.5 * Daily growth (zero if negative)				
180 day (KB)	105,850,122	Static space + 180 (daily growth + daily spread)				
180 day (GB)	100.95	Excludes OS, Paging and RDBMS Logs				
Log per N-O txn	2.53	Number of 2K blocks per New-Order transaction				
8 Hour Log (GB)	3.92					

6. Performance Metrics and Response Time Related Items

6.1 Reporting All Data

Measured tpmC must be reported.

All the data required by Clause 5 is reported below in Section 6.2 through 6.10. The measured tpmC for the AlphaServer 1000A 5/300 1 CPU C/S configuration was 1691.04 tpmC.

6.2 Response Times

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

Response Times (seconds)

Transaction	90th percentile	Average	Maximum
New-Order	1.7	1.1	11.1
Payment	1.3	0.8	11.3
Order-Status	1.5	1.0	6.8
Delivery (interactive)	0.3	0.3	0.4
Delivery (deferred)	3.0	1.0	6.0
Stock-Level	8.3	4.2	24.5
Menu	0.3	0.3	1.2

6.3 Think and Keying Times

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

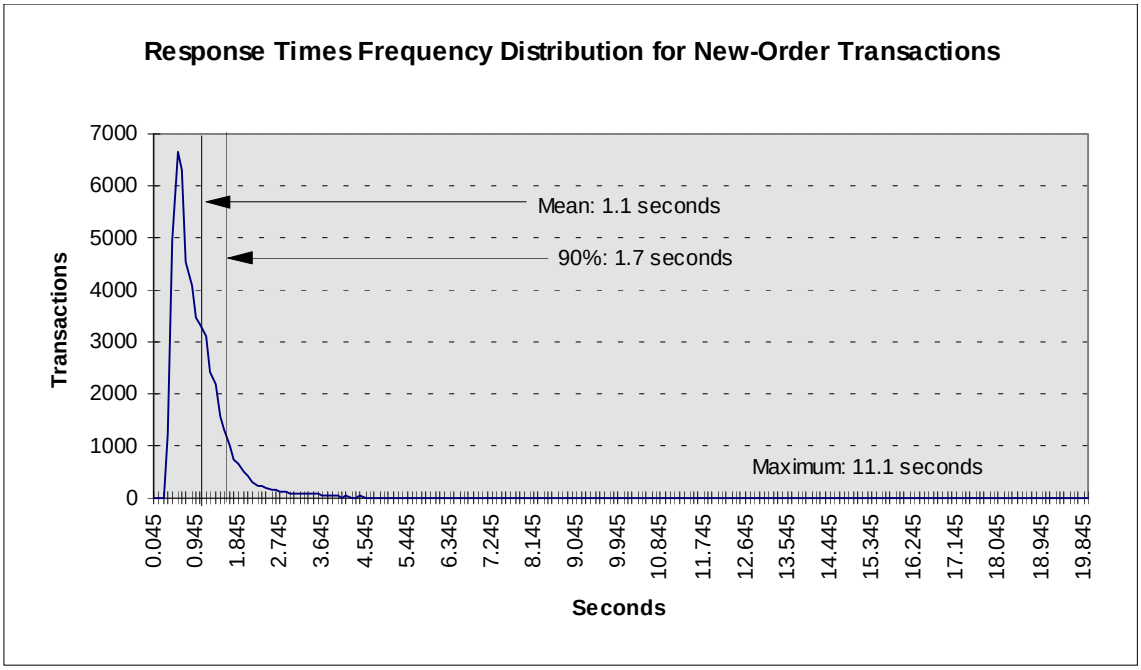
Keying/Think Times (seconds)

Transaction	Min.	Average	Max.
New-Order	18.0/0.0	18.0/12.1	18.1/159.7
Payment	3.0/0.0	3.0/12.2	3.1/135.9
Order-Status	2.0/0.0	2.0/10.1	2.1/98.1
Delivery	2.0/0.0	2.0/5.1	2.1/45.4
Stock-Level	2.0/0.0	2.0/5.0	2.1/42.1

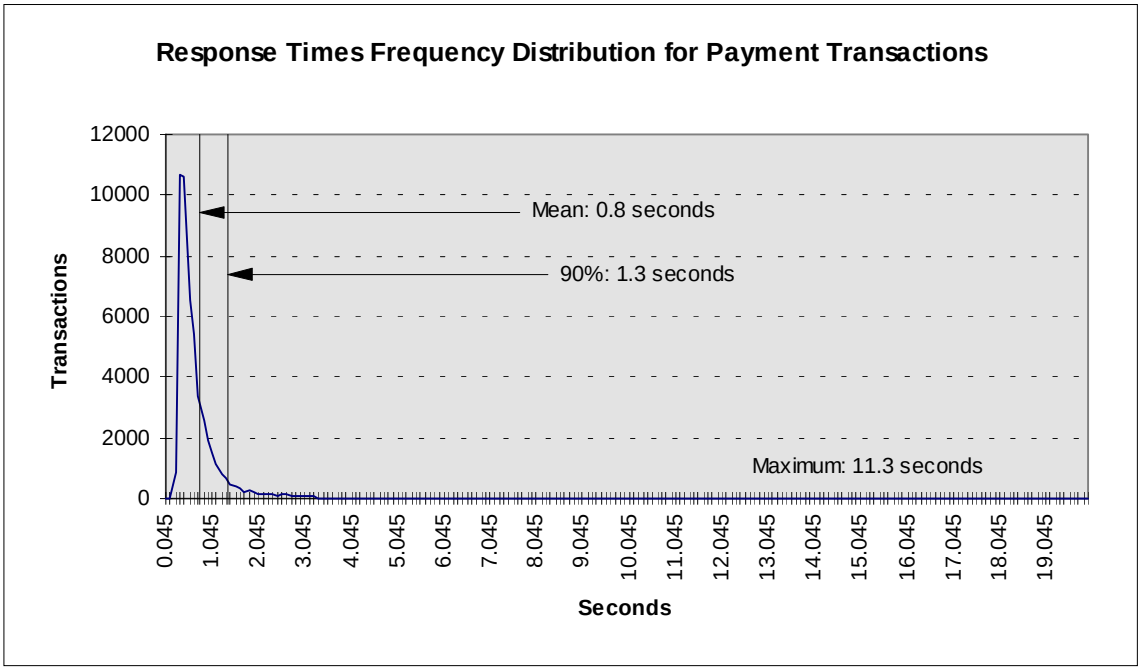
6.4 Response Times Frequency Distribution

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

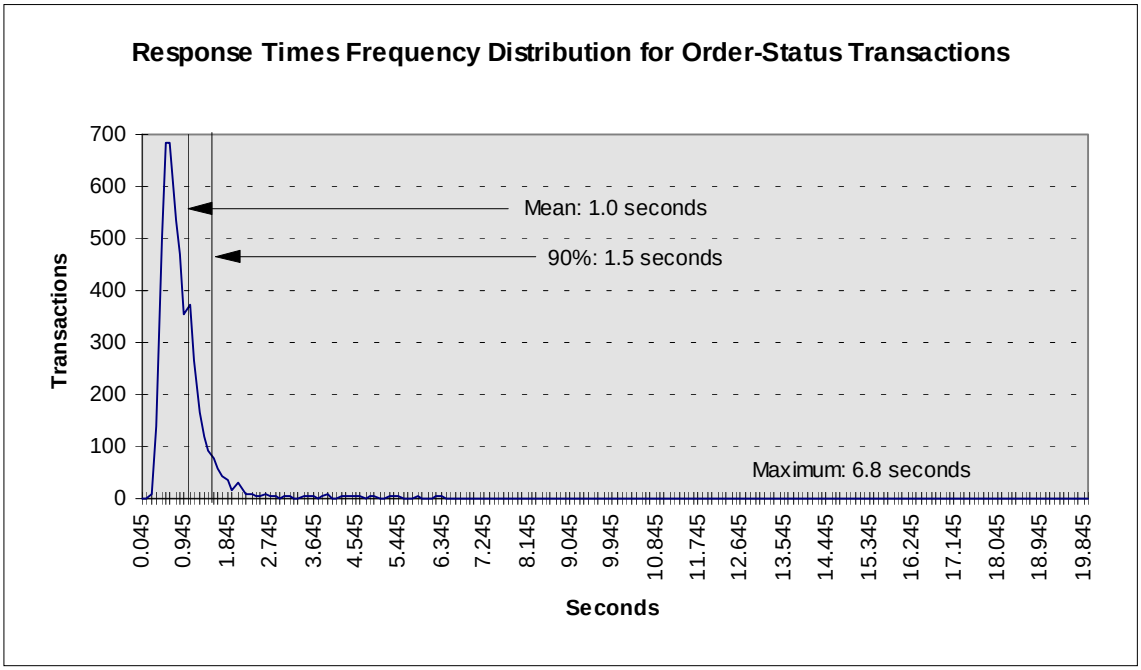
Response Times Frequency Distribution for New-Order Transactions



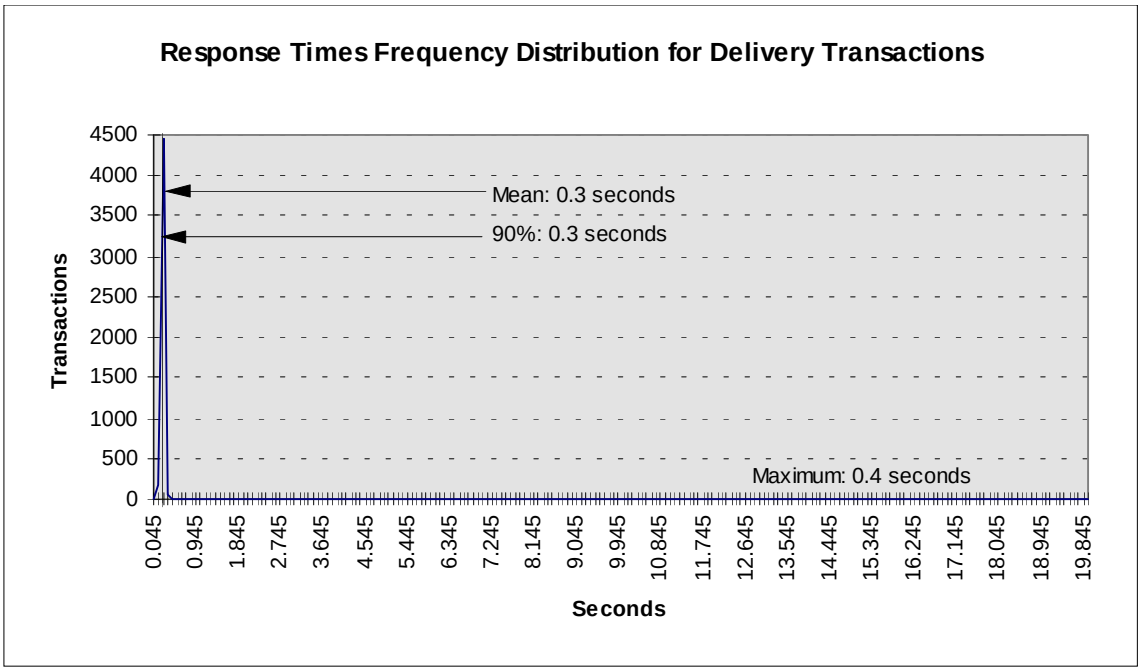
Response Times Frequency Distribution for Payment Transactions



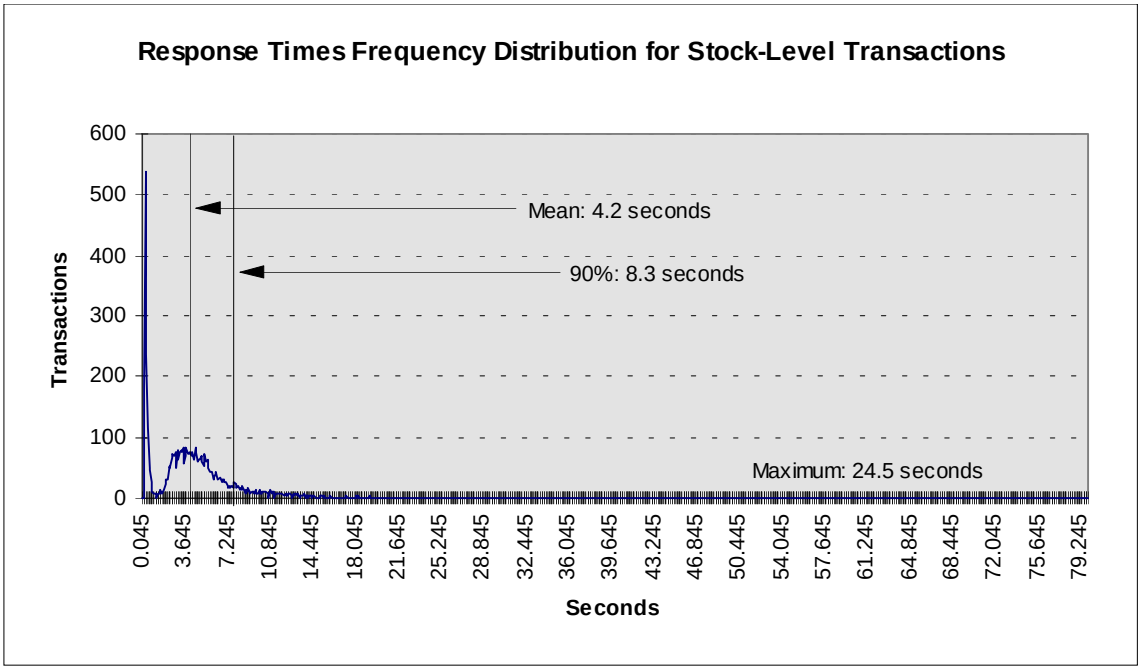
Response Times Frequency Distribution for Order-Status Transactions



Response Times Frequency Distribution for Delivery Transactions



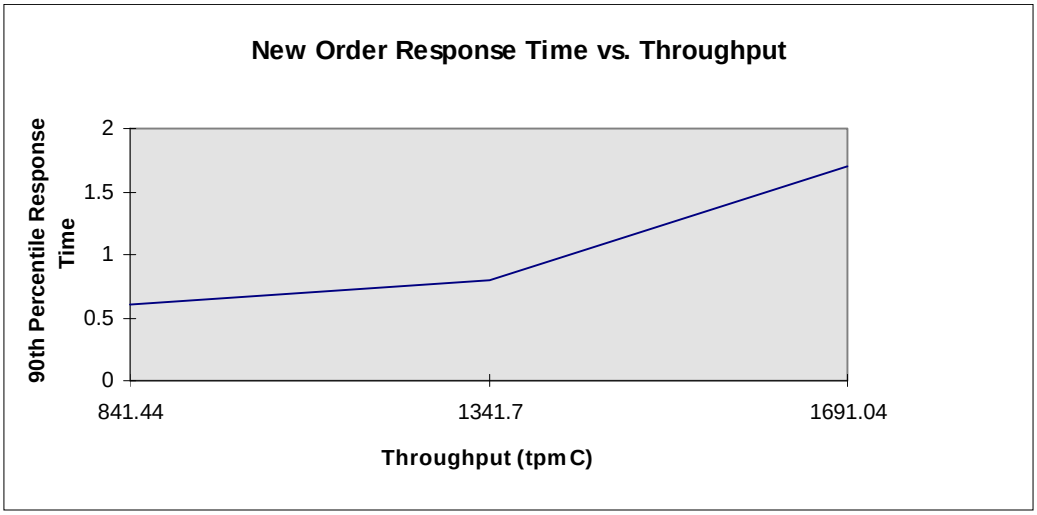
Response Times Frequency Distribution for Stock-Level Transactions



6.5 Response Time versus Throughput Performance Curve

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

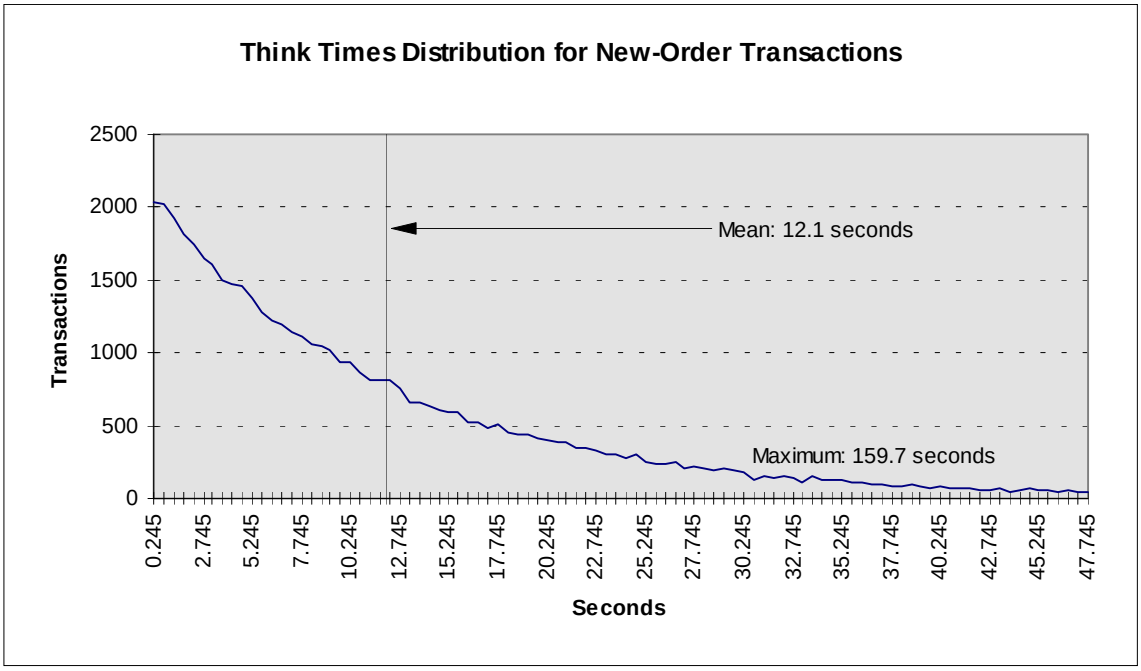
New-Order Response Time versus Throughput



6.6 Think Times Frequency Distribution

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

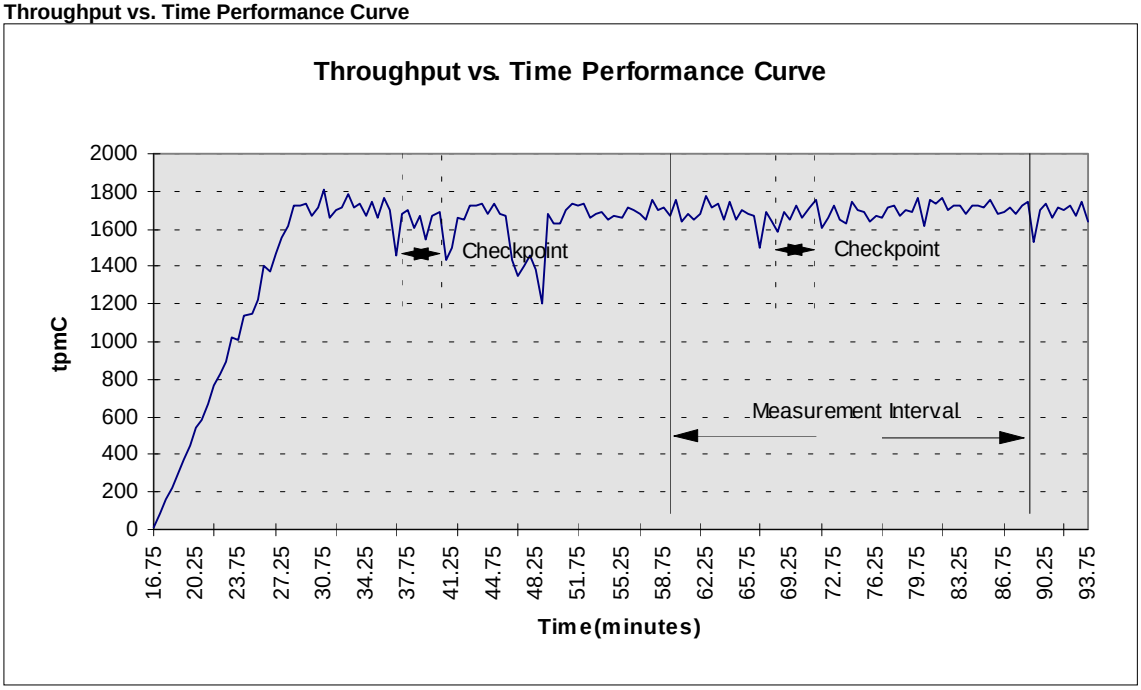
Think Times Distribution for New-Order Transactions



6.7 New-Order Throughput vs. Elapsed Time

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

Throughput vs. Time Performance Curve



6.8 Steady State

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.

Confirmation that the SUT has reached steady state prior to the beginning of the data collection measurement interval is based on a visual inspection of the plots of tpmC versus time.

The graph in Section 6.8 plots the average tpmC versus time, averaged over 30 second intervals, and shows that steady state was reached before the data was collected. The ramp-up and steady-state stages are clearly visible.

6.9 Work Performed During Steady State

A description of how the work normally performed during a sustained test actually occurred during the measurement interval must be reported.

6.9.1 Transaction Flow

For each of the TPC Benchmark C transaction types, the following steps are executed.

TUXEDO for Digital UNIX, which is based on TUXEDO System/T Transaction Processing Monitor Version 4.2, was used as a transaction manager (TM). Each transaction was divided into two programs. The front-end program handled all screen I/O, while the back-end program handled all database operations. Both the front-end and back-end programs ran on the client system.

The front-end program communicates with the back-end program through TUXEDO messages. The back-end program communicates with the server system over Ethernet using Sybase Open Client DB-Library/C calls. Besides telling TUXEDO functions for user connection and message communication, all other functions are transparent to the application code. TUXEDO routes the transaction and balances the load according to the options defined in the TUXEDO configuration file listed in Appendix B. The transaction flow is described next.

- When TUXEDO boots up, it creates one or more server process(es) for each transaction. Several server processes were defined in the TUXEDO configuration file.
- Each TPC-C user invokes the TPC-C main (front-end) program.
- The TPC-C main program connects to the TUXEDO before starting any transaction operation.
- The TPC-C main program displays the TPC-C transaction menu on the user terminal.
- The TPC-C user chooses the transaction type and proceeds to fill the screen fields required for that transaction.
- The TPC-C main program accepts all values entered by the user and transmits those values to one of the TPC-C back-end programs. The transmission is performed through a TUXEDO function call. Each TPC-C back-end program has a “service-name.” This service-name is specified whenever the TPC-C main program requests a TUXEDO service. TUXEDO routes that message according to the service-name and the information defined in the TUXEDO configuration file.
- A TPC-C back-end server receives a message from its queue and proceeds to execute all database operations related to the service-name specified. All the information entered on the user terminal is contained in the TUXEDO message.
- Once the transaction is committed, the TPC-C back-end server program loads the message buffer with the transaction output and returns control to the TUXEDO manager.
- TUXEDO manager routes the message back to the TPC-C main program.

- The TPC-C main program takes the message content and writes the transaction output on the user terminal.

6.9.2 Database Transaction

All database operations are performed by the TPC-C back-end programs. The process is described next.

- Using Sybase Open Client DB-Library calls, the TPC-C back-end program interacts with Sybase SQL Server to perform SQL data manipulations such as update, select, delete, and insert, as required by the transaction. After all database operations are performed for a transaction, the transaction is committed.
- Sybase SQL Server proceeds to update the database as follows:

When Sybase SQL Server changes a database table with an update, insert, or delete operation, the change is initially made in memory, not on disk. When there is not enough space in the memory buffer to read in or write additional data pages, Sybase SQL Server will make space by flushing some modified pages to disk. Modified pages are also written to disk when a checkpoint occurs. Before a change is made to the database, it is first recorded in the transaction log. This ensures that the database can be recovered completely in the event of a failure. Using the transaction log, transactions that started but did not complete prior to a failure can be undone, and transactions recorded as complete in the transaction log but not yet written to disk can be redone.

6.10 Determining Reproducibility

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

The experiment at the maximum targeted tpmC level was repeated once to ensure reproducibility. The computed tpmC for each experiment was within 1% of the reported tpmC, and all 90th percentile response times were under their respective limits.

6.11 Duration of Measurement Period

A statement of the duration of the measurement period for the reported maximum qualified throughput (tpmC) must be included.

Each experiment was run for a minimum of 60 minutes. The data collection period was 30 minutes and started approximately 30 minutes after all simulated users had begun executing transactions.

6.12 Method of Regulation of the Transaction Mix

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

The weighted distribution method was used. No adjustment was made by the RTE scripts. See Appendix C for more details.

6.13 Percentage of the Total Mix

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

See Section 3.10.

6.14 Percentage of New-Order Transactions Rolled Back

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

See Section 3.10.

6.15 Average Number of Order-Lines

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

See Section 3.10.

6.16 Percentage of Remote Order-Lines

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

See Section 3.10.

6.17 Percentage of Remote Payment Transactions

The percentage of remote Payment transactions must be disclosed.

See Section 3.10.

6.18 Percentage of Customer Selections

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

See Section 3.10.

6.19 Percentage of Delivery Transactions

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

See Section 3.10.

6.20 Number of 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.

A checkpoint is the process of writing all modified data pages to disk. The TPC-C benchmark on the AlphaServer 1000A 5/300 1 CPU C/S system was set up to automatically checkpoint every 30 minutes. One checkpoint occurs during the warm-up period, and another occurs during the measurement period.

7. SUT, Driver, and Communication Definition Related Items

7.1 Description of RTE

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 emulated 1420 Multias connecting over three 10 megabits per second (Mbps) network. The front-end clients are also connected by a 10 Mbps network to the server.

7.2 Driver Functionality and Performance

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.

Due to the large number of PCs and associated hardware that would be required to run these tests, a Remote Terminal Emulator was used to emulate the connected PCs and LAN.

The PC emulated in the test is the Multia Multi-Client Desktop System. The Multia is a hardware and software desktop system that combines features of a PC (Windows applications), workstation (graphics performance), and terminal within a single desktop device. Its hardware includes a 166 Mhz Digital Alpha processor, with a custom graphics accelerator and supports VT340 graphics. Systems include a 340 MB hard drive and 24 MB SIMM memory, ThinWire, thick wire and twisted pair Ethernet, two serial lines, and a bi-directional parallel port. Multia software includes Microsoft Windows NT Workstation V3.5 software, and optimized X11.R6 Server, and multiple terminal emulators (including VT100 emulation), and many network protocols, including TCP/IP, LAT, and DECnet.

As configured for this test, the driver software emulates the traffic that would be observed from Multias connected by Ethernet to the front-end clients using the Telnet protocol.

7.3 Functional Diagrams and Details of Driver System

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).

The diagrams in Section 1.7 show the tested and priced benchmark configurations.

7.4 Network Configurations and Driver System

The network configurations of both the tested service 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).

Section 1.7 in this report has a picture of the network configurations of both the tested service and the proposed (target) services.

7.5 Network Bandwidth

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

The Ethernet used in the local area network (LAN) between the emulated terminals and the front-end systems complies with the IEEE 802.3 standard and has a bandwidth of 10 megabits per second (Mbps).

The thin-wire between the front-end clients and the server has a bandwidth of 10 Mbps.

7.6 Operator Intervention

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

No operator intervention was required.

8. Pricing Related Items

8.1 Hardware and Software Components

A detailed list of hardware and software used in the priced system. Each item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package pricing is used, contents of the package must be disclosed.

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.

The detailed list of all hardware and software for the priced configuration is listed in the system pricing summary.

8.1.1 Hardware Pricing

Digital Equipment Corporation's TPC Benchmark C tests used packaged hardware systems whenever possible to simplify configurations to the fewest number of line items.

All hardware and software products have been priced to satisfy the 5 years warranty and 5 x 8 with 4 hours response requirements.

The hardware prices for the AlphaServer 1000A 5/300 1 CPU, one AlphaServer 1000A 4/266, and their associated components (e.g., memory, storage subsystem, etc.) are based on Bay State Computer Group and MacWarehouse Warehouse price quotations.

The 8 GB DAT tape drive and terminal server service warranty are based on Digital Equipment Corporation’s product and service pricing.

8.1.2 Software Pricing

The priced system uses the following software products:

- Digital UNIX operating system
- TUXEDO transaction processing monitor
- Sybase SQL Server relational database management system
- DEC C compiler

The license purchase includes 1 year of warranty service. Four years of additional software warranty is provided for a total of 5 years extended warranty. The software warranty and service level are the same as the service level for the hardware system on which the software operates.

The level of post-warranty software service is Layered Product Support (LPS).

8.1.3 Warranty Pricing

In addition to the base warranty, additional warranty has been priced to satisfy the 5-year TPC-C warranty requirements of all products.

8.1.4 Price Discounts

See Appendix F.

8.2 Availability Status

The committed delivery date for general 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.

The version of Sybase SQL Server used in the measurement and all other software and hardware components used in the tested and priced systems are available now.

8.3 Performance and Price/Performance

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

The following table shows the measured tpmC and price/tpmC results for the tested systems:

CPU Model	Configuration	Software	tpmC	Price per tpmC $tpmC
AlphaServer 1000AA 5/300	1 CPU	Digital UNIX V3.2F-2 Sybase SQL Server 11.0 TUXEDO V4.2	1691.04	$158.24 /tpmC

8.4 Country Specific Pricing

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

None.

8.5 Usage Pricing

For any usage pricing, the sponsor must disclose:

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

None.

9. Audit Related Items

9.1 Audit

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

Appendix E contains the complete independent auditor's by Francois Raab of Information Paradigm for the test described in this report.

Appendix A

TPC-C Client Code

```
/*************************************************************
*******************
*******************************  tpcc.c
*****************************************
*************************************************************
*******************/
/*
**  tpcc.c: Main client code for TPC-C.
*/

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "tpcc.h"
#include "socket.h"
#include "screen.e"
#ifdef TPMONITOR
#include "monitor.e"
#else
#include "db_funcs.e"
#endif
main(int argc, char **argv)
{
  int menu_selection;
  void do_transaction(int);

   initialize(argc,argv);
   Send_Menu();
   while ((menu_selection = Get_Menu_Input()) != 9) {
      if ( (menu_selection < 1) || (menu_selection > 5)) continue;
      do_transaction(menu_selection-1);
      Send_Menu();
   }
   rundown(OKAY);
}

initialize(int argc, char **argv)
{
 int menu_selection, start,m,n;
 char
list[]="0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNO
PQRSTUVWXYZ";


#ifdef SOCKET

/* 6/26/95 Changed to only look at SVR4, which starts at /dev/lat621 */

   start = atoi(argv[1]) + 620;
   sprintf(tty_name,"/dev/lat/%d",start);

   if ((start-620)%100 == 0) printf("opening device %d %s\n", start,
tty_name);
/*
   printf("opening %s\n", tty_name);
*/
   if ((tty_in=tty_out=open(tty_name, O_RDWR ,0)) == -1)
       syserr("Can't open terminal");

   if (imitate_login_sequence() == -1) argc = 1;
   else argc = 3;
#else
    tty_in = 0;
    tty_out =1;
#endif


    if (argc < 3)
       prompt_for_inputs(argc,argv);
#ifndef SOCKET
   else {
      w_id = atoi(argv[1]);
      d_id = atoi(argv[2]);
   }
#endif


#ifdef TPMONITOR

   if (Init_Monitor() ) {
      fprintf(stderr,"Unable to connect to TP Monitor\n\01");
      rundown (NO_TPM_CONNECT);
   }

#else
   if (DBinit()) {
      syserr("Problems in DBinit\n");
   }

#endif /*TP MONITOR*/


   Init_Screen();
}

rundown(int status)
{

   Restore_Screen();
#ifdef TPMONITOR

   if (status != NO_TPM_CONNECT) Rundown_Monitor();
#else

   DBdone();

#endif /*TPMONITOR */

#ifdef SOCKET
   close(tty_in);
#endif
}


prompt_for_inputs(int argc,char **argv)
{
int i;
   char *endp;
   static char buffer[30];
   BOOLEAN good_answer = FALSE;

      while (!good_answer) {
        i=write(tty_out,"Enter Warehouse ID:",19);
        i=read(tty_in,buffer,sizeof(buffer));
        if (i == -1)
              {
               syserr("writei");
              }
        w_id = (int) strtol(buffer,&endp,0);
        if (w_id > 0) good_answer = TRUE;
      }
   write(tty_out,"Enter District ID:" ,19);
   read(tty_in,buffer,sizeof(buffer));
   d_id = atoi(buffer);
}

#ifdef SOCKET
int imitate_login_sequence()

{
   char buffer[128],temp[128];
   int i;

   if (read(tty_in,buffer,sizeof(buffer)) == -1)
      syserr("read error");
```

```
   if (write(tty_out,"login:",6) == -1)
      syserr("write error");
   if (read (tty_in,buffer,sizeof(buffer))== -1)
      syserr("read error");

   if (write(tty_out,"Password:",9) == -1)
      syserr("write error");
   if (read (tty_in,buffer,sizeof(buffer)) == -1)
      syserr("read error");

   if (write(tty_out,">",1) == -1)
      syserr("write error");
   if (read (tty_in,buffer,sizeof(buffer)) == -1)
      syserr("read error");

         /* find out if warehouse & district were specified */
   i = sscanf(buffer,"%s %*s %d %d",temp,&w_id,&d_id);
   if (!strcmp(temp,"csh")) {
      freopen(tty_name,"r",stdin);
      freopen(tty_name,"w",stdout);
      freopen(tty_name,"w",stderr);
      system("exec csh");
      exit (1);
   }
   if (i == 2) i = sscanf(buffer,"%*s %d %d",&w_id,&d_id);

   return i-1;
}

#endif
void do_transaction(int num) {
   int status;


   Paint_Screen(num);
   status=Get_Form_Data(num);
   if (status == QUIT)
      return;

#ifdef TPMONITOR
   status = Snd_Txn_To_Monitor(num);
   if (status == TPM_ERROR) {
      syserr("\033[24;1H\033[0mTPM Error detected -- See
$TPCC_HOME/CLIENTLOG for details\n");
   }

#else
   status = Do_DBtxn(num);
   if (status != 0) {
      syserr("ERROR:");
   }
#endif /*TPMONITOR*/
   Display_Results(num);
}


----------
/*************************************************************
******************
*******************************  tpcc.h
*************************************
*************************************************************
******************/
/*
**  tpcc.h:  This header file declares data structures for use in application
**           and server
*/
#include <time.h>

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



#define BOOLEAN int
#define VMS     0
#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 MAX_OL  15


/* error codes */
#define OKAY                            0
#define TPM_ERROR     1
#define DB_ERROR      2
#define ITEM_ERROR  3
#define NO_TPM_CONNECT  4

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 io_order_line {
   int     ol_i_id;
   int     ol_supply_w_id;
   int     ol_quantity;
   char    i_name[25];
   int     s_quantity;
   char    b_g[2];
   double  i_price;
};

struct io_neworder {
   int     w_id;
   int     d_id;
   int     c_id;
   int     o_ol_cnt;
   int     o_all_local;
   struct io_order_line o_ol[MAX_OL];

   char    o_entry_d[20];
   char    c_last[17];
   char    c_credit[3];
   float   c_discount;
   float   w_tax;
   float   d_tax;
   int     o_id;
   int     status;
   double  tax_n_discount;
};

struct  io_payment {
   BOOLEAN byname;
   int     w_id;
   int     d_id;
   int     c_id;
   char    c_last[17];
   int     c_w_id;
   int     c_d_id;
   double  h_amount;
```

```
    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[10];
    char    d_street_1[21];
    char    d_street_2[21];
    char    d_city[21];
    char    d_state[3];
    char    d_zip[10];
    char    c_first[17];
    char    c_middle[3];
    char    c_street_1[21];
    char    c_street_2[21];
    char    c_city[21];
    char    c_state[3];
    char    c_zip[10];
    char    c_phone[17];
    char    c_since[11];
    char    c_credit[3];
    double  c_credit_lim;
    float   c_discount;
    double  c_balance;
    char    c_data[201];
};

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

struct  io_ordstat {
    BOOLEAN byname;
    int     w_id;
    int     d_id;
    int     c_id;
    char    c_last[17];

    char    c_first[17];
    char    c_middle[3];
    double  c_balance;
    int     o_id;
    char    o_entry_d[64];
    int     o_carrier_id;
    int     ol_cnt;
    struct status_order_line    s_ol[MAX_OL];
};

struct  io_delivery  {
    int     w_id;
    int     o_carrier_id;
    time_t queue_time;
    int     status;
};

struct  io_stocklev {

    int     w_id;
    int     d_id;
    int     threshold;

    int     low_stock;
};

/*
**  Data structure for input & output data
*/

struct io_tpcc {
    int type;
    union {
        struct io_neworder neworder;
        struct io_payment payment;
        struct io_ordstat ordstat;
        struct io_delivery delivery;
        struct io_stocklev stocklev;
    }  info;
};

--------------------------
/*************************************************************
*******************
****************************** screen.c
*************************************
*************************************************************
******************/
/*
**  screen.c:  Contains all routines for the TPC-C screen display, input and
**             output.
**
*************************************************************
*****************

                    COPYRIGHT (C) 1994 BY
                DIGITAL EQUIPMENT CORPORATION, MAYNARD
                 MASSACHUSETTS.  ALL RIGHTS RESERVED.

  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY
BE USED AND COPIED
  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE
AND WITH THE INCLUSION
  OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY
OTHER COPIES
  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE
AVAILABLE TO ANY OTHER
  PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS
HEREBY TRANSFERRED.

  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO
CHANGE WITHOUT NOTICE AND
  SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
EQUIPMENT CORPORATION.

  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS
  SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY
DIGITAL.

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

 **/

#include <stdio.h>
#include <sys/termio.h>
#include <stdlib.h>

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

#include "tpcc.h"
#include "screen.e"
#include "screen.h"
#include "loopback.e"
#ifdef TPMONITOR
#include "monitor.e"
#else
#include "no_tpm.h"
#endif

extern void Clog(char *, ...);
extern void SCREENlog(int ,char *);
#define MAXLINE 256

struct termio orig_tbuf_in,orig_tbuf_out;

void setraw()  /** put terminals into rawmode  **/
{
    struct termio tbuf;
```

```
   int status;
   if (ioctl(tty_out, TCGETA, &tbuf) == -1)
     syserr("ioctl_ERROR#1 - getting the original input term setting
error");
   orig_tbuf_out = tbuf;
   if (ioctl(tty_in, TCGETA, &tbuf) == -1)
     syserr("ioctl_ERROR#1 - getting the original output term setting
error");
   orig_tbuf_in = 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;

   ioctl(tty_in, TCFLSH, 2 );
   ioctl(tty_out, TCFLSH, 2);
   ioctl(tty_in, TCSETAW, &orig_tbuf_in);
   ioctl(tty_out, TCSETAW, &orig_tbuf_out);
/*
   if (ioctl(tty_out, TCSETAF, &orig_tbuf_out) == -1)
      syserr("ioctl_ERROR#3 - restoring original output terminal settings
error");

   if (ioctl(tty_in, TCSETAF, &orig_tbuf_in) == -1)
      syserr("ioctl_ERROR#4 - Forcing the original settings back for
STDIN error");
*/
}

int Get_Menu_Input()
{
   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;
   c = atoi(inbuf);
   return c;
}

int Get_Form_Data(int txn_type)
{
   BOOLEAN done=FALSE;
   int i, returned_key;
   io_elem *ioptr;
   int last_input;



   BOOLEAN check_neworder_inputs(int *);
   BOOLEAN check_payment_inputs(int *);
   BOOLEAN check_ordstat_inputs(int *);
   BOOLEAN check_delivery_inputs(int *);
   BOOLEAN check_stocklev_inputs(int *);


   BOOLEAN (*p_check_function[])() = {
      &check_neworder_inputs,
      &check_payment_inputs,
      &check_ordstat_inputs,
      &check_delivery_inputs,
      &check_stocklev_inputs
   };

   memset(ip,'\0',sizeof(struct io_tpcc));
   memset(orig_ol,'\0',sizeof(orig_ol));
   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 (i == 5 && txn_type == PAYMENT) payment_input = TRUE;
      if (i != 5 && txn_type == PAYMENT) 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 LF: case TAB:
            if (i == last_input) i = 0;
            else i++;
            break;
         case QUIT:
            done = TRUE;
            break;
         case SUBMIT:
            payment_input = FALSE;
            done = (p_check_function[txn_type])(&i);
            ip->type = txn_type;
            break;
      }
    }
    return returned_key;

}

BOOLEAN check_neworder_inputs(int *pos)
{
   BOOLEAN done = FALSE;
   struct io_order_line *real_ol_ptr;
   struct orig_order_line_struct *orig_ol_ptr;
   int i;


   iNO->w_id = w_id;
   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  {

      orig_ol_ptr = orig_ol;
      real_ol_ptr = iNO->o_ol;

      iNO->o_all_local = 1;

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

                        /* Is there data on this line? */
         if (orig_ol_ptr->o_ol.ol_i_id || orig_ol_ptr->o_ol.ol_supply_w_id
               || orig_ol_ptr->o_ol.ol_quantity) {
                        /* and is that data complete */
            if (orig_ol_ptr->o_ol.ol_i_id && orig_ol_ptr-
>o_ol.ol_supply_w_id
                  && orig_ol_ptr->o_ol.ol_quantity)        {
                        /* if fine, then copy to io struct */
```

```
                iNO->o_ol_cnt++;
                memcpy(real_ol_ptr,orig_ol_ptr,
                        sizeof(struct io_order_line));
                if (iNO->w_id != real_ol_ptr->ol_supply_w_id)
                   iNO->o_all_local = 0;
                real_ol_ptr++;
                orig_ol_ptr->ol_loc= iNO->o_ol_cnt;
              }                 /* if not fine, go back */
              else {
                *pos = 2 + 3*i;
                PAINTSCR(INCOMPLINE_MSG);
                message = TRUE;
                iNO->o_ol_cnt = 0;
                iNO->o_all_local = 1;
                return FALSE;
              }
            }
          }
          if (!iNO->o_ol_cnt) {
            *pos = 2;
            PAINTSCR(MANDATORY_MSG);
            message = TRUE;
            iNO->o_ol_cnt=0;
            memset(orig_ol,'\0',sizeof(struct orig_order_line_struct));
            return FALSE;
          }

          done = TRUE;
#ifdef DEBUG
Clog("Recieved proper Neworder inputs\n");
#endif
      }

      return done;
}



BOOLEAN check_payment_inputs(int *pos)
{

      BOOLEAN 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 = 3;
         message = TRUE;
                  PAINTSCR(MANDATORY_MSG);
      }
      else if (iPT->c_d_id <= 0) {
         *pos = 4;
         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 {
            iPT->byname = TRUE;
                     done = TRUE;
         }
      }
      else
         done = TRUE;

      byname = iPT->byname;
      iPT->h_amount = int_h_amount/100.0;

      return done;
}



BOOLEAN check_ordstat_inputs (int *pos)

{
      BOOLEAN 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 {
            iOS->byname = TRUE;
            done = TRUE;
         }
      }
      else
         done = TRUE;


      byname = iOS->byname;

      return done;
}

BOOLEAN check_delivery_inputs (int *pos)
{

      BOOLEAN 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;
}

BOOLEAN check_stocklev_inputs(int *pos)
{
      BOOLEAN 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;
}


void Init_Screen()
{
      int i;
      char    buf[128];
```

```
    void setup_io_elems();

    setraw();
    for (i=0; i < MAX_FORMS; i++)
        setup_screen_buffer(&Forms[i],i);
    setup_io_elems();
    CLRSCN(buf);
    PAINTSCR(buf);
}

void Restore_Screen()
{
    restore_terminal();
}

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

void Send_Menu()
{
    if (PAINTSCR(menu_buf) == -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 = &orig_ol[i].o_ol.ol_supply_w_id;
        p++->dptr = &orig_ol[i].o_ol.ol_i_id;
        p++->dptr = &orig_ol[i].o_ol.ol_quantity;
    }

    p = Forms[PAYMENT].input_elems;
    p++->dptr = &iPT->d_id;
    p++->dptr = &iPT->c_id;
    p++->dptr = (int *) &iPT->c_last[0];
    p++->dptr = &iPT->c_w_id;
    p++->dptr = &iPT->c_d_id;
    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;
    text_elem *tbuf;
    char    *bufp;
    int     ct;
    char    blanks[] = "                          ";
    char    input_display_buf[64];
    char    fname[MAXLINE];
    io_elem *io_ptr;

    bufp = form_ptr->blank_form;
    bufp += CLRSCN(bufp);

    tbuf = form_ptr->tp;

    while ( tbuf->text ) {
      bufp += DISPLAY(bufp,tbuf->y,tbuf->x,tbuf->text);
      tbuf++;
    }

    bufp += SWITCH_TO_UNDERL(bufp);
    ct = 0;
    for (io_ptr=form_ptr->input_elems ; io_ptr->y != 999; io_ptr++) {
        strncpy(input_display_buf, blanks, io_ptr->len);
        input_display_buf[io_ptr->len]='\0';
        bufp += DISPLAY(bufp,io_ptr->x,io_ptr->y,input_display_buf);
        ct++;
    }

    form_ptr->num_input_elems = ct;

    bufp += SWITCH_TO_NORMAL(bufp);

    if (txn_type == PAYMENT)
        bufp += DISPLAY_INT(bufp,4,12,4,w_id);
    else
        bufp += DISPLAY_INT(bufp,4,12,2,w_id);
    if (txn_type == STOCKLEV)
        bufp += DISPLAY_INT(bufp,2,29,2,d_id);
    bufp += SWITCH_TO_UNDERL(bufp);

    *bufp++ = '\1';
    *bufp = '\0';
}



int read_integer(col, row, size, flags, data)
int col, row, size, flags, *data;

/** Function to read in integer data from the screen.
    col   - first column position of the data field on the screen.
    row   - row position of the data field on the screen.
    size  - the length of the data field in number of characters to read in.
    flags - boolean flag that was used to identify the mandatory data fields.
    data  - pointer to the actual location where the returning value should be
stored in.

    This function reads in the input data until either a TAB, BACKTAB,
LINEFEED,
    SUBMIT character is entered or the maximum number of characters
have been
    entered into the data field.

    The data is read as character and converted to an integer before being
    stored in the appropriate location.

 **/
{
    BOOLEAN 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, *tempbuf,temp[50],  blanks[]="        ";
    float q;
    char bsspbs[]="\033[D \033[D", erase_field[20];
    static char screen_buf[200];

    strncpy(temp, "\0", 20);
    screen_buf[0] = '\0';
    bufp = screen_buf;                      /* Position cursor at start of field */
    bufp += GOTOXY(bufp,col+size-1,row);
    PAINTSCR(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");
      }
/* BOOLEAN 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')
         {
                                                  /* Consume all integers
in buffer */
            for ( ;curbuf[curbuf_read] >= '0'
                     && curbuf[curbuf_read] <= '9' ;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))/100;
               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);
              }
              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';
             *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);
              q=(q/100);
```

```
                curbuf[curbuf_read] = '\0';
                strncpy(erase_field, blanks, size);
                erase_field[size] = '\0';
                bufp = screen_buf;
                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;
                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);

              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);
           message=TRUE;
        }
        *data = atoi(temp);
              return_status = curbuf[curbuf_read];
              curbuf[curbuf_read] = '\0';
              curbuf_read=0;
              OVERFLOW = FALSE;
           }
           else
           {
             PAINTSCR(screen_buf);
             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;
         }
        return (return_status);
}


int read_string(col, row, size, flags, data)
int col, row, size, flags;
char *data;
{
    BOOLEAN 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 blanks[]="                              ";
    char bsspbs[]="\033[4m                \033[4m", erase_field[20];
    static char screen_buf[200];

    strncpy(temp, "\0", 20);
    curbuf_consumed = 0;
    screen_buf[0] = '\0';
    bufp = screen_buf;

    bufp += GOTOXY(bufp,col,row);   /* Goto input area */
    PAINTSCR(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 = screen_buf;
*/
              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);
             }
             else
             {
               if (message == FALSE)
               {
                 bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                                       EXC_FLD_LIM_MSG);
                 bufp += BEEP(bufp);
                 PAINTSCR(screen_buf);
                 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);
           message = TRUE;
```

```
            }
            OVERFLOW = FALSE;
            temp[curbuf_consumed] = '\0';
            strcpy(data, temp);
            curbuf_consumed--;
            return_status = curbuf[curbuf_read];
        }
        else
        {
          PAINTSCR(screen_buf);
          bufp = screen_buf;
        }
    }
if (message == TRUE)
 {
       bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW,
ERASE_MSG);
       message=FALSE;
       PAINTSCR(screen_buf);
 }
   return (return_status);
}


Display_Results(int txn_type)
{

   void print_neworder_output();
   void print_payment_output();
   void print_ordstat_output();
   void print_delivery_output();
   void print_stocklev_output();

   void (*p_print_function[])() = {
                &print_neworder_output,
                &print_payment_output,
                &print_ordstat_output,
                &print_delivery_output,
                &print_stocklev_output
   };

   (p_print_function[txn_type])();
}

void print_neworder_output()
{

   struct orig_order_line_struct *orig_ol_ptr;
   struct io_order_line *ool;
   char *bufp;
   int i,r;
   double ol_amount,total_amount=0.0;

   bufp = output_screen;

   if (oNO->status == ITEM_ERROR)  {
      PAINTSCR(EXECUTION_STATUS_MSG);
      return;
   }
   else {
#ifdef DEBUG
   Clog("c_last: %s c_discount %f\n",oNO->c_last,oNO->c_discount);
#endif

      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*100.0);
      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*100.0);
      bufp += DISPLAY_FLOAT(bufp,5,74,4,oNO->d_tax*100.0);

               orig_ol_ptr = orig_ol;
      for (i = 0; i < MAX_OL; i++,orig_ol_ptr++) {
         if (orig_ol_ptr->ol_loc > 0) {
            ool = &oNO->o_ol[orig_ol_ptr->ol_loc-1];
            r = i + FIRST_OL_ROW;

            ol_amount = orig_ol_ptr->o_ol.ol_quantity * ool->i_price;
            total_amount += ol_amount;

                  bufp += DISPLAY(bufp,19,r,ool->i_name);
            bufp += DISPLAY_INT(bufp,3,51,r,ool->s_quantity);
            bufp += DISPLAY(bufp,58,r,ool->b_g);
            bufp += DISPLAY_MONEY(bufp,6,62,r,ool->i_price);
            bufp += DISPLAY_MONEY(bufp,7,71,r,ol_amount);
         }
/*
         else bufp += BLANK_LINE(bufp,FIRST_OL_ROW + i);
*/
      }
      oNO->tax_n_discount = (1.0 + oNO->w_tax + oNO->d_tax) *
           (1.0 - oNO->c_discount);
      total_amount *= oNO->tax_n_discount;

      bufp += DISPLAY_MONEY(bufp,8,70,22,total_amount);
      *bufp = '\0';
      PAINTSCR(output_screen);
   }

#ifdef DEBUG
   Clog("DBG:  Screen output chars = %d\n",(bufp - &output_screen[0]));
#endif
}

void print_payment_output()
{

   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));

   if (byname == TRUE) bufp += DISPLAY_INT(bufp,4,11,9,oPT->c_id);
   else 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*100.0);
   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);

   /**   Display the first 200 lines if Credit is BC  **/
   if (oPT->c_data[0] != NULL)
   {
      bufp += DISPLAY50(bufp,12,18,oPT->c_data);
      bufp += DISPLAY50(bufp,12,19,&oPT->c_data[50]);
      bufp += DISPLAY50(bufp,12,20,&oPT->c_data[100]);
```

```
      bufp += DISPLAY50(bufp,12,21,&oPT->c_data[150]);
   }
   if (!oPT->h_date) bufp +=
DISPLAY(bufp,MESSAGE_COL,MESSAGE_ROW-2,BAD_INPUTS);
   *bufp = '\0';
   PAINTSCR(output_screen);
#ifdef DEBUG
   Clog("DBG:  Screen output chars = %d\n",(bufp - &output_screen[0]));
#endif
}

void print_ordstat_output()
{

   struct status_order_line *sol;
   char *bufp;
   int i=0, r=8;

   bufp = output_screen;

   bufp += SWITCH_TO_NORMAL(bufp);    /* jr */
#ifdef DEBUG
Clog("order_line_count = %d\n",oOS->ol_cnt);
#endif

   if (byname == TRUE) bufp += DISPLAY_INT(bufp,4,11,3,oOS->c_id);
   else 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 = '\0';
   PAINTSCR(output_screen);
#ifdef DEBUG
   Clog("DBG:  Screen output chars = %d\n",(bufp - &output_screen[0]));
#endif
}

void print_delivery_output()
{
   char *bufp;
      PAINTSCR(DELIVERY_QUEUED_MSG);
#ifdef DEBUG
   Clog("DBG:  Screen output chars = %d\n",(bufp - &output_screen[0]));
#endif
}

void print_stocklev_output()
{
   char *bufp;

   bufp = output_screen;

   bufp += SWITCH_TO_NORMAL(bufp);    /* jr */
   bufp += DISPLAY_INT(bufp,3,12,6,oSL->low_stock);
   *bufp = '\0';
   PAINTSCR(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;
}
---------------------------------
/*************************************************************
******************
******************************** screen.e
*************************************
*************************************************************
******************/
/*
**  screen.e -- All external definitions for use of screen functions
*/
extern void    Init_Screen();
extern void    Restore_Screen();
extern void    Paint_Screen(int);
extern void    Send_Menu();
extern int     Get_Menu_Input();
extern int    Get_Form_Data();
extern void    Check_Input_Date();


#define QUIT 3
#define SUBMIT 13

-------------------------
/*************************************************************
******************
******************************** screen.h
*****************************************
*************************************************************
******************/
/*
**  screen.h -- All definitions and structures for screen I/O
*/


#include <sys/termio.h>
extern int tty_in;
extern int tty_out;


#define MAX_FORMS   5

#define MESSAGE_ROW              24
```

```
#define MESSAGE_COL                 1


#define RTE_SYNCH_CHARACTER '\1'

#define SCRBUF_LEN   1536

#define FIRST_OL_ROW    7


#define BLANK_TO_END_OF_LINE(buf)          sprintf(buf,"\033[K")
#define BLANK_TO_END_OF_SCREEN(buf)     sprintf(buf,"\033[J")
#define CLRSCN(buf)
                sprintf(buf,"\033[0m\033[2J")
/*
#define DISPLAY_INT(buf,wid,x,y,ip)
                sprintf(buf,"\033[%d\;%dH%d.1d", y, x,ip)
#define DISPLAY_MONEY(buf,wid,x,y,fp)
sprintf(buf,"\033[%d\;%dH$%#wid.2f",y,x,fp)
#define DISPLAY_FLOAT(buf,wid,x,y,fp)
sprintf(buf,"\033[%d\;%dH%#wid.2f", y, x,fp)
*/
#define DISPLAY_INT(buf,wid,x,y,ip)
                sprintf(buf,"\033[%d\;%dH%*.1d", y, x,wid,ip)
#define DISPLAY_MONEY(buf,wid,x,y,fp)
sprintf(buf,"\033[%d\;%dH$%#*.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 DISPLAY_NOW(x,y,txt)                fprintf(stdout,
"\033[%d\;%dH%s", y, x,txt)
#define PAINTSCR(buf)
                write(tty_out,buf,strlen(buf))
#define SWITCH_TO_NORMAL(buf)               sprintf(buf,"\033[0m")
#define SWITCH_TO_REVERSE(buf)              sprintf(buf,"\033[7m")
#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_LINE(buf,line)
                sprintf(buf,"\033[%d\;%1H\033[K",line);
#define BLANK_UNDERLINE(buf,x,y,txt)
sprintf(buf,"\033[4m;\033[%d;\%dH%s",y,x,txt);


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

#define CANCELLED       3
#define PREVIOUS_FIELD  4

/**
 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          127
#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 SUBMIT                            20      /**Using CTRL-T for
debugging purposes  */
#define TAB             9
#define UNDERLINE       95

#define LEAVE_SCREEN_MIN 255    /** 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;
static char curbuf[300];
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();

BOOLEAN byname;

void    Init_Screen();
void    Paint_Screen(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 */
/*    sending off to the TPM                                             */
struct orig_order_line_struct {
   struct io_order_line o_ol;
   short ol_loc;
} orig_ol[MAX_OL];
int int_h_amount;

/*  All the possible messages to print out */
static char MANDATORY_MSG[]=
   "\033[24;1H\033[0mMandatory data field!  Please enter
data.\033[K\033[4m\1";
static char INVALID_MSG[]=
   "\007\033[24;1H\033[0mAn invalid character was entered.  Please enter
again.\033[K\033[4m\1";
static char ERASE_MSG[]=
   "\033[24\;1H\033[K\033[4m";
static char MIN1DIGIT_MSG[]=
   "\033[24;1H\033[0mYou must enter atleast 1 digit.  Please
reenter.\033[4m\1";
static char BAD_INPUTS[]=
   "       #### Bad input data was entered -- Select again #### \1";
static char INCOMPLINE_MSG[]=
   "\033[24;1H\033[0mOrder line is incomplete.  Please complete the
whole line.\033[4m\1";
static char ID_OR_LAST_MSG[]=
   "\033[24;1H\033[0mYou must enter either the Last Name or the
Customer Number.\033[4m\1";
static char EXC_MAX_LFT_DEC_DGT_MSG[]=
   "\033[24;1H\033[0mMaximum digits left of decimal point already
entered.  '.' expected\033[4m\1";
static char EXC_FLD_LIM_MSG[]=
```

```
   "\007\033[24\;1H\033[0mMaximum digits already entered. Tab or
<CR> expected\033[4m\1"; /* jr */

static char EXECUTION_STATUS_MSG[]="\033[0m\033[22\;18HItem
number is not valid";
static char DELIVERY_QUEUED_MSG[]="\033[0m\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[122] = "\033[0m\033[23\;1H1 - New Order   2 - Payment
3 - Order Status   4 - Delivery   5 - Stock Level\033[24\;1H9 -
Exit\033[24\;35HChoice:\1";

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,
      10,    29,    16,    0,    0,   &read_string,
      9,    33,    4,    0,    0,   &read_integer,
      9,    54,    2,    0,    0,   &read_integer,
      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
};

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

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
};

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
};

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
};

text_elem DY_text_elem[] = {
                1,              38,             "Delivery",
                2,              1,              "Warehouse:",
                4,              1,              "Carrier Number:",
                6,              1,              "Execution Status:",
                0
};

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
};



struct form_info {
        text_elem *tp;
        char    blank_form[SCRBUF_LEN];
        io_elem *input_elems;
        int     num_input_elems;
};

char output_screen[SCRBUF_LEN];

struct form_info     Forms[MAX_FORMS] = {
    { NO_text_elem," ", neworder_inputs, 0},
    { PT_text_elem," ", payment_inputs, 0},
    { OS_text_elem," ", ordstat_inputs, 0},
    { DY_text_elem," ", delivery_inputs, 0},
    { SL_text_elem," ", stocklev_inputs, 0}
};
------------------------------

/*************************************************************
******************
***************************** monitor.c
****************************************
*************************************************************
******************/
/*
**   monitor.c -- All functions for Tuxedo call and return
*/
#include <stdio.h>
#include <stdarg.h>
#include "tpcc.h"
#include "monitor.h"
#include "screen.e"
#include <atmi.h>



int Snd_Txn_To_Monitor(int txn_type)
{

    int status;

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


    memcpy(Tpmbuf, (char *)ip,
        ilen);

    if (txn_type == DELIVERY) {
        status = tpacall(svc_names[txn_type], Tpmbuf, ilen, TPNOREPLY);
                                /* Check for ugly Tuxedo failures */
/*
        if (status == -1 && tperrno == TPESVCFAIL) {
*/
        if (status == -1 ) {
            Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
                svc_names[txn_type],tpstrerror(tperrno));
            return(TPM_ERROR);
        }
    return(tpurcode);
    }
    else {
        status = tpcall(svc_names[txn_type], Tpmbuf, ilen, &Tpmbuf,
&olen,0);
                                /* Check for ugly Tuxedo failures */
/*
        if (status == -1 && tperrno == TPESVCFAIL) {
*/
        if (status == -1 ) {
            Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
                svc_names[txn_type],tpstrerror(tperrno));
            return(TPM_ERROR);
        }
                                /* return user-defined failures */


        return(tpurcode);
    }
}


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 ((Tpmbuf=tpalloc("CARRAY",NULL,ilen)) == NULL) {
        tpmerror("tpalloc",tperrno);
        return (-1);
    }
    return (NULL);
}

Rundown_Monitor()
{

int status;


    tpfree(Tpmbuf);

    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,"TUXEDO:  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: byname: %d, w_id: %d, d_id: %d\n", iPT-
>byname,
                              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: byname: %d, c_id: %d, c_last: %s\n",
         iOS->byname,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 */

----------
/*************************************************************
******************
*********************************  monitor.e
*************************************
*************************************************************
******************/
/*
**  monitor.e -- external definitions for monitor calls -- includes some
**                              handy #defines for access to the returned
monitor data.
*/
extern int Snd_Txn_To_Monitor(int);
extern int Init_Monitor();
extern void Rundown_Monitor();

extern struct io_tpcc IO_Data;

extern char *Tpmbuf;



/*  Convenient definitions for accessing inputs and outputs */
#define oNO  (&((struct io_tpcc *) Tpmbuf)->info.neworder)
#define oPT  (&((struct io_tpcc *) Tpmbuf)->info.payment)
#define oOS  (&((struct io_tpcc *) Tpmbuf)->info.ordstat)
#define oDY  (&((struct io_tpcc *) Tpmbuf)->info.delivery)
#define oSL  (&((struct io_tpcc *) Tpmbuf)->info.stocklev)

#define ip (&IO_Data)

#define iNO (&IO_Data.info.neworder)
#define iPT (&IO_Data.info.payment)
#define iOS (&IO_Data.info.ordstat)
#define iDY (&IO_Data.info.delivery)
#define iSL (&IO_Data.info.stocklev)

----------
/*************************************************************
******************
*********************************  monitor.h
*************************************
*************************************************************
******************/
/*
** monitor.h -- All Tuxedo definitions and storage
**
*/



   long ilen;
   long olen;


char *svc_names[] = { "NEWORDER_SVC",
                "PAYMENT_SVC",
                "ORDSTAT_SVC",
                "DELIVERY_SVC",
                "STOCKLEV_SVC" };


/* The actual IO storage is allocated here as IO_Data.   The routines in
screen.c fill this up.  The data is copied into Tpmbuf to send to tuxedo.
Data is returned in Tpmbuf.  Some #defines are set up here for easy
access/naming of the input and output areas
*/


struct io_tpcc IO_Data;
char *Tpmbuf;

extern void Clog(char *, ...);

#define oNO  (&((struct io_tpcc *) Tpmbuf)->info.neworder)
#define oPT  (&((struct io_tpcc *) Tpmbuf)->info.payment)
#define oOS  (&((struct io_tpcc *) Tpmbuf)->info.ordstat)
#define oDY  (&((struct io_tpcc *) Tpmbuf)->info.delivery)
#define oSL  (&((struct io_tpcc *) Tpmbuf)->info.stocklev)

#define ip (&IO_Data)
#define iNO (&IO_Data.info.neworder)
#define iPT (&IO_Data.info.payment)
#define iOS (&IO_Data.info.ordstat)
#define iDY (&IO_Data.info.delivery)
#define iSL (&IO_Data.info.stocklev)
```

```
----------------------
/*************************************************************
******************
********************************  socket.h
*****************************************
*************************************************************
******************/
/*
**  socket.h:  Declares stuff used if the screen code is going to take over a
**           LAT line and wait for client logins
*/



#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
int tty_in;
int tty_out;
•••••
-------------------------
/*************************************************************
******************
******************************** clientlog.c
*****************************************
*************************************************************
******************/
/*
**  clientlog.c -- Routine for writing out messages form client processes -
**                                 useful for detailed error reporting and for
debugging
*/

#include <stdio.h>
#include <stdarg.h>

#define BACKTAB          2     /** Decided to use the CTRL B for now **/
#define DELETE          127
#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 RTE_SYNCH_CHARACTER '\1'
/*
#define TPCC_HOME "/usr/users/sybase/kits/tpcc"
*/

   static FILE *clientlog;
   static int Clog_open=0;

void Clog(char *fmt, ...)

{
   char tmpfname[256];
   char    fname[]="CLIENTLOG";
   va_list argp;

   if (!Clog_open) {
/*

sprintf(fname,"%s/%s.%d",getenv("TPCC_HOME"),"CLIENTLOG",getpi
d());
      sprintf(fname,"%s/CLIENTLOG.%d",TPCC_HOME,getpid());

      strcpy(fname,(getenv("TPCC_LOG")));
      strncat(fname,"/CLIENTLOG.",11);
      strcat(fname,getpid());

sprintf(fname,"%s/CLIENTLOG.%d",getenv("TPCC_HOME"),getpid());
*/
      clientlog = fopen(fname,"w");
      Clog_open = 1;
   }
   va_start(argp, fmt);
   vfprintf(clientlog, fmt, argp);
   va_end(argp);
   fflush(clientlog);
}


void SCREENlog (int *flag,char *screen)
{
   char fname[256];
   int  i,char_ct;

   if (!Clog_open) {

sprintf(fname,"%s/%s.%d",getenv("TPCC_HOME"),"CLIENTLOG",getpi
d());
      clientlog = fopen(fname,"w");
      Clog_open = 1;
   }
   fprintf(clientlog,"** %d **\n",flag);
   char_ct = 0;
   fprintf(clientlog,"SCR: ");
   for (i = 0; screen[i] != 0; char_ct++,i++) {
      switch (screen[i]) {
         case BACKTAB:   fprintf(clientlog,"<BACKTAB>"); break;
         case DELETE:    fprintf(clientlog,"<DEL>"); break;
         case ESCAPE:    fprintf(clientlog,"<ESC>"); break;
         case LF:        fprintf(clientlog,"<LF>"); break;
         case QUIT:      fprintf(clientlog,"<^C>");break;
         case SUBMIT:    fprintf(clientlog,"<CR>"); break;
         case TAB:       fprintf(clientlog,"<TAB>"); break;
         case RTE_SYNCH_CHARACTER:
                         fprintf(clientlog,"<^A>"); break;
         default:        fprintf(clientlog,"%c",screen[i]);
      }
      if (char_ct > 192) {
         char_ct = 0;
/*          fprintf(screenlog,"\n"); */
      }
   }
   fprintf(clientlog,"\n");
   fflush(clientlog);
}

void SCREENlog2 (char *screen,int len)
{
   char fname[256];
   int  char_ct;

   if (!Clog_open) {

sprintf(fname,"%s/%s.%d",getenv("TPCC_HOME"),"CLIENTLOG",getpi
d());
      clientlog = fopen(fname,"w");
      Clog_open = 1;
   }
   fprintf(clientlog,"SCR: %d chars: ",len);
   for (char_ct = 0; char_ct < len; char_ct++) {
      switch (screen[char_ct]) {
         case BACKTAB:   fprintf(clientlog,"<BACKTAB>"); break;
         case DELETE:    fprintf(clientlog,"<DEL>"); break;
         case ESCAPE:    fprintf(clientlog,"<ESC>"); break;
         case LF:        fprintf(clientlog,"<LF>"); break;
         case QUIT:      fprintf(clientlog,"<^C>");break;
         case SUBMIT:    fprintf(clientlog,"<CR>"); break;
         case TAB:       fprintf(clientlog,"<TAB>"); break;
         case RTE_SYNCH_CHARACTER:
                         fprintf(clientlog,"<^A>"); break;
         default:        fprintf(clientlog,"%c",screen[char_ct]);
      }
   }
   fprintf(clientlog,"\n");
   fflush(clientlog);
}

-------------------------
```

```
/*************************************************************
*******************
******************************** syserr.c
****************************************
*************************************************************
******************/
/*
** syserr.c -- Call for exiting program and printing and error message to
**                      the screen
*/

#include <stdio.h>

void syserr(msg)  /* print system call error message and terminate */
char *msg;
{
   extern int errno, sys_nerr;
   extern char *sys_errlist[];
   extern char tty_name[];

   fprintf(stderr, "\007ERROR: (%s) %s (%d", tty_name, msg, errno);
   if (errno > 0 && errno < sys_nerr)
      fprintf(stderr, ";%s)\n", sys_errlist[errno]);
   else
      fprintf(stderr, ")\n");
   rundown (0);
}
```

TPC-C Tuxedo Server Code

```
/*************************************************************
*******************
********************************  services.c
**************************************
*************************************************************
******************/
/*
**              services.c -- Code for each Tuxedo services.
**
*/

#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <time.h>
#include "tpcc.h"
#include "db_funcs.e"
#include <atmi.h>
#include "services.h"



int tpsvrinit(argc,argv)
int argc;
char **argv;
{

#ifdef DEBUG
   printf("DBG: In tpsvrinit\n");
#endif


   if (DBinit("tpcc") != 0) {
      printf("ERR:  Problems occurred in DBinit for database \n");
      return (-1);
   }

}


void
tpsvrdone()
 {
#if DEBUG
   printf("In tpsvrdone\n");
#endif

   DBdone();
}

NEWORDER_SVC (tbuf)
TPSVCINFO *tbuf;
{
   int status;

   IOptr = (struct io_tpcc *)tbuf->data;

#ifdef DEBUG
   printf("DBG: In NEWORDER_SVC with Customer: %d\n",
      iNO->c_id);
   printf("DBG: Type %d\n",xact_type);
#endif

                  /* Call database routine */
   status = neworder_rpc(iNO,oNO);
                  /* Check for errors */
   if (tperrno) {
      userlog ("%s",tpstrerror(tperrno));
      tpreturn(TPFAIL,status,NULL,0,0);
   }


                  /* Return to client */
   tpreturn(TPSUCCESS,status,(char *)IOptr,sizeof(struct io_tpcc),0);
}

PAYMENT_SVC (tbuf)
TPSVCINFO *tbuf;

{


   int status;

   IOptr = (struct io_tpcc *) tbuf->data;



#ifdef DEBUG
   printf("DBG: In PAYMENT_SVC with Name: %s ID:  %d\n",
      iPT->c_last,iPT->c_id);
   tperrno = 0;
#endif
                  /* Call database routine */
   if (iPT->byname)
      status = payment_byname_rpc(iPT,oPT);
   else
      status = payment_byid_rpc(iPT,oPT);


                  /* Check for errors */
   if (tperrno) {
      userlog ("%s",tpstrerror(tperrno));
      tpreturn(TPFAIL,status,NULL,0,0);
   }


                  /* Return to client */
   tpreturn(TPSUCCESS,status,(char *) IOptr,sizeof(struct io_tpcc),0);
}
ORDSTAT_SVC (tbuf)
TPSVCINFO *tbuf;

{


   int status;

   IOptr = (struct io_tpcc *) tbuf->data;
```

```
#ifdef DEBUG
   printf("DBG: In ORDSTAT_SVC with Name: %s ID: %d\n",
      iOS->c_last,iOS->c_id);
   tperrno = 0;
#endif
                  /* Call database routine */
   if (iOS->byname)
      status = ordstat_byname_rpc(iOS,oOS);
   else
      status = ordstat_byid_rpc(iOS,oOS);


                  /* Check for errors */
   if (tperrno) {
      userlog ("%s",tpstrerror(tperrno));
      tpreturn(TPFAIL,status,NULL,0,0);
   }

#ifdef DEBUG
            Slog("balance = %f\n",oOS->c_balance);
#endif


                  /* Return to client */
   tpreturn(TPSUCCESS,status,(char *) IOptr,sizeof(struct io_tpcc),0);
}

DELIVERY_SVC (tbuf)
TPSVCINFO *tbuf;

{


   int status;

   IOptr = (struct io_tpcc *) tbuf->data;



#ifdef DEBUG
   printf("DBG: In DELIVERY_SVC with carrier_no: %d\n",iDY-
>o_carrier_id);
#endif

   status = delivery_rpc(iDY,oDY);


                  /* Check for errors */
/*
   if (tperrno) {
      userlog ("%s",tpstrerror(tperrno));
      tpreturn(TPFAIL,status,NULL,0,0);
   }
*/
   tpreturn(TPSUCCESS,status,NULL,0,0);

}
STOCKLEV_SVC (tbuf)
TPSVCINFO *tbuf;
{


   int status;

   IOptr = (struct io_tpcc *) tbuf->data;



#ifdef DEBUG
   printf("DBG: In STOCKLEV_SVC with District: %d\n",
      iSL->d_id);
   tperrno = 0;
#endif

                  /* Call database routine */
   status = stocklev_rpc(iSL,oSL);
                  /* Check for errors */
   if (tperrno) {
      userlog ("%s",tpstrerror(tperrno));
      tpreturn(TPFAIL,status,NULL,0,0);
   }


                  /* Return to client */
   tpreturn(TPSUCCESS,status,(char *)IOptr,sizeof(struct io_tpcc),0);
}

------------------------
/*************************************************************
******************
*********************************  services.h
************************************
*************************************************************
******************/
/*
**    services.h -- definitions for services.c
*/


#define iNO (&IOptr->info.neworder)
#define iPT (&IOptr->info.payment)
#define iOS (&IOptr->info.ordstat)
#define iDY (&IOptr->info.delivery)
#define iSL (&IOptr->info.stocklev)

#define oNO (&IOptr->info.neworder)
#define oPT (&IOptr->info.payment)
#define oOS (&IOptr->info.ordstat)
#define oDY (&IOptr->info.delivery)
#define oSL (&IOptr->info.stocklev)

#define xact_type IOptr->type

struct io_tpcc *IOptr;

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



/*************************************************************
******************
*********************************  services.e
************************************
*************************************************************
******************/
/*
** services.e -- external definitions from services -- includes handy
#defines
**             for accesses to the IO data structure
*/


#define iNO (&IOptr->info.neworder)
#define iPT (&IOptr->info.payment)
#define iOS (&IOptr->info.ordstat)
#define iDY (&IOptr->info.delivery)
#define iSL (&IOptr->info.stocklev)

#define oNO (&IOptr->info.neworder)
#define oPT (&IOptr->info.payment)
#define oOS (&IOptr->info.ordstat)
#define oDY (&IOptr->info.delivery)
#define oSL (&IOptr->info.stocklev)

#define xact_type IOptr->type

struct io_tpcc *IOptr;
-------------------
```

```
/*************************************************************
*******************
******************************  sybase_funcs.c
*****************************
*************************************************************
*******************/
/*
**  sybase_funcs.c -- Calls to the sybase stored procedures
*/

#include <stdio.h>
#include <sys/timers.h>
#include <time.h>
#include <sybfront.h>
#include <sybdb.h>
#include "tpcc.h"
#include "sybase_funcs.h"
#include "services.h"

#ifndef TPMONITOR
#include "screen.e"
int xact_type;
#endif

DBDATETIME              syb_datetime;
DBDATETIME              syb_date;

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);
}

int DBinit()

{

   char dbname[128];
   char *p;
   int i;
   LOGINREC *login;

   dberrhandle(err_handler);
   dbmsghandle(msg_handler);
   login = dblogin();


   DBSETLUSER(login, USER);
   DBSETLPACKET(login,4096);
/*
   DBSETLCHARSET(login, getenv("CHARSET"));
*/

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

/*#ifdef DEBUG
   printf("DBG:  Completed dbopen\n");
   i = JAN_1_1992;
   printf("JAN 1 1992 = %s\n",ctime(&i));
#endif*/

   if ((p = (char *)getenv("TPCC_DBNAME")) == NULL)
      strncpy(dbname,"tpcc\0",5);
   else strcpy(dbname,p);

   /* Use the the right database */
   dbuse(dbproc, dbname);

   return OKAY;
}

DBdone() {

   dbexit();

}
#ifndef TPMONITOR
Do_DBtxn(int txn_type) {

  int status;
  struct io_tpcc *iptr;
  struct io_tpcc *optr;
  struct io_payment *ptptr;
  struct io_ordstat *osptr;

  iptr = &io_data.io.input;
  optr = &io_data.io.output;


  xact_type = iptr->type;


  switch (txn_type) {
    case NEWORDER:
      status = neworder_rpc((struct io_neworder *) &iptr->info,
                  (struct io_neworder *) &optr->info);
      break;
    case PAYMENT:
      ptptr = (struct io_payment *) &iptr->info;
      if (ptptr->byname)
        status = payment_byname_rpc((struct io_payment *) &iptr->info,
                    (struct io_payment *) &optr->info);
      else
        status = payment_byid_rpc((struct io_payment *) &iptr->info,
                    (struct io_payment *) &optr->info);

      break;
    case ORDSTAT:
      osptr = (struct io_ordstat *) &iptr->info;
      if (osptr->byname)
        status = ordstat_byname_rpc((struct io_ordstat *) &iptr->info,
                   (struct io_ordstat *) &optr->info);
      else
        status = ordstat_byid_rpc((struct io_ordstat *) &iptr->info,
                    (struct io_ordstat *) &optr->info);

      break;
    case DELIVERY:
/* If we do this without the TP monitor we have to change this so it queues
*/
      status = delivery_rpc((struct io_delivery *) &iptr->info,
                  (struct io_delivery *) &optr->info);
      break;
    case STOCKLEV:
      status = stocklev_rpc((struct io_stocklev *) &iptr->info,
                  (struct io_stocklev *) &optr->info);
      break;
```

```
    }

    return;
}

#endif /* ifndef TPMONITOR */
int neworder_rpc(struct io_neworder *INO,struct io_neworder *ONO)
{
    int         try;

    xact_type = NEWORDER;

#ifdef DEBUG
   display_xction(" *** DEBUG ***");
#endif
    for (try=0; try<MAXTRIES; try++)
    {
        if (try > 0) display_xction("Repeating NO");

        deadlock = 0;
        if (new_order_body(INO,ONO) != TRUE) break;;
        dbcancel(dbproc);
        sleep_before_retry();
    }
    if (try >= MAXTRIES) display_xction("Failed");
}

int new_order_body (struct io_neworder *INO, struct io_neworder *ONO)
{
    int i;

    DBINT retcode;

    deadlock = 0;

    if (INO->o_all_local)
        dbrpcinit(dbproc, "neworder_local", 0);
    else
        dbrpcinit(dbproc, "neworder_remote", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &(INO->w_id));
    dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, &INO->d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, &INO->c_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &INO->o_ol_cnt);

    for(i = 0; i < (int)INO->o_ol_cnt; i++)
    {
        dbrpcparam(dbproc,NULL,0,SYBINT4,-1,-1, &INO-
>o_ol[i].ol_i_id);
        if (!INO->o_all_local)
                dbrpcparam(dbproc,NULL,0,SYBINT2,-1,-1, &INO-
>o_ol[i].ol_supply_w_id);
        dbrpcparam(dbproc,NULL,0,SYBINT2,-1,-1, &INO-
>o_ol[i].ol_quantity);
    }

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

    ONO->status = OKAY;

    for (i = 0; i < (int)INO->o_ol_cnt; i++)
    {
        if (dbresults(dbproc) != SUCCEED || deadlock)
            return TRUE;
        else
        {
            dbbind(dbproc, 1, NTBSTRINGBIND,  sizeof(ONO-
>o_ol[i].i_name),
                ONO->o_ol[i].i_name);
            dbbind(dbproc, 2, FLT8BIND,       0, &ONO->o_ol[i].i_price);
            dbbind(dbproc, 3, INTBIND,      0, &ONO->o_ol[i].s_quantity);
            dbbind(dbproc, 4, NTBSTRINGBIND,  sizeof(ONO->o_ol[i].b_g),
                ONO->o_ol[i].b_g);
            if (dbnextrow(dbproc) != REG_ROW) return TRUE;
            if(ONO->o_ol[i].i_name[0] == '\0')
            {
            ONO->status = ITEM_ERROR;
 /*             display_xction("Invalid item in");
              bad_items++;  */
            }
            if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;

        }
        if (dbhasretstat(dbproc))
        {
            if ((retcode = dbretstatus(dbproc)) == -3)
            {
                deadlock = 1;
                printf("Deadlock victim:");
            }
            else if (retcode<0)
            {
                fprintf(stderr, "Unknown return status %d:", retcode);
                printf("");
            }
            return TRUE;
        }
/*
        total_amount += i_price * ol[i].quantity;
*/
    }

    if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
    dbbind(dbproc, 1, REALBIND,         0,              &ONO->w_tax);
    dbbind(dbproc, 2, REALBIND,         0,              &ONO->d_tax);
    dbbind(dbproc, 3, INTBIND,          0,              &ONO->o_id);
    dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(ONO->c_last),ONO-
>c_last);
    dbbind(dbproc, 5, REALBIND,         0,              &ONO->c_discount);
    dbbind(dbproc, 6, NTBSTRINGBIND, sizeof(ONO->c_credit),ONO-
>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, ONO->o_entry_d);

    return FALSE;
}


int payment_byid_rpc ()
{
    int try;

    xact_type = PAYMENT;


#ifdef DEBUG
   display_xction(" *** DEBUG ***");
#endif
    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("Failed");
}

int payment_byid_begin ()
{
```

```
   deadlock = 0;
   dbrpcinit(dbproc, "payment_byid", 0);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &iPT->w_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &iPT->c_w_id);
   dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &iPT->h_amount);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &iPT->d_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &iPT->c_d_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &iPT->c_id);
   return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

int payment_byname_rpc ()
{
   int try;

   xact_type = PAYMENT + 100;

#ifdef DEBUG
  display_xction(" *** DEBUG ***");
#endif

   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("Failed");
}

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

int payment_end ()
{
   if (dbsqlok(dbproc) != SUCCEED) return TRUE;
   if (dbresults(dbproc) != SUCCEED || deadlock)
                return TRUE;
   else
   {
      dbbind(dbproc,  1, INTBIND,     0, &oPT->c_id);
      dbbind(dbproc,  2, NTBSTRINGBIND, sizeof(oPT->c_last), oPT-
>c_last);
            dbbind(dbproc,  3, DATETIMEBIND,  0, &syb_datetime);
      dbbind(dbproc,  4, NTBSTRINGBIND, sizeof(oPT->w_street_1),
                                oPT->w_street_1);
      dbbind(dbproc,  5, NTBSTRINGBIND, sizeof(oPT->w_street_2),
                                oPT->w_street_2);
      dbbind(dbproc,  6, NTBSTRINGBIND, sizeof(oPT->w_city), oPT-
>w_city);
      dbbind(dbproc,  7, NTBSTRINGBIND, sizeof(oPT->w_state), oPT-
>w_state);
      dbbind(dbproc,  8, NTBSTRINGBIND, sizeof(oPT->w_zip), oPT-
>w_zip);
      dbbind(dbproc,  9, NTBSTRINGBIND, sizeof(oPT->d_street_1),
                                oPT->d_street_1);
      dbbind(dbproc, 10, NTBSTRINGBIND, sizeof(oPT->d_street_2),
                                oPT->d_street_2);
      dbbind(dbproc, 11, NTBSTRINGBIND, sizeof(oPT->d_city), oPT-
>d_city);
      dbbind(dbproc, 12, NTBSTRINGBIND, sizeof(oPT->d_state), oPT-
>d_state);
      dbbind(dbproc, 13, NTBSTRINGBIND, sizeof(oPT->d_zip), oPT-
>d_zip);

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

   }
   return FALSE;
}

int ordstat_byid_rpc()
{
   int try;

   xact_type = ORDSTAT;

#ifdef DEBUG
  display_xction(" *** DEBUG ***");
#endif

   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("Failed");
}

int order_status_byid_begin ()
```

```
{
   deadlock = 0;
   dbrpcinit(dbproc, "order_status_byid", 0);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &iOS->w_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &iOS->d_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &iOS->c_id);
   return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

int ordstat_byname_rpc ()
{
   int try;

  xact_type = ORDSTAT + 100;
#ifdef DEBUG
  display_xction(" *** DEBUG ***");
#endif

   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("Failed");
}

int order_status_byname_begin ()
{
   deadlock = 0;
   dbrpcinit(dbproc, "order_status_byname", 0);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &iOS->w_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &iOS->d_id);
   dbrpcparam(dbproc,NULL, 0, SYBCHAR, -1, strlen(iOS->c_last), iOS-
>c_last);
   return (dbrpcsend(dbproc) == SUCCEED ? FALSE : TRUE);
}

int order_status_end ()
{

   struct status_order_line temp_ol;
   int i;

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

   if (dbresults(dbproc) != SUCCEED || deadlock)
                return TRUE;
   else {
                dbbind(dbproc, 1, INTBIND,     0,
&temp_ol.ol_supply_w_id);
                dbbind(dbproc, 2, INTBIND,       0,     &temp_ol.ol_i_id);
                dbbind(dbproc, 3, INTBIND,     0,     &temp_ol.ol_quantity);
                dbbind(dbproc, 4, FLT8BIND,      0,     &temp_ol.ol_amount);
                dbbind(dbproc, 5, DATETIMEBIND,  0,
                &syb_date);
                for (i = 0; (code = dbnextrow(dbproc)) == REG_ROW &&
!deadlock ; i++)
                {
                   sybdate2date(&syb_date, temp_ol.ol_delivery_d);
          memcpy(&oOS->s_ol[i],&temp_ol,sizeof(struct
status_order_line));
                }
       oOS->ol_cnt = i ;
                if (code != NO_MORE_ROWS || deadlock) return TRUE;
   }

   if (dbresults(dbproc) != SUCCEED || deadlock)
                   return TRUE;
   else
   {
                dbbind(dbproc, 1, INTBIND,     0,     &oOS->c_id);
                dbbind(dbproc, 2, NTBSTRINGBIND, sizeof(oOS->c_last),
oOS->c_last);
                dbbind(dbproc, 3, NTBSTRINGBIND, sizeof(oOS->c_first),
oOS->c_first);
                dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(oOS-
>c_middle),oOS->c_middle);
                dbbind(dbproc, 5, FLT8BIND,      0,     &oOS->c_balance);
                dbbind(dbproc, 6, INTBIND,       0,     &oOS->o_id);
                dbbind(dbproc, 7, DATETIMEBIND,  0,
                &syb_datetime);
                dbbind(dbproc, 8, INTBIND,     0,     &oOS->o_carrier_id);
      if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
      if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
                sybdate2datetime(&syb_datetime, oOS->o_entry_d);

   }

   return FALSE;
}

int delivery_rpc ()
{


   int try,d_id, o_id[11], rollback;
   static char outbuf[1000];
   char *p;
   static int delivery_no = 1;
   static time_t t;

  xact_type = DELIVERY;
#ifdef DEBUG
  display_xction(" *** DEBUG ***");
#endif

   p = outbuf;

   p += sprintf(p,"@W_ID: %2d, O_CARRIER_ID: %3d, Queued:  %s",
      iDY->w_id, iDY->o_carrier_id, ctime(&iDY->queue_time));

   for (d_id = 1, try = 0; try < MAXTRIES; try++) {
      if (try > 0) display_xction("Resuming");
      if (delivery_body(d_id,o_id) == TRUE) {
         dbcancel(dbproc);
         sleep_before_retry();
         continue;
      }
      break;
   }
   if (try >= MAXTRIES) display_xction("Failed");

   rollback = 0;

   for (d_id=1; d_id<=10; d_id++) {
      if (o_id[d_id] != 0)
         p += sprintf(p,"D_ID: %2d, O_ID:  %5d\n", d_id, o_id[d_id]);
      else {
         p += sprintf(p,"D_ID %2d, O_ID:  No outstanding order\n",d_id);
         rollback += 1;
      }
   }

   time(&t);
   p+= sprintf (p, " Rollbacks: %d\n",rollback);
   p+= sprintf (p," End: %s\n", ctime(&t));
   printf("%s",outbuf);

}

int delivery_body (d_id,o_id)
```

```
int d_id;
int o_id[];
{

   deadlock = 0;

   dbrpcinit(dbproc, "delivery", 0);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &iDY->w_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &iDY->o_carrier_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &d_id);

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

   for (; d_id <= 10; d_id++)
   {
   if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
   dbbind(dbproc, 1, INTBIND,       0,    &o_id[d_id]);
   if (dbnextrow(dbproc) != REG_ROW || deadlock) return TRUE;
   if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;
   if (dbhasretstat(dbproc) && dbretstatus(dbproc) != 0) return TRUE;
   }

   return FALSE;

}

int stocklev_rpc()
{
   int try;

  xact_type = STOCKLEV;

#ifdef DEBUG
  display_xction(" *** DEBUG ***");
#endif
   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("Failed");
}

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

   deadlock = 0;
   dbrpcinit(dbproc, "stock_level", 0);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &iSL->w_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &iSL->d_id);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &iSL->threshold);
   if (dbrpcsend(dbproc) != SUCCEED) return TRUE;
   if (dbsqlok(dbproc) != SUCCEED) return TRUE;

   if (dbresults(dbproc) != SUCCEED || deadlock) return TRUE;
   dbbind(dbproc, 1, INTBIND,  0, &iid);
   count=0;
   while (dbnextrow(dbproc) == REG_ROW && !deadlock)
   {
                for (j=0; j<count; j++)
                   if (iid == uniq[j]) break;
                if (j >= 500)
                   display_xction("Too many rows returned by");
                else
                   uniq[count++] = iid;
   }
   if (deadlock) return TRUE;
   if (dbcanquery(dbproc) != SUCCEED || deadlock) return TRUE;

#ifdef DEBUG
  printf(" *** DEBUG *** low_stock = %d\n", oSL->low_stock);

#endif

   oSL->low_stock = count;
   return FALSE;
}

void ins_rpc()
{
   dbfcmd(dbproc,"insert into foo values(%d, 'kjhkjhkjhkjhkjh')", w_id);
   dbsqlexec(dbproc);
   dbresults(dbproc);
}

void sleep_before_retry ()
{
   sleep(irand()%3+1);
}


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


/*
** error.c:   1.1          4/30/91      19:47:32
**          Standard error handler for RungenII and supporting code
**
**          HMS [04/30/91]
*/


/* message numbers that we don't want to deal with */
#define      DUMB_MESSAGE      5701
#define      ABORT_ERROR       6104

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);
      printf("ERRNO: %d, XACT_TYPE: %d\n",errno,xact_type);
                printf("\nDB-LIBRARY Error: \n\t%s \txact_type %d\n",
                                                dberrstr(errno),
xact_type);
                if (oserr != DBNOERR)
                printf("\nO/S Error: \n\t%s \txact_type: %d\n",
                                                dboserrstr(oserr),
xact_type);

                /* exit on any error */
      display_xction("ERROR");

      return(INT_CANCEL);
}


int
msg_handler(dbproc,msgno,msgstate,severity,msgtext,servername,procna
me,line)
DBPROCESS            *dbproc;
int                  msgno;
int                  msgstate;
int                  severity;
char                 *msgtext;
char         *servername;
```

```
char          *procname;
int           line;
{

                /* changing database messages */
                if (msgno == DUMB_MESSAGE || msgno ==
ABORT_ERROR || msgno == 5703 ||
msgno == 5704 ||
        msgno == 2409)
                        return(SUCCEED);

                /* Is this a deadlock message */
                if (msgno == 1205)
                {
                        /* Set the deadlock indicator */
                        /* *((DBBOOL *) dbgetuserdata(dbproc)) =
TRUE; */
                        printf("/nDeadlock occurerd %d\n",xact_type);
              display_xction("DEADLOCK");
                        deadlock = 1;
                        return(SUCCEED);

                }

                printf("/nMSG: no %d at line %d of severity %d -\n%s",
msgno, line, severity,msgtext);
                printf("\nxact_type: %d %d\n", xact_type, deadlock);
        display_xction(servername);

                /* exit on any error */

        return (0);
}

void display_xction(msg)
char * msg;
{
   int i;

   if (xact_type > 100) xact_type -= 100;

   switch(xact_type)
   {
   case NEWORDER:
      printf("%s NEWORDER \n",msg);
      printf("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++)
      printf("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:
      printf("%s PAYMENT \n",msg);
      printf("DBG: byname: %d, w_id: %d, d_id: %d\n", iPT->byname,
            iPT->w_id,iPT->d_id);
      printf("DBG: c_last: %s ",iPT->c_last);
      printf(" c_id: %d",iPT->c_id);
      printf(" c_w_id: %d, c_d_id: %d\n",iPT->c_w_id,iPT->c_d_id);
      printf("DBG: h_amount: %f\n",iPT->h_amount);
      break;
   case ORDSTAT:
      printf("%s ORDER STATUS \n",msg);
      printf("DBG: w_id: %d, d_id: %d\n",iOS->w_id,iOS->d_id);
      printf("DBG: byname: %d, c_id: %d, c_last: %s\n",
         iOS->byname,iOS->c_id,iOS->c_last);
      break;
   case DELIVERY:
      printf("%s DELIVERY \n",msg);
      printf("DBG: w_id: %d, o_carrier_id: %d\n",iDY->w_id,iDY-
>o_carrier_id);
      break;
   case STOCKLEV:
      printf("%s STOCK LEVEL\n",msg);
      printf("DBG: w_id: %d, d_id: %d, threshold: %d\n",iSL->w_id,iSL-
>d_id,
            iSL->threshold);
      break;
   other:
      printf("DBG: Txn_type = %d is illegal\n",xact_type);
   }
}
------------------------


/*************************************************************
**********************
*******************************  sybase_funcs.h
*******************************
*************************************************************
******************/
/*
**  sybase_funcs.h -- definitions used by sybase_funcs.c
*/


/* #define PARANOID */
/* #define SORT_LINES */
#ifdef PARANOID
#define FlushRows(dbproc)       dbcanquery(dbproc)
#define Cancel(dbproc)          dbcancel(dbproc)
#else
#define FlushRows(dbproc)       {}
#define Cancel(dbproc)          {}
#endif


#define smaller(x,y) (x<y ? x : y)

#define MaxTries 5
#define SERVER          NULL
#define USER            "sa"
#define LINES_PER_CALL  15
#define MAXTRIES        1

#define JAN_1_1992 694242000

void sleep_before_retry();

int oc_status;

extern char     filler[];
/*
extern DBFLT8   balance;
extern DBSMALLINT num_ware;
*/
extern char del_tab[9];
extern char proc_name[20];
extern int max_ware;

extern struct io_tpcc *IOptr;



int err_handler();
int msg_handler();
void display_xction(char *);


int deadlock;

int     bad_items;

int code;

int o_ol_done,o_ol_now;
double tax_n_discount;
BOOLEAN commit_flag;

DBPROCESS *dbproc;
••••••••••••
----------------------
#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:   1.1          4/30/91      19:47:32
**          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 */
#define     DUMB_MESSAGE     5701
#define     ABORT_ERROR      6104

int
err_handler(dbproc, severity, errno, oserr)
  DBPROCESS *dbproc;
  int severity;
  int errno;
  int oserr;
{
            /* changing databases message */
            if (errno == DUMB_MESSAGE || errno ==
ABORT_ERROR)
                        return(INT_CANCEL);

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

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

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

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

            /* changing database messages */
            if (msgno == DUMB_MESSAGE || msgno ==
ABORT_ERROR || msgno == 5703 ||
msgno == 5704)
                        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);
}
--------------------


/*************************************************************
******************
*******************************  random.c
****************************************
*************************************************************
******************/
/*
**  random.c -- contains all the random functions needed for TPC-C
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>
#define SCO


/*#include <sys/time.h> */
#ifndef SCO
#include <timers.h>
#endif

#define DAY_AND_TIME    0
#define DAY_ONLY        1



void LastName(int, char *);
int NURandomNumber(int, int, int, int);
void MakeDate(char *,int);
void MakeAddress(char *, char *, char *, char *, char *);
int RandomNumber(int,int);
int MakeAlphaString(int, int, char *);
int MakeNumberString(int, int, char *);


typedef int           I32;    /* 32 bit integer on ALL Sun/VAX hardware.
                                 Change this for each platform. */

#define A           16807
#define M       2147483647
#define Q         127773      /* M div A */
#define R           2836      /* M mod A */

static I32      Seed = 1;       /* seed value for all functions */


double drand48();

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

```
void 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;
   char 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 MakeAlphaString(min, max, str)
   int min;
   int max;
   char str[];
   {
   static char character[] =
/***
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0
123456789"; */
     "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)
   char 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 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;
   }

void MakeDate(date, time_length)
char *date;
int time_length;


{
/*   struct timespec  time; */
#ifdef SCO
   time_t ttime;
#else
   struct timeval  time;
#endif
   struct tm  calendar_time;
  /* struct timezone tzp; /* Tareef */
   char timestr[64];
#ifdef SCO
   time(&ttime);
   calendar_time = *(localtime(&ttime));
#else
   getclock (1,&time);
   calendar_time = *(localtime(&time));
#endif
```

```
   if (time_length = DAY_ONLY)
      strftime(date, 11, "%d-%m-%Y",&calendar_time);
   else
      strftime(date, 21, "%d-%m-%Y %H:%M:%S",&calendar_time);
}


/************************************************************
*******************
*
*
*
* random -
*
*       Implements a GOOD pseudo random number generator.  This
generator
*
*       will/should? run the complete period before repeating.
*
*
*
* Copied from:
*
*       Random Numbers Generators: Good Ones Are Hard to Find.
*
*       Communications of the ACM - October 1988 Volume 31 Number 10
*
*
*
* Machine Dependencies:
*
*       I32 must be 2 ^ 31 - 1 or greater.
*
*
*
************************************************************
*******************
/

/

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

* seed - load the Seed value used in irand and drand.  Should be used
before
*
*       first call to irand or drand.
*
************************************************************
*******************
/
void
seed( val )
int     val;
{
      if ( val < 0 )
            val = abs(val);

      Seed = val;
}

/************************************************************
*******************
*                                                          *
* irand - returns a 32 bit integer pseudo random number with a period of
*
*       1 to 2 ^ 32 - 1.                                   *
*                                                          *
* parameters:                                              *
*       none.                                              *
*                                                          *
* returns:                                                 *
*       32 bit integer - defined as I32 ( see above ).     *
*                                                          *
* side effects:                                            *
*       seed get recomputed.                               *
************************************************************
******************/
I32
irand()
{
      register I32    s;      /* copy of seed */
      register I32    test;   /* test flag */
      register I32    hi;     /* tmp value for speed */
      register I32    lo;     /* tmp value for speed */

      s = Seed;

      hi = s / Q;
      lo = s % Q;

      test = A * lo - R * hi;

      if ( test > 0 )
            Seed = test;
      else
            Seed = test + M;

      return( Seed );
}

/************************************************************
*******************
*                                                          *
* drand - returns a double pseudo random number between 0.0 and 1.0.
*
*       See irand.                                         *
************************************************************
******************/
double
drand()
{

      return( (double)irand() / 2147483647.0);
}

---------------------
/************************************************************
******************
*********************************
random.e*************************************
************************************************************
******************/
/*
**  random.e -- external declarations and definitions for random.c
*/



#define DAY_AND_TIME    0
#define DAY_ONLY        1

#ifndef _RANDOM_E_
#define _RANDOM_E_

extern void LastName(int, char *);
extern int NURandomNumber(int, int, int, int);
extern void MakeDate(char *,int);
extern void MakeAddress(char *, char *, char *, char *, char *);
extern int RandomNumber(int,int);
extern int MakeAlphaString(int, int, char *);
extern int MakeNumberString(int, int, char *);

#endif /* _RANDOM_E_ */
••
----------------
```

TPC-C Application Makefile

```
# change the all: target depending on what tpcc environment you want to
build

# Generic variables
OPT = -O2
#OPT = -O2 -Kpentium
#DEBUG = -DDEBUG
#DEBUG = -g3
SHR          = -non_shared
LAT          = -DSOCKET

#NOTE: Get rid of this definition if you're running without the tpmonitor
MONITOR = -DTPMONITOR

CFLAGS = $(OPT) $(DEBUG) $(SHR) $(MONITOR) $(LAT)

#Tuxedo specific variables
TUXDIR                  = /usr/opt/tuxedo
TUXINC        = $(TUXDIR)/include
#TUXLDFLAGS     = -L$(TUXDIR)/lib
TUX_SERVICES =
DELIVERY_SVC,ORDSTAT_SVC,NEWORDER_SVC,STOCKLEV_SV
C,PAYMENT_SVC
BUILDSVR = $(TUXDIR)/bin/buildserver -v  -b shm

#ODBC specific variables
ODBC          = /usr/users/client/kits/tpcc/scocode/odbcdir
ODBCINC             = $(ODBC)/include
ODBCLIB             = $(ODBC)/lib
ODBCLDFLAGS    = -L$(ODBCLIB) -lodbc_s -lnsl_s
ODBCCFLAGS     = -I$(ODBCINC) -DVG_UNIX


#Sybase specific variables
SYBDIR                  = /sybase
SYBINC                  = $(SYBDIR)/include
SYBLIB                  = $(SYBDIR)/lib
SYBCFLAGS               = -I$(SYBINC)
SYBLDFLAGS              = -L$(SYBLIB) -lsybdb -lm -lc -ldnet_stub


#all: msodbc_noptm deliverer kill_deliverer
#all: tpcc_client mssql_server
#all: tpcc_client sybase_server
all: tpcc_client sybase_server


#Build specific variables

###############################################
#####      generic TPCC client         ######
#####      -- connects through Tuxedo   ######
###############################################
CLIENT_OBJS = tpcc.o monitor.o screen.o\
              syserr.o clientlog.o
CLIENT_CFLAGS = $(CFLAGS) -I$(TUXINC)
CLIENT_LDFLAGS =


tpcc_client: $(CLIENT_OBJS)
              buildclient -v -f '$(CLIENT_OBJS) $(CFLAGS)
$(CLIENT_LDFLAGS)' -o $@
              echo "made tpcc_client with monitor calls"

###############################################
#####    TPCC client with canned data    #####
#####    returned -- no monitor call     #####
###############################################
CLIENT_LOOP_OBJS = tpcc.o screen.o syserr.o \
                        clientlog.o loopback_funcs.o random.o
CLIENT_LOOP_CFLAGS = $(CFLAGS)
CLIENT_LOOP_LDFLAGS =

tpcc_loopback_client: $(CLIENT_LOOP_OBJS)
              cc $(CLIENT_LOOP_OBJS) -o tpcc_client
$(CLIENT_LOOP_CFLAGS)$
              echo "made tpcc_client to loopback without monitor calls"

###############################################
#####      ODBC client without monitor  ######
####       and ODBC deliverer          ######
###############################################
MSODBC_NOTPM_OBJS = tpcc.o screen.o syserr.o \
                        clientlog.o msodbc_funcs.o random.o delfifo.o
MSODBC_NOTPM_CFLAGS = $(CFLAGS)  $(ODBCCFLAGS)
MSODBC_NOTPM_LDFLAGS = $(ODBCLDFLAGS)

MSODBC_DEL_OJBS = deliverer.o delfifo.o msodbc_funcs.o random.o
MSODBC_DEL_CFLAGS = $(CFLAGS) -DDELIVERER
MSODBC_DEL_LDFLAGS = $(ODBCLDFLAGS)

msodbc_noptm: $(MSODBC_NOTPM_OBJS)
              cc $(MSODBC_NOTPM_CFLAGS) -o tpcc_client \
                        $(MSODBC_NOTPM_OBJS)
$(MSODBC_DEL_LDFLAGS)

deliverer: $(MSODBC_DEL_OJBS)
              cc $(MSODBC_DEL_CFLAGS) -o $@
$(MSODBC_DEL_OJBS)  $(MSODBC_DEL_LDFLAGS)

###############################################
#####      generic server definititions ######
###############################################
SERVER_OBJS             = services.o serverlog.o random.o
SERVER_CFLAGS       = $(CFLAGS) -DTPMONITOR -I$(TUXINC)
SERVER_LDFLAGS  = $(TUXLDFLAGS) $(SHR)

###############################################
#####      TPCC server with canned data ######
#####      returned from monitor       ######
###############################################
LOOPBACK_SVR_OBJS = $(SERVER_OBJS) loopback_funcs.o
LOOPBACK_SVR_CFLAGS = $(SERVER_CFLAGS)
LOOPBACK_SVR_LDFLAGS =  $(SHR)

tpcc_loopback_server: $(LOOPBACK_SVR_OBJS)
              $(BUILDSVR) -f '$(LOOPBACK_SVR_LDFLAGS)
$(LOOPBACK_SVR_OBJS)' \
                        -o tpcc_server -s $(TUX_SERVICES)
#                       -s $(SVR_SERVICES) -l '
$(LOOPBACK_SVR_LDFLAGS)'
              echo "#### Made tpcc_server for loopback"


###############################################
#####      server for MSoft SQLServer   ######
#####                  using ODBC client       ######
###############################################
MSODBC_SVR_OBJS = $(SERVER_OBJS) msodbc_funcs.o
MSODBC_CFLAGS =  $(SERVER_CFLAGS) $(ODBCCFLAGS)
MSODBC_LDFLAGS = $(SERVER_LDFLAGS) $(ODBCLDFLAGS)

###############################################
#####      server for MSoft SQLServer   ######
#####                  using DBLIB client      ######
###############################################
MSSQL_SVR_OJBS = $(SERVER_OBJS) mssql_funcs.o
MSSQL_CFLAGS = $(SERVER_CFLAGS) $(SYBCFLAGS )
MSSQL_LDFLAGS = $(SERVER_LDFLAGS) $(SYBLDFLAGS)

###############################################
#####      server for Sybase           ######
###############################################
SYB_SVR_OBJS = $(SERVER_OBJS) sybase_funcs.o
SYB_SVR_CFLAGS = $(SERVER_CFLAGS) $(SYBCFLAGS)
SYB_SVR_LDFLAGS =  $(SHR) $(SERVER_LDFLAGS)
$(SYBLDFLAGS)
#SYB_SVR_OJBS = $(SERVER_OBJS) sybase_funcs.o
#SYB_SVR_CFLAGS = $(SERVER_CFLAGS) $(SYBCFLAGS )
#SYB_SVR_LDFLAGS = $(SERVER_LDFLAGS) $(SYBLDFLAGS)
```

```
sybase_server: $(SYB_SVR_OBJS)
		echo $(SYB_SVR_OBJS)
#		$(BUILDSVR) -f '$(SYB_SVR_CFLAGS)
$(SYB_SVR_LDFLAGS) $(SYB_SVR_OBJS)' \
#				-s $(TUX_SERVICES)
		$(BUILDSVR) -f '$(SYB_SVR_CFLAGS)
$(SYB_SVR_OBJS)' \
				-o tpcc_server -s $(TUX_SERVICES) -l '
$(SYB_SVR_LDFLAGS)'
		echo "Made tpcc_server for sybase"


kill_deliverer: delfifo.c
		cc -O2 delfifo.c -DKILL_DELIVERER -o kill_deliverer


tpcc_server: $(SVR_OBJECTS)
		$(BUILDSVR) -f '$(SVR_CFLAGS) $(SVR_OBJECTS)' \
				-s $(TUX_SERVICES) -l '$(SVR_LDFLAGS)'

clean:
		rm *.o


deliverer.o:  deliverer.c tpcc.h db_funcs.e no_tpm.h
delfifo.o:	delfifo.c
		cc $(CFLAGS) -c $*.c
clientlog.o:
monitor.o:	tpcc.h monitor.h db_funcs.e
		cc $(CLIENT_CFLAGS) -c $*.c
loopback_funcs.o: tpcc.h loopback_funcs.c random.e screen.e no_tpm.h
#		cc $(LOOPBACK_SVR_CFLAGS) -c $*.c
msodbc_funcs.o:		tpcc.h sybase_funcs.h
#		cc $(CFLAGS) $(ODBCCFLAGS) -c $*.c
sybase_funcs.o:
		cc $(CFLAGS) $(SYB_SVR_CFLAGS) -c $*.c
random.o:
#screen.o:	tpcc.h screen.h loopback.e monitor.e
screen.o:	tpcc.h screen.h loopback.e monitor.e no_tpm.h
services.o:	tpcc.h services.c db_funcs.e
		cc $(SERVER_CFLAGS) -c $*.c
serverlog.o:
syserr.o:
tpcc.o: tpcc.c tpcc.h screen.e monitor.e
•••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
••••••
-----------------
```

Sybase Stored Procedures

```
#!/bin/sh -f

# Stored procedure for TPC-C 3.0 on SQL Server 11.0 and later
# Copyright Sybase 1995

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_id			smallint,
		@d_id			tinyint,
		@c_id			int,
		@o_ol_cnt	tinyint,

		@i_id		int = 0, @ol_qty		tinyint = 0,
		@i_id2		int = 0, @ol_qty2		tinyint = 0,
		@i_id3		int = 0, @ol_qty3		tinyint = 0,
		@i_id4		int = 0, @ol_qty4		tinyint = 0,
		@i_id5		int = 0, @ol_qty5		tinyint = 0,
		@i_id6		int = 0, @ol_qty6		tinyint = 0,
		@i_id7		int = 0, @ol_qty7		tinyint = 0,
		@i_id8		int = 0, @ol_qty8		tinyint = 0,
		@i_id9		int = 0, @ol_qty9		tinyint = 0,
		@i_id10		int = 0, @ol_qty10		tinyint = 0,
		@i_id11		int = 0, @ol_qty11		tinyint = 0,
		@i_id12		int = 0, @ol_qty12		tinyint = 0,
		@i_id13		int = 0, @ol_qty13		tinyint = 0,
		@i_id14		int = 0, @ol_qty14		tinyint = 0,
		@i_id15		int = 0, @ol_qty15		tinyint = 0
)
as

declare
		@w_tax			real,			@d_tax
				real,
		@c_last			char(16),	@c_credit	char(2),
		@c_discount		real,
		@commit_flag		tinyint,

		@i_price	real,
		@i_name			char(24),	@i_data
		char(50),

		@s_quantity		smallint,
		@s_ytd			int,
		@s_order_cnt		int,
		@s_dist			char(24),	@s_data
		char(50),
		@s_dist_01 char(24),	@s_dist_02 char(24),
		@s_dist_03 char(24),	@s_dist_04 char(24),
		@s_dist_05 char(24),	@s_dist_06 char(24),
		@s_dist_07 char(24),	@s_dist_08 char(24),
		@s_dist_09 char(24),	@s_dist_10 char(24),

		@ol_number		tinyint,	@o_id
		int,
		@o_entry_ddatetime,	@b_g			char(1)

declare	@0  tinyint, @1  tinyint, @2  tinyint, @3  tinyint,
		@4  tinyint, @5  tinyint, @6  tinyint, @7  tinyint,
		@8  tinyint, @9  tinyint, @10 tinyint,@11 tinyint,
		@12 tinyint,@13 tinyint,@14 tinyint,@15 tinyint,
		@one smallint,		@ten smallint,
		@ol_qty_smallint smallint

declare c_no_wdc CURSOR FOR
		SELECT	w_tax, d_tax, d_next_o_id,
				c_last, c_discount, c_credit
				FROM	warehouse HOLDLOCK,
					district HOLDLOCK,
					customer (index c_clu prefetch 2
mru) HOLDLOCK
				WHERE	d_w_id		= @w_id
				AND	d_id		= @d_id
				AND	w_id		= d_w_id
				AND	c_w_id		= w_id
				AND	c_d_id		= d_id
				AND	c_id		= @c_id
				FOR UPDATE OF d_next_o_id

declare c_no_is CURSOR FOR
		SELECT	i_price, i_name, i_data,
				s_quantity, s_data,
				s_ytd, s_order_cnt,				/* for
update */
				s_dist_01, s_dist_02, s_dist_03, s_dist_04,
s_dist_05,
				s_dist_06, s_dist_07, s_dist_08, s_dist_09,
s_dist_10
				FROM	item HOLDLOCK,
					stock(index s_clu prefetch 2 lru)
HOLDLOCK
				WHERE	s_w_id = @w_id
				AND	s_i_id = i_id
				AND	  i_id = @i_id
```

```
                FOR UPDATE OF s_quantity, s_ytd, s_order_cnt
begin
        select  @0=0, @1=1, @2=2, @3=3, @4=4, @5=5,
@6=6, @7=7, @8=8,
                @9=9, @10=10, @11=11, @12=12, @13=13,
@14=14, @15=15,
                @one=1, @ten=10

        begin transaction NO

        OPEN c_no_wdc
        FETCH c_no_wdc INTO
                @w_tax, @d_tax, @o_id,
                @c_last, @c_discount, @c_credit
        SELECT  @commit_flag= 1
        UPDATE  district
                SET     d_next_o_id = @o_id + 1
                WHERE CURRENT OF c_no_wdc
        CLOSE c_no_wdc

        SELECT @ol_number = @0
        while (@ol_number < @o_ol_cnt) begin
                SELECT @ol_number = @ol_number + @1

                if @ol_number < @5
                        if @ol_number < @3 begin
                                if @ol_number >= @2
        SELECT @i_id = @i_id2,  @ol_qty = @ol_qty2
                        end
                        else    if @ol_number < @4
        SELECT @i_id = @i_id3,  @ol_qty = @ol_qty3
                                else
                SELECT @i_id = @i_id4,  @ol_qty = @ol_qty4
                else if @ol_number < @9
                        if @ol_number < @7
                                if @ol_number < @6
        SELECT @i_id = @i_id5,  @ol_qty = @ol_qty5
                                else
                SELECT @i_id = @i_id6,  @ol_qty = @ol_qty6
                        else    if @ol_number < @8
        SELECT @i_id = @i_id7,  @ol_qty = @ol_qty7
                                else
                SELECT @i_id = @i_id8,  @ol_qty = @ol_qty8
                else if @ol_number < @13
                        if @ol_number < @11
                                if @ol_number < @10
        SELECT @i_id = @i_id9,  @ol_qty = @ol_qty9
                                else
                SELECT @i_id = @i_id10, @ol_qty =
@ol_qty10
                        else    if @ol_number < @12
        SELECT @i_id = @i_id11, @ol_qty = @ol_qty11
                                else
                SELECT @i_id = @i_id12, @ol_qty =
@ol_qty12
                        else if @ol_number < @15
                                if @ol_number < @14
        SELECT @i_id = @i_id13, @ol_qty = @ol_qty13
                                else
                SELECT @i_id = @i_id14, @ol_qty =
@ol_qty14
                        else
                        SELECT @i_id = @i_id15, @ol_qty =
@ol_qty15


                        OPEN c_no_is
                        FETCH c_no_is INTO
                                @i_price, @i_name, @i_data,
                                @s_quantity, @s_data,
                                @s_ytd, @s_order_cnt,
                                @s_dist_01, @s_dist_02,
@s_dist_03, @s_dist_04, @s_dist_05,
                                @s_dist_06, @s_dist_07,
@s_dist_08, @s_dist_09, @s_dist_10

                        if (@@sqlstatus != 0) begin             /* item not
found */
                                SELECT @commit_flag = 0
                                select
                /* Return to client */
                                        NULL, NULL, NULL,
NULL
                                        break
                        end

                        if @d_id < @5
                          if @d_id < @3
                                if @d_id < @2           SELECT
@s_dist = @s_dist_01
                                else                    SELECT
@s_dist = @s_dist_02
                           else if @d_id < @4       SELECT @s_dist =
@s_dist_03
                                else                    SELECT
@s_dist = @s_dist_04
                        else if @d_id < @7
                                if @d_id < @6           SELECT
@s_dist = @s_dist_05
                                else                    SELECT
@s_dist = @s_dist_06
                           else if @d_id < @9
                                 if @d_id < @8          SELECT
@s_dist = @s_dist_07
                                 else                   SELECT
@s_dist = @s_dist_08
                        else if @d_id < @10      SELECT @s_dist =
@s_dist_09
                           else                         SELECT
@s_dist = @s_dist_10

                        select @ol_qty_smallint = @ol_qty
                        if @s_quantity >= @ol_qty_smallint + @ten
                                SELECT @s_quantity =
@s_quantity - @ol_qty_smallint
                        else
                                SELECT @s_quantity =
@s_quantity - @ol_qty_smallint + 91

                        UPDATE stock set
                                s_quantity   = @s_quantity,
                                s_ytd                   = @s_ytd
+ @ol_qty,
                                s_order_cnt = @s_order_cnt +
@one
                                WHERE CURRENT OF c_no_is

                        if (patindex("%ORIGINAL%", @i_data) > 0)
and
                          (patindex("%ORIGINAL%", @s_data) > 0)
                                SELECT @b_g = "B"
                        else
                                SELECT @b_g = "G"

                        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_smallint,
                                @ol_qty * @i_price, @s_dist)

                        select
                 /* Return to client */
                                @i_name, @i_price, @s_quantity,
@b_g
                        CLOSE c_no_is
                end
```

```
        SELECT @o_entry_d = getdate()
        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_id           smallint,
        @d_id           tinyint,
        @c_id           int,
        @o_ol_cnt       tinyint,

        @i_id           int = 0, @s_w_id        smallint = 0, @ol_qty
        tinyint = 0,
        @i_id2          int = 0, @s_w_id2       smallint = 0, @ol_qty2
        tinyint = 0,
        @i_id3          int = 0, @s_w_id3       smallint = 0, @ol_qty3
        tinyint = 0,
        @i_id4          int = 0, @s_w_id4       smallint = 0, @ol_qty4
        tinyint = 0,
        @i_id5          int = 0, @s_w_id5       smallint = 0, @ol_qty5
        tinyint = 0,
        @i_id6          int = 0, @s_w_id6       smallint = 0, @ol_qty6
        tinyint = 0,
        @i_id7          int = 0, @s_w_id7       smallint = 0, @ol_qty7
        tinyint = 0,
        @i_id8          int = 0, @s_w_id8       smallint = 0, @ol_qty8
        tinyint = 0,
        @i_id9          int = 0, @s_w_id9       smallint = 0, @ol_qty9
        tinyint = 0,
        @i_id10         int = 0, @s_w_id10      smallint = 0,
@ol_qty10  tinyint = 0,
        @i_id11         int = 0, @s_w_id11      smallint = 0,
@ol_qty11  tinyint = 0,
        @i_id12         int = 0, @s_w_id12      smallint = 0,
@ol_qty12  tinyint = 0,
        @i_id13         int = 0, @s_w_id13      smallint = 0,
@ol_qty13  tinyint = 0,
        @i_id14         int = 0, @s_w_id14      smallint = 0,
@ol_qty14  tinyint = 0,
        @i_id15         int = 0, @s_w_id15      smallint = 0,
@ol_qty15  tinyint = 0
)
as

declare
        @w_tax          real,           @d_tax
                real,
        @c_last         char(16),   @c_credit   char(2),
        @c_discount     real,
        @commit_flag    tinyint,

        @i_price        real,
        @i_name         char(24),   @i_data
        char(50),

        @s_quantity     smallint,
        @s_ytd          int,
        @s_order_cnt    int,
        @s_dist         char(24),   @s_data
        char(50),
        @s_dist_01 char(24),    @s_dist_02 char(24),
        @s_dist_03 char(24),    @s_dist_04 char(24),
        @s_dist_05 char(24),    @s_dist_06 char(24),
        @s_dist_07 char(24),    @s_dist_08 char(24),
        @s_dist_09 char(24),    @s_dist_10 char(24),
        @s_remote_cnt   int,            @remote
                int,

        @ol_number      tinyint,    @o_id
        int,
        @o_entry_ddatetime,     @b_g            char(1)

declare @0 tinyint, @1 tinyint, @2 tinyint, @3 tinyint,
        @4 tinyint, @5 tinyint, @6 tinyint, @7 tinyint,
        @8 tinyint, @9 tinyint, @10 tinyint,@11 tinyint,
        @12 tinyint,@13 tinyint,@14 tinyint,@15 tinyint,
        @one smallint,          @ten smallint,
        @ol_qty_smallint smallint

declare c_no_wdc CURSOR FOR
        SELECT  w_tax, d_tax, d_next_o_id,
                c_last, c_discount, c_credit
                FROM    warehouse HOLDLOCK,
                        district HOLDLOCK,
                        customer (index c_clu prefetch 2
mru) HOLDLOCK
                WHERE   d_w_id  = @w_id
                AND     d_id    = @d_id
                AND     w_id    = d_w_id
                AND     c_w_id  = w_id
                AND     c_d_id  = d_id
                AND     c_id    = @c_id
                FOR UPDATE OF d_next_o_id

declare c_no_is CURSOR FOR
        SELECT  i_price, i_name, i_data,
                s_quantity, s_data,
                s_ytd, s_order_cnt, s_remote_cnt,       /* for
update */
                s_dist_01, s_dist_02, s_dist_03, s_dist_04,
s_dist_05,
                s_dist_06, s_dist_07, s_dist_08, s_dist_09,
s_dist_10
                FROM    item HOLDLOCK,
                        stock(index s_clu prefetch 2 lru)
HOLDLOCK
                WHERE   s_w_id = @s_w_id
                AND     s_i_id = i_id
                AND      i_id = @i_id
                FOR UPDATE OF s_quantity, s_ytd,
s_order_cnt, s_remote_cnt

begin
        select  @0=0, @1=1, @2=2, @3=3, @4=4, @5=5,
@6=6, @7=7, @8=8,
                @9=9, @10=10, @11=11, @12=12, @13=13,
@14=14, @15=15,
                @one=1, @ten=10

        begin transaction NO

        OPEN c_no_wdc
        FETCH c_no_wdc INTO
                @w_tax, @d_tax, @o_id,
                @c_last, @c_discount, @c_credit
        SELECT  @commit_flag    = 1
        UPDATE  district
                SET     d_next_o_id = @o_id + 1
                WHERE CURRENT OF c_no_wdc
        CLOSE c_no_wdc

        SELECT @ol_number = @0
        while (@ol_number < @o_ol_cnt) begin
```

```
                        SELECT @ol_number = @ol_number + @1

                        if @ol_number < @5
                                if @ol_number < @3 begin
                                        if @ol_number >= @2
                SELECT @i_id = @i_id2,  @s_w_id = @s_w_id2,  @ol_qty
= @ol_qty2
                                end
                                else            if @ol_number < @4
                SELECT @i_id = @i_id3,  @s_w_id = @s_w_id3,  @ol_qty
= @ol_qty3
                                        else
                        SELECT @i_id = @i_id4,  @s_w_id =
@s_w_id4,  @ol_qty = @ol_qty4
                        else if @ol_number < @9
                                if @ol_number < @7
                                        if @ol_number < @6
                SELECT @i_id = @i_id5,  @s_w_id = @s_w_id5,  @ol_qty
= @ol_qty5
                                        else
                        SELECT @i_id = @i_id6,  @s_w_id =
@s_w_id6,  @ol_qty = @ol_qty6
                                else            if @ol_number < @8
                SELECT @i_id = @i_id7,  @s_w_id = @s_w_id7,  @ol_qty
= @ol_qty7
                                        else
                        SELECT @i_id = @i_id8,  @s_w_id =
@s_w_id8,  @ol_qty = @ol_qty8
                        else if @ol_number < @13
                                if @ol_number < @11
                                        if @ol_number < @10
                SELECT @i_id = @i_id9,  @s_w_id = @s_w_id9,  @ol_qty
= @ol_qty9
                                        else
                        SELECT @i_id = @i_id10, @s_w_id =
@s_w_id10, @ol_qty = @ol_qty10
                                else            if @ol_number < @12
                SELECT @i_id = @i_id11, @s_w_id = @s_w_id11, @ol_qty
= @ol_qty11
                                        else
                        SELECT @i_id = @i_id12, @s_w_id =
@s_w_id12, @ol_qty = @ol_qty12
                        else if @ol_number < @15
                                if @ol_number < @14
                SELECT @i_id = @i_id13, @s_w_id = @s_w_id13, @ol_qty
= @ol_qty13
                                else
                        SELECT @i_id = @i_id14, @s_w_id =
@s_w_id14, @ol_qty = @ol_qty14
                        else
                        SELECT @i_id = @i_id15, @s_w_id =
@s_w_id15, @ol_qty = @ol_qty15

                        OPEN c_no_is
                        FETCH c_no_is INTO
                                @i_price, @i_name, @i_data,
                                @s_quantity, @s_data,
                                @s_ytd, @s_order_cnt,
@s_remote_cnt,
                                @s_dist_01, @s_dist_02,
@s_dist_03, @s_dist_04, @s_dist_05,
                                @s_dist_06, @s_dist_07,
@s_dist_08, @s_dist_09, @s_dist_10

                        if (@@sqlstatus != 0) begin             /* item not
found */
                                SELECT @commit_flag = 0
                                select
                /* Return to client */
                                        NULL, NULL, NULL,
NULL
                                        break
                        end

                        if @d_id < @5
                          if @d_id < @3
                                if @d_id < @2           SELECT
@s_dist = @s_dist_01
                                else                    SELECT
@s_dist = @s_dist_02
                           else if @d_id < @4       SELECT @s_dist =
@s_dist_03
                                else                    SELECT
@s_dist = @s_dist_04
                        else if @d_id < @7
                                if @d_id < @6           SELECT
@s_dist = @s_dist_05
                                else                    SELECT
@s_dist = @s_dist_06
                         else if @d_id < @9
                                 if @d_id < @8          SELECT
@s_dist = @s_dist_07
                                 else                   SELECT
@s_dist = @s_dist_08
                        else if @d_id < @10      SELECT @s_dist =
@s_dist_09
                         else                           SELECT
@s_dist = @s_dist_10

                        select @ol_qty_smallint = @ol_qty
                        if @s_quantity >= @ol_qty_smallint + @ten
                                SELECT @s_quantity =
@s_quantity - @ol_qty_smallint
                        else
                                SELECT @s_quantity =
@s_quantity - @ol_qty_smallint + 91

                        if (@s_w_id = @w_id)
                                SELECT     @remote = 0
                        else
                                SELECT     @remote = 1

                        UPDATE stock set
                                s_quantity   = @s_quantity,
                                s_ytd                   = @s_ytd
+ @ol_qty,
                                s_remote_cnt            =
@s_remote_cnt + @remote,
                                s_order_cnt = @s_order_cnt +
@one
                                WHERE CURRENT OF c_no_is

                        if (patindex("%ORIGINAL%", @i_data) > 0)
and
                          (patindex("%ORIGINAL%", @s_data) > 0)
                                SELECT @b_g = "B"
                        else
                                SELECT @b_g = "G"

                        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, "19000101",
@ol_qty_smallint,
                                @ol_qty * @i_price, @s_dist)

                        select
                 /* Return to client */
                                @i_name, @i_price, @s_quantity,
@b_g
                        CLOSE c_no_is
                end

                SELECT @o_entry_d = getdate()
                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_id			smallint,	@c_w_id
		smallint,
		@h_amount		float,
		@d_id			tinyint,	@c_d_id
		tinyint,
		@c_id		int
as
declare	@c_last			char(16)

declare	@w_street_1		char(20),	@w_street_2
		char(20),
		@w_city			char(20),	@w_state	char(2),
		@w_zip			char(9),	@w_name
		char(10),
		@w_ytd			float

declare	@d_street_1		char(20),	@d_street_2
		char(20),
		@d_city			char(20),	@d_state	char(2),
		@d_zip			char(9),	@d_name
		char(10),
		@d_ytd			float

declare	@c_first	char(16),	@c_middle	char(2),
		@c_street_1	char(20),	@c_street_2	char(20),
		@c_city			char(20),	@c_state	char(2),
		@c_zip			char(9),	@c_phone	char(16),
		@c_since	datetime,	@c_credit	char(2),
		@c_credit_lim		numeric(12,2),
		@c_balance	float,
		@c_discount		real,
		@c_ytd_payment		float,
		@c_payment_cnt		smallint,	@1
		smallint,
		@data1			char(250),	@data2
		char(250),
		@c_data_1	char(250),	@c_data_2	char(250)

declare @screen_data	char(200),	@today		datetime

declare c_pay_wd CURSOR FOR
		SELECT	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
		FROM	warehouse HOLDLOCK,
			district HOLDLOCK
		WHERE	d_w_id		= @w_id
		AND	d_id		= @d_id
		AND	w_id		= d_w_id
		FOR UPDATE OF w_ytd, d_ytd

declare c_pay_c CURSOR FOR
		SELECT	c_first, c_middle, c_last, c_street_1, c_street_2,
			c_city, c_state, c_zip, c_phone, c_credit,
c_credit_lim,
			c_discount, c_balance - @h_amount,
c_ytd_payment + @h_amount,
			c_payment_cnt + @1, c_since, c_data1, c_data2
		FROM	customer(index c_clu prefetch 2 mru)
HOLDLOCK
		WHERE	c_w_id		= @c_w_id
		AND	c_d_id		= @c_d_id
		AND	c_id		= @c_id
		FOR UPDATE OF c_balance, c_payment_cnt,
c_ytd_payment, c_data1, c_data2

BEGIN TRANSACTION PID
		select @1 = 1
		OPEN c_pay_wd
		FETCH c_pay_wd INTO
			@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

		OPEN c_pay_c
		FETCH c_pay_c INTO
			@c_first, @c_middle, @c_last, @c_street_1,
@c_street_2,
			@c_city, @c_state, @c_zip, @c_phone,
@c_credit, @c_credit_lim,
			@c_discount, @c_balance, @c_ytd_payment,
@c_payment_cnt,
			@c_since, @data1, @data2

		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),convert(numeric(18,2), @h_amount))) +
				substring(@data1, 1, 208),
			@screen_data = substring(@c_data_1, 1, 200)

			UPDATE customer SET
				c_payment_cnt = @c_payment_cnt,
				c_ytd_payment = @c_ytd_payment,
				c_balance = @c_balance,
				c_data1 = @c_data_1, c_data2 =
@c_data_2
			WHERE CURRENT OF c_pay_c
		end
		else begin
			UPDATE	customer SET
				c_payment_cnt		=
@c_payment_cnt,
				c_balance	= @c_balance,
				c_ytd_payment		=
@c_ytd_payment
				WHERE CURRENT OF c_pay_c
		end
		CLOSE c_pay_c

		/* Create the history record */
		SELECT @today = getdate()
		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, @c_d_id, @w_id,
                @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_id                   smallint,       @c_w_id
        smallint,
        @h_amount               float,
        @d_id                   tinyint,        @c_d_id
        tinyint,
        @c_last     char(16)
as
declare     @n          int,        @c_id       int

declare     @w_street_1             char(20),   @w_street_2
        char(20),
        @w_city                 char(20),   @w_state    char(2),
        @w_zip                  char(9),    @w_name
        char(10),
        @w_ytd                  float

declare     @d_street_1             char(20),   @d_street_2
        char(20),
        @d_city                 char(20),   @d_state    char(2),
        @d_zip                  char(9),    @d_name
        char(10),
        @d_ytd                  float

declare     @c_first    char(16),   @c_middle   char(2),
        @c_street_1 char(20),   @c_street_2 char(20),
        @c_city                 char(20),   @c_state    char(2),
        @c_zip                  char(9),    @c_phone    char(16),
        @c_since    datetime,   @c_credit   char(2),
        @c_credit_lim           numeric(12,2),
        @c_balance  float,
        @c_discount             real,
        @c_ytd_payment          float,
        @c_payment_cnt          smallint,   @1
        smallint,
        @data1                  char(250),  @data2
        char(250),
        @c_data_1   char(250),  @c_data_2   char(250)

declare @screen_data    char(200),  @today      datetime

declare c_pay_wd CURSOR FOR
        SELECT      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
        FROM        warehouse HOLDLOCK,
                    district HOLDLOCK
        WHERE       d_w_id      = @w_id
        AND         d_id        = @d_id
        AND         w_id        = d_w_id
        FOR UPDATE OF w_ytd, d_ytd

declare c_pay_c CURSOR FOR
        SELECT      c_first, c_middle, c_last, c_street_1, c_street_2,
                    c_city, c_state, c_zip, c_phone, c_credit,
c_credit_lim,
                    c_discount, c_balance - @h_amount,
c_ytd_payment + @h_amount,
                    c_payment_cnt + @1, c_since, c_data1, c_data2
        FROM        customer(index c_clu prefetch 2 mru)
HOLDLOCK
        WHERE       c_w_id      = @c_w_id
        AND         c_d_id      = @c_d_id
        AND         c_id        = @c_id
        FOR UPDATE OF c_balance, c_payment_cnt,
c_ytd_payment, c_data1, c_data2

declare c_find CURSOR FOR
        SELECT  c_id
        FROM customer (index c_non1 prefetch 2 mru) 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 mru) HOLDLOCK
        WHERE       c_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
        select @1 = 1
        OPEN c_pay_wd
        FETCH c_pay_wd INTO
                    @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

        OPEN c_pay_c
        FETCH c_pay_c INTO
                    @c_first, @c_middle, @c_last, @c_street_1,
@c_street_2,
                    @c_city, @c_state, @c_zip, @c_phone,
@c_credit, @c_credit_lim,
                    @c_discount, @c_balance, @c_ytd_payment,
@c_payment_cnt,
```

```
                @c_since, @data1, @data2

        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),convert(numeric(18,2), @h_amount))) +
                        substring(@data1, 1, 208),
                @screen_data = substring(@c_data_1, 1, 200)

                UPDATE customer SET
                        c_payment_cnt = @c_payment_cnt,
                        c_ytd_payment = @c_ytd_payment,
                        c_balance = @c_balance,
                        c_data1 = @c_data_1, c_data2 =
@c_data_2
                WHERE CURRENT OF c_pay_c
        end
        else begin
                UPDATE  customer SET
                        c_payment_cnt           =
@c_payment_cnt,
                        c_balance   = @c_balance,
                        c_ytd_payment           =
@c_ytd_payment
                        WHERE CURRENT OF c_pay_c
        end
        CLOSE c_pay_c

        /* Create the history record */
        SELECT @today = getdate()
        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, @c_d_id, @w_id,
                @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_id           smallint,
        @d_id           tinyint,
        @c_id           int
as

DECLARE  @o_id          int,
        @o_entry_ddatetime,
        @o_carrier_id   smallint

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
        FROM    orders (index o_clu prefetch 16 lru)
HOLDLOCK
        WHERE   o_w_id  = @w_id
        AND     o_d_id  = @d_id
        AND     o_c_id  = @c_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
        FROM    order_line HOLDLOCK
        WHERE   ol_o_id = @o_id
        AND     ol_d_id = @d_id
        AND     ol_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
        FROM    customer(index c_clu prefetch 2 mru)
HOLDLOCK
        WHERE   c_id    = @c_id
        AND     c_d_id  = @d_id
        AND     c_w_id  = @w_id

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

DECLARE  @o_id          int,
        @o_entry_ddatetime,
        @o_carrier_id   smallint

declare     @n          int,        @c_id       int
declare c_find CURSOR FOR
        SELECT  c_id
        FROM customer (index c_non1 prefetch  2 mru)
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 mru) HOLDLOCK
        WHERE   c_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
        FROM    orders (index o_clu prefetch 16 lru)
HOLDLOCK
        WHERE   o_w_id      = @w_id
        AND     o_d_id      = @d_id
        AND     o_c_id      = @c_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
        FROM    order_line HOLDLOCK
        WHERE   ol_o_id = @o_id
        AND     ol_d_id = @d_id
        AND     ol_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
        FROM    customer(index c_clu prefetch 2 mru)
HOLDLOCK
        WHERE   c_id    = @c_id
        AND     c_d_id  = @d_id
        AND     c_w_id  = @w_id

COMMIT TRANSACTION OSNM
go

use tpcc
go

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

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

declare @no_o_id      int,          @o_c_id       smallint,
    @ol_total     float,        @ol_amount    float,
    @junk_id      smallint,
    @today        datetime,     @1            smallint,
    @one          tinyint,      @ten          tinyint

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.
    */

declare c_del_ol CURSOR FOR
    SELECT  ol_amount
    FROM    order_line HOLDLOCK
    WHERE   ol_o_id = @no_o_id
    AND     ol_d_id = @d_id
    AND     ol_w_id = @w_id
    FOR UPDATE OF ol_delivery_d

declare c_del_o CURSOR FOR
    SELECT  o_c_id, o_carrier_id
    FROM    orders HOLDLOCK
    WHERE   o_id    = @no_o_id
    AND     o_d_id  = @d_id
    AND     o_w_id  = @w_id
    FOR UPDATE OF o_carrier_id

begin
  select @1 = 1, @one = 1, @ten = 10

  while (@d_id <= @ten) 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
    end
    else
    begin
        DELETE  FROM new_order
        WHERE   CURRENT OF c_del_no
        CLOSE   c_del_no

        SELECT  @ol_total = 0.0, @today = getdate()
        OPEN    c_del_ol
        FETCH   c_del_ol INTO @ol_amount

        while (@@sqlstatus = 0)
        begin
            SELECT  @ol_total = @ol_total + @ol_amount
            UPDATE  order_line
            SET     ol_delivery_d = @today
            WHERE   CURRENT OF c_del_ol
            FETCH   c_del_ol INTO @ol_amount
        end
        CLOSE c_del_ol

        OPEN    c_del_o
        FETCH   c_del_o INTO @o_c_id, @junk_id
        UPDATE  orders
        SET     o_carrier_id = @o_carrier_id
        WHERE   CURRENT OF c_del_o
        CLOSE   c_del_o

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

        COMMIT TRANSACTION DEL

        select      /* Return to client */
                @no_o_id
    end
    select @d_id = @d_id + @one
  end
end
go

if exists ( SELECT name FROM sysobjects WHERE name = 'stock_level')
```

```
   DROP PROC stock_level
go

CREATE PROC stock_level
                @w_id       smallint,
                @d_id       tinyint,
                @threshold smallint
as
                select  s_i_id
                FROM        district,
                            order_line (index ol_clu prefetch 2 mru),
                            stock (index s_clu prefetch 2 lru)
                WHERE       d_w_id      =           @w_id
                AND         d_id        =           @d_id
                AND         ol_w_id     =           @w_id
                AND         ol_d_id     =           @d_id
                AND         ol_o_id     between (d_next_o_id - 20) and
(d_next_o_id - 1)
                AND         s_w_id      =           ol_w_id
                AND         s_i_id      =           ol_i_id
                AND     s_quantity < @threshold
go
EOF

--------------------
```

Appendix B

Sybase Device Init and Database Create Code

```
disk init
  name = 'orders01',
  physname = '/tpcc_devs/orders01',
  vdevno = 1,
  size = 51200
go
disk init
  name = 'orders02',
  physname = '/tpcc_devs/orders02',
  vdevno = 2,
  size = 51200
go
disk init
  name = 'orders03',
  physname = '/tpcc_devs/orders03',
  vdevno = 3,
  size = 51200
go
disk init
  name = 'orders04',
  physname = '/tpcc_devs/orders04',
  vdevno = 4,
  size = 51200
go
disk init
  name = 'history01',
  physname = '/tpcc_devs/history01',
  vdevno = 5,
  size = 358400
go
disk init
  name = 'c_idx01',
  physname = '/tpcc_devs/c_idx01',
  vdevno = 6,
  size = 128000
go
disk init
  name = 'tpcc_log',
  physname = '/tpcc_devs/tpcc_log',
  vdevno = 7,
  size = 512000
go
disk init
  name = 'order_line01',
  physname = '/tpcc_devs/order_line01',
  vdevno = 8,
  size = 409600
go
disk init
  name = 'order_line02',
  physname = '/tpcc_devs/order_line02',
  vdevno = 9,
  size = 409600
go
disk init
  name = 'order_line03',
  physname = '/tpcc_devs/order_line03',
  vdevno = 10,
  size = 409600
go
disk init
  name = 'order_line04',
  physname = '/tpcc_devs/order_line04',
  vdevno = 11,
  size = 409600
go
disk init
  name = 'stock01',
  physname = '/tpcc_devs/stock01',
  vdevno = 12,
  size = 116224
go
disk init
  name = 'stock02',
  physname = '/tpcc_devs/stock02',
  vdevno = 13,
  size = 116224
go
disk init
  name = 'stock03',
  physname = '/tpcc_devs/stock03',
  vdevno = 14,
  size = 116224
go
disk init
  name = 'stock04',
  physname = '/tpcc_devs/stock04',
  vdevno = 15,
  size = 116224
go
disk init
  name = 'stock05',
  physname = '/tpcc_devs/stock05',
  vdevno = 16,
  size = 116224
go
disk init
  name = 'stock06',
  physname = '/tpcc_devs/stock06',
  vdevno = 17,
  size = 116224
go
disk init
  name = 'stock07',
  physname = '/tpcc_devs/stock07',
  vdevno = 18,
  size = 116224
go
disk init
  name = 'stock08',
  physname = '/tpcc_devs/stock08',
  vdevno = 19,
  size = 116224
go
disk init
  name = 'stock09',
  physname = '/tpcc_devs/stock09',
  vdevno = 20,
  size = 116224
go
disk init
  name = 'stock10',
  physname = '/tpcc_devs/stock10',
  vdevno = 21,
  size = 116224
go
disk init
  name = 'stock11',
  physname = '/tpcc_devs/stock11',
  vdevno = 22,
  size = 116224
go
disk init
  name = 'stock12',
  physname = '/tpcc_devs/stock12',
  vdevno = 23,
  size = 116224
go
disk init
  name = 'stock13',
  physname = '/tpcc_devs/stock13',
  vdevno = 24,
  size = 116224
go
```

```
disk init
  name = 'stock14',
  physname = '/tpcc_devs/stock14',
  vdevno = 25,
  size = 116224
go
disk init
  name = 'stock15',
  physname = '/tpcc_devs/stock15',
  vdevno = 26,
  size = 116224
go
disk init
  name = 'stock16',
  physname = '/tpcc_devs/stock16',
  vdevno = 27,
  size = 116224
go
disk init
  name = 'stock17',
  physname = '/tpcc_devs/stock17',
  vdevno = 28,
  size = 116224
go
disk init
  name = 'stock18',
  physname = '/tpcc_devs/stock18',
  vdevno = 29,
  size = 116224
go
disk init
  name = 'stock19',
  physname = '/tpcc_devs/stock19',
  vdevno = 30,
  size = 116224
go
disk init
  name = 'stock20',
  physname = '/tpcc_devs/stock20',
  vdevno = 31,
  size = 116224
go
disk init
  name = 'stock21',
  physname = '/tpcc_devs/stock21',
  vdevno = 32,
  size = 204800
go
disk init
  name = 'customer01',
  physname = '/tpcc_devs/customer01',
  vdevno = 33,
  size = 122880
go
disk init
  name = 'customer02',
  physname = '/tpcc_devs/customer02',
  vdevno = 34,
  size = 122880
go
disk init
  name = 'customer03',
  physname = '/tpcc_devs/customer03',
  vdevno = 35,
  size = 122880
go
disk init
  name = 'customer04',
  physname = '/tpcc_devs/customer04',
  vdevno = 36,
  size = 122880
go
disk init
  name = 'customer05',
  physname = '/tpcc_devs/customer05',
  vdevno = 37,
  size = 122880
go
disk init
  name = 'customer06',
  physname = '/tpcc_devs/customer06',
  vdevno = 38,
  size = 122880
go
disk init
  name = 'customer07',
  physname = '/tpcc_devs/customer07',
  vdevno = 39,
  size = 122880
go
disk init
  name = 'customer08',
  physname = '/tpcc_devs/customer08',
  vdevno = 40,
  size = 122880
go
disk init
  name = 'customer09',
  physname = '/tpcc_devs/customer09',
  vdevno = 41,
  size = 122880
go
disk init
  name = 'customer10',
  physname = '/tpcc_devs/customer10',
  vdevno = 42,
  size = 122880
go
disk init
  name = 'customer11',
  physname = '/tpcc_devs/customer11',
  vdevno = 43,
  size = 122880
go
disk init
  name = 'customer12',
  physname = '/tpcc_devs/customer12',
  vdevno = 44,
  size = 204800
go
create database tpcc
on master = 500, orders01 = 100, orders02 = 100, orders03 = 100,
orders04 = 100,  history01 = 700, c_idx01 = 250, order_line01 = 800,
order_line02 = 800, order_line03 = 800, order_line04 = 800, stock01 =
227, stock02 = 227, stock03 = 227, stock04 = 227, stock05 = 227, stock06
= 227, stock07 = 227, stock08 = 227, stock09 = 227, stock10 = 227,
stock11 = 227, stock12 = 227, stock13 = 227, stock14 = 227, stock15 =
227, stock16 = 227, stock17 = 227, stock18 = 227, stock19 = 227, stock20
= 227, stock21 = 400, customer01 = 240, customer02 = 240, customer03 =
240, customer04 = 240, customer05 = 240, customer06 = 240, customer07
= 240, customer08 = 240, customer09 = 240, customer10 = 240,
customer11 = 240, customer12 = 400
log on tpcc_log = 1000
go
use tpcc
go
sp_addsegment Scache , tpcc , master
go
sp_addsegment Scidx , tpcc , c_idx01
go
sp_addsegment Scustomer , tpcc , customer01
go
sp_extendsegment Scustomer , tpcc ,  customer02
go
sp_extendsegment Scustomer , tpcc ,  customer03
go
sp_extendsegment Scustomer , tpcc ,  customer04
go
sp_extendsegment Scustomer , tpcc ,  customer05
go
sp_extendsegment Scustomer , tpcc ,  customer06
go
sp_extendsegment Scustomer , tpcc ,  customer07
go
```

```
sp_extendsegment Scustomer , tpcc ,  customer08
go
sp_extendsegment Scustomer , tpcc ,  customer09
go
sp_extendsegment Scustomer , tpcc ,  customer10
go
sp_extendsegment Scustomer , tpcc ,  customer11
go
sp_extendsegment Scustomer , tpcc ,  customer12
go
sp_addsegment Shistory , tpcc , history01
go
sp_addsegment Sorder_line , tpcc , order_line01
go
sp_extendsegment Sorder_line , tpcc ,  order_line02
go
sp_extendsegment Sorder_line , tpcc ,  order_line03
go
sp_extendsegment Sorder_line , tpcc ,  order_line04
go
sp_addsegment Sorders , tpcc , orders01
go
sp_extendsegment Sorders , tpcc ,  orders02
go
sp_extendsegment Sorders , tpcc ,  orders03
go
sp_extendsegment Sorders , tpcc ,  orders04
go
sp_addsegment Sstock , tpcc , stock01
go
sp_extendsegment Sstock , tpcc ,  stock02
go
sp_extendsegment Sstock , tpcc ,  stock03
go
sp_extendsegment Sstock , tpcc ,  stock04
go
sp_extendsegment Sstock , tpcc ,  stock05
go
sp_extendsegment Sstock , tpcc ,  stock06
go
sp_extendsegment Sstock , tpcc ,  stock07
go
sp_extendsegment Sstock , tpcc ,  stock08
go
sp_extendsegment Sstock , tpcc ,  stock09
go
sp_extendsegment Sstock , tpcc ,  stock10
go
sp_extendsegment Sstock , tpcc ,  stock11
go
sp_extendsegment Sstock , tpcc ,  stock12
go
sp_extendsegment Sstock , tpcc ,  stock13
go
sp_extendsegment Sstock , tpcc ,  stock14
go
sp_extendsegment Sstock , tpcc ,  stock15
go
sp_extendsegment Sstock , tpcc ,  stock16
go
sp_extendsegment Sstock , tpcc ,  stock17
go
sp_extendsegment Sstock , tpcc ,  stock18
go
sp_extendsegment Sstock , tpcc ,  stock19
go
sp_extendsegment Sstock , tpcc ,  stock20
go
sp_extendsegment Sstock , tpcc ,  stock21
go
use tpcc
go
sp_dropsegment 'default', tpcc , c_idx01
go
sp_dropsegment 'system', tpcc , c_idx01
go
sp_dropsegment 'default', tpcc , customer01
go
sp_dropsegment 'system', tpcc , customer01
go
sp_dropsegment 'default', tpcc , customer02
go
sp_dropsegment 'system', tpcc , customer02
go
sp_dropsegment 'default', tpcc , customer03
go
sp_dropsegment 'system', tpcc , customer03
go
sp_dropsegment 'default', tpcc , customer04
go
sp_dropsegment 'system', tpcc , customer04
go
sp_dropsegment 'default', tpcc , customer05
go
sp_dropsegment 'system', tpcc , customer05
go
sp_dropsegment 'default', tpcc , customer06
go
sp_dropsegment 'system', tpcc , customer06
go
sp_dropsegment 'default', tpcc , customer07
go
sp_dropsegment 'system', tpcc , customer07
go
sp_dropsegment 'default', tpcc , customer08
go
sp_dropsegment 'system', tpcc , customer08
go
sp_dropsegment 'default', tpcc , customer09
go
sp_dropsegment 'system', tpcc , customer09
go
sp_dropsegment 'default', tpcc , customer10
go
sp_dropsegment 'system', tpcc , customer10
go
sp_dropsegment 'default', tpcc , customer11
go
sp_dropsegment 'system', tpcc , customer11
go
sp_dropsegment 'default', tpcc , customer12
go
sp_dropsegment 'system', tpcc , customer12
go
sp_dropsegment 'default', tpcc , history01
go
sp_dropsegment 'system', tpcc , history01
go
sp_dropsegment 'default', tpcc , order_line01
go
sp_dropsegment 'system', tpcc , order_line01
go
sp_dropsegment 'default', tpcc , order_line02
go
sp_dropsegment 'system', tpcc , order_line02
go
sp_dropsegment 'default', tpcc , order_line03
go
sp_dropsegment 'system', tpcc , order_line03
go
sp_dropsegment 'default', tpcc , order_line04
go
sp_dropsegment 'system', tpcc , order_line04
go
sp_dropsegment 'default', tpcc , orders01
go
sp_dropsegment 'system', tpcc , orders01
go
sp_dropsegment 'default', tpcc , orders02
go
sp_dropsegment 'system', tpcc , orders02
go
sp_dropsegment 'default', tpcc , orders03
go
```

```
sp_dropsegment 'system', tpcc , orders03
go
sp_dropsegment 'default', tpcc , orders04
go
sp_dropsegment 'system', tpcc , orders04
go
sp_dropsegment 'default', tpcc , stock01
go
sp_dropsegment 'system', tpcc , stock01
go
sp_dropsegment 'default', tpcc , stock02
go
sp_dropsegment 'system', tpcc , stock02
go
sp_dropsegment 'default', tpcc , stock03
go
sp_dropsegment 'system', tpcc , stock03
go
sp_dropsegment 'default', tpcc , stock04
go
sp_dropsegment 'system', tpcc , stock04
go
sp_dropsegment 'default', tpcc , stock05
go
sp_dropsegment 'system', tpcc , stock05
go
sp_dropsegment 'default', tpcc , stock06
go
sp_dropsegment 'system', tpcc , stock06
go
sp_dropsegment 'default', tpcc , stock07
go
sp_dropsegment 'system', tpcc , stock07
go
sp_dropsegment 'default', tpcc , stock08
go
sp_dropsegment 'system', tpcc , stock08
go
sp_dropsegment 'default', tpcc , stock09
go
sp_dropsegment 'system', tpcc , stock09
go
sp_dropsegment 'default', tpcc , stock10
go
sp_dropsegment 'system', tpcc , stock10
go
sp_dropsegment 'default', tpcc , stock11
go
sp_dropsegment 'system', tpcc , stock11
go
sp_dropsegment 'default', tpcc , stock12
go
sp_dropsegment 'system', tpcc , stock12
go
sp_dropsegment 'default', tpcc , stock13
go
sp_dropsegment 'system', tpcc , stock13
go
sp_dropsegment 'default', tpcc , stock14
go
sp_dropsegment 'system', tpcc , stock14
go
sp_dropsegment 'default', tpcc , stock15
go
sp_dropsegment 'system', tpcc , stock15
go
sp_dropsegment 'default', tpcc , stock16
go
sp_dropsegment 'system', tpcc , stock16
go
sp_dropsegment 'default', tpcc , stock17
go
sp_dropsegment 'system', tpcc , stock17
go
sp_dropsegment 'default', tpcc , stock18
go
sp_dropsegment 'system', tpcc , stock18
go
sp_dropsegment 'default', tpcc , stock19
go
sp_dropsegment 'system', tpcc , stock19
go
sp_dropsegment 'default', tpcc , stock20
go
sp_dropsegment 'system', tpcc , stock20
go
use master
go
checkpoint
go
---------------------------
```

Sybase Table and Index Definition

```
#!/bin/sh -f

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_id                    smallint,
        w_name                  char(10),
        w_street_1      char(20),
        w_street_2      char(20),
        w_city                  char(20),
        w_state                 char(2),
        w_zip                   char(9),
        w_tax                   real,
        w_ytd                   float                   /*-
Updated by PID, PNM */
) on Scache
go

if exists ( select name from sysobjects where name = 'district' )
        drop table district
go
create table district (
        d_id                    tinyint,
        d_w_id                  smallint,
        d_name                  char(10),
        d_street_1      char(20),
        d_street_2      char(20),
        d_city                  char(20),
        d_state                 char(2),
        d_zip                   char(9),
        d_tax                   real,
        d_ytd                   float,                  /*-
Updated by PID, PNM */
        d_next_o_id             int                     /*-
Updated by NO */
) on Scache
go

if exists ( select name from sysobjects where name = 'customer' )
        drop table customer
go
create table customer (
        c_id                    int,
        c_d_id                  tinyint,
        c_w_id                  smallint,
        c_first                 char(16),
        c_middle        char(2),
```

```
        c_last                  char(16),
        c_street_1      char(20),
        c_street_2      char(20),
        c_city                  char(20),
        c_state                 char(2),
        c_zip                   char(9),
        c_phone                 char(16),
        c_since                 datetime,
        c_credit        char(2),
        c_credit_lim            numeric(12,2),
        c_discount      real,
        c_delivery_cnt          smallint,
        c_payment_cnt           smallint,       /*- Updated by PNM,
PID */
        c_balance       float,                  /*- Updated by PNM,
PID */
        c_ytd_payment           float,                  /*-
Updated by PNM, PID */
        c_data1                 char(250),      /*- Updated (?) by
PNM, PID */
        c_data2                 char(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_id                  int,
        h_c_d_id        tinyint,
        h_c_w_id        smallint,
        h_d_id                  tinyint,
        h_w_id                  smallint,
        h_date                  datetime,
        h_amount        float,
        h_data                  char(24)
) on Shistory
go
/* alter table history unpartition */
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_id                 int,
        no_d_id                 tinyint,
        no_w_id                 smallint,
) on Scache
go
create unique clustered index no_clu
        on new_order(no_w_id, no_d_id, no_o_id)
        on Scache
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips, 100, new_order)
go

if exists ( select name from sysobjects where name = 'orders' )
        drop table orders
go
create table orders (
        o_id                    int,
        o_c_id                  int,
        o_d_id                  tinyint,
        o_w_id                  smallint,
        o_entry_d       datetime,
        o_carrier_id    smallint,       /*- Updated by D */
        o_ol_cnt        tinyint,
        o_all_local     tinyint
) 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_id                 int,
        ol_d_id                 tinyint,
        ol_w_id                 smallint,
        ol_number       tinyint,
        ol_i_id                 int,
        ol_supply_w_id          smallint,
        ol_delivery_d           datetime,       /*- Updated by D */
        ol_quantity     smallint,
        ol_amount       float,
        ol_dist_info char(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_id                    int,
        i_im_id                 int,
        i_name                  char(24),
        i_price                 float,
        i_data                  char(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_id                  int,
        s_w_id                  smallint,
        s_quantity      smallint,       /*- Updated by NO */
        s_ytd                   int,                    /*-
Updated by NO */
        s_order_cnt     smallint,       /*- Updated by NO */
        s_remote_cnt            smallint,       /*- Updated by NO */
        s_dist_01       char(24),
        s_dist_02       char(24),
        s_dist_03       char(24),
        s_dist_04       char(24),
        s_dist_05       char(24),
        s_dist_06       char(24),
        s_dist_07       char(24),
        s_dist_08       char(24),
        s_dist_09       char(24),
```

```
        s_dist_10     char(24),
        s_data                  char(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
-------------------------
```

Sybase Cache Binding

```
#!/bin/sh -f

isql -Usa -P$PASSWORD << 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", "sysindexes"
go
sp_bindcache "c_tinyhot", "tpcc", "sysindexes.sysindexes"
go

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

use tpcc
go
checkpoint
go

/*
** Cache c_tinyhot (continued)
*/

sp_bindcache "c_tinyhot", "tpcc", "item"
go
sp_bindcache "c_tinyhot", "tpcc", "item.i_clu"
go
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

/*
** Cache c_no_ol
*/

sp_bindcache "c_no_ol", "tpcc", "new_order"
go
sp_bindcache "c_no_ol", "tpcc", "new_order.no_clu"
go
sp_bindcache "c_no_ol", "tpcc", "order_line"
go

/*
** Cache c_ol_index
*/

sp_bindcache "c_ol_index", "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_customer_index", "tpcc", "customer.c_clu"
go
sp_bindcache "c_customer_index", "tpcc", "customer.c_non1"
go

EOF
----------------------
```

Sybase Data Load

```
typedef     unsigned long           BitVector;
#define WSZ                         (sizeof(BitVector)*8)
/* For axposf use WAREBATCH 0f 150 */
#define     WAREBATCH               150
#define     nthbit(map,n)           map[(n)/WSZ] &
(((BitVector)0x1)<< ((n)%WSZ))
#define     setbit(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()
{
        int             i = 1;

        bulk_w = bulk_open("tpcc", "warehouse", password);

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


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


end_warehouse_load()
{
        bulk_close(bulk_w);
}



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

District

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


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

```
int bulk_d;

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

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

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

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

                        MakeAlphaString(6, 10, d_name);
                        MakeAddress(d_street_1,
d_street_2, d_city, d_state, d_zip);
                        d_tax = RandomNumber(10,20) /
100.0;

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



begin_district_load()
{
        int             i = 1;

        bulk_d = bulk_open("tpcc", "district", password);

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

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

end_district_load()
{
        bulk_close(bulk_d);
}



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

Item

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



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

int bulk_i;

LoadItems()
{

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

        printf("Loading items\n");

        begin_item_load();

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

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

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

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

         item_load();
         }

        end_item_load();
        return;
}



begin_item_load()
{
        int             i = 1;

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

        /* 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()
{
        int             i = 1;

        bulk_h = bulk_open("tpcc", "history", password);

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

history_load()
{

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


end_history_load()
{
        bulk_close(bulk_h);
}




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

Customer

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

/* static variables containing fields for customer record */
ID c_id;
ID c_d_id;
ID c_w_id;
TEXT c_first[16+1];
TEXT c_middle[2+1] = "OE";
TEXT c_last[16+1];
TEXT c_street_1[20+1];
TEXT c_street_2[20+1];
TEXT c_city[20+1];
TEXT c_state[2+1];
TEXT c_zip[9+1];
TEXT c_phone[16+1];
DATE c_since;
TEXT c_credit[2+1] = "?C";
MONEY c_credit_lim = 50000.0;       /* 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];
ID              len;
```

```
int bulk_c;


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

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

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

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

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

        LastName(i<1000?i:NURandomNumber(255,123,0,999),c_la
st);
                        MakeAlphaString(8, 16, c_first);

        MakeAddress(c_street_1,c_street_2,c_city,c_state,c_zip);
                        MakeNumberString(16, 16,
c_phone);
                        MakeAlphaString(300, 500, c_data);
                        datetime(&c_since);
                        c_credit[0] = nthbit(badcredit[d_id-
1],i) ? 'B' : 'G';
                        c_discount = RandomNumber(0, 50)
/ 100.0;
                        c_balance = -10.0;
                        customer_load();
                }
        }
}

begin_customer_load()
{
        int             i = 1;

        bulk_c = bulk_open("tpcc", "customer", password);

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

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


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

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

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

end_customer_load()
{
        bulk_close(bulk_c);
}



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

Order, Order line, New order

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

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

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

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


int  o_bulk;
int ol_bulk;
int no_bulk;



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

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

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

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

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

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

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

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

        RandomPermutation(cust, ORD_PER_DIST);


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

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

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

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

		o_bulk = bulk_open("tpcc", "orders", password);

		bulk_bind(o_bulk, i++, "o_id", &o_id, ID_T);
		bulk_bind(o_bulk, i++, "o_c_id", &o_c_id, ID_T);
		bulk_bind(o_bulk, i++, "o_d_id", &o_d_id, ID_T);
		bulk_bind(o_bulk, i++, "o_w_id", &o_w_id, ID_T);
		bulk_bind(o_bulk, i++, "o_entry_d", &o_entry_d, DATE_T);
		bulk_bind(o_bulk, i++, "o_carrier_id", &o_carrier_id, ID_T);
		bulk_bind(o_ol_cnt, i++, "o_ol_cnt", &o_ol_cnt,
COUNT_T);
		bulk_bind(o_all_local, i++, "o_all_local", &o_all_local,
LOGICAL_T);
}

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

end_order_load()
{
		bulk_close(o_bulk);
}




begin_order_line_load()
{
		int		i = 1;

		ol_bulk = bulk_open("tpcc", "order_line", password);

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

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


end_order_line_load()
{
		bulk_close(ol_bulk);
}

begin_new_order_load()
{
		int		i = 1;

		no_bulk = bulk_open("tpcc", "new_order", password);

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

new_order_load()
{
		debug("  no_o_id=%d  \n", no_o_id);
		bulk_load(no_bulk);
}


end_new_order_load()
{
		bulk_close(no_bulk);
}


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

Stock

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


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

int bulk_s;

/*
** On loading stock in major order of item_id:
** 10% of the MAXITEMS items in each warehouse need to marked as
original
** (i.e., s_data like '%ORIGINAL%'.)  This is a bit harder to do when we
** load by item number, rather than by warehouses.  The trick is to first
** generate a huge WAREBATCH * MAXITEMS bitmap, initialize all
bits to zero,
```

```
** and then set 10% of bits in each row to 1.  While loading item i in
** warehouse w, we simply lookup bitmap[w][i] to see whether it needs to
** be marked as original.
*/

LoadStock(w1, w2)
        ID w1, w2;
{
        ID w_id;
        BitVector original[WAREBATCH][((MAXITEMS+(WSZ-
1))/WSZ)], * bmp;
        int w, i, j;


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

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

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

begin_stock_load()
{
        int             i = 1;

        bulk_s = bulk_open("tpcc", "stock", password);

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



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


end_stock_load()
{
        bulk_close(bulk_s);
}



test(){}


getargs(argc, argv)

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

        /* define the defaults */
```

```
        load_item = load_warehouse = load_district = load_history =
        load_orders = load_new_order = load_order_line =
        load_customer = load_stock = NO;


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

        }

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

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

        /* Check if warehouse is within the range */
        if (w1 <= 0 || w2 > 1000 || 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[] =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0
123456789";
        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;
}
------------------
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED

#include <sybfront.h>
#include <sybdb.h>
#include <time.h>


/* Population constants */
#ifdef CACHED
#define MAXITEMS 10000
#define CUST_PER_DIST 300
#define DIST_PER_WARE 10
#define ORD_PER_DIST 300
#else
#define MAXITEMS 100000
#define CUST_PER_DIST 3000
#define DIST_PER_WARE 10
#define ORD_PER_DIST 3000
#endif

/* Types of application variables */
typedef int     COUNT;
typedef int     ID;
typedef double  MONEY;
typedef double  FLOAT;
typedef char    TEXT;
typedef struct { int x[2];} DATE;
typedef int     LOGICAL;

typedef enum
   {COUNT_T, ID_T, MONEY_T, FLOAT_T, TEXT_T,
    DATE_T, LOGICAL_T, MAX_T}
   DATA_TYPE;

typedef struct timeval TIME;

#define YES 1
#define NO 0
#define EOF (-1)

#ifndef NULL
#define NULL ((void *)0)
#endif

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

/* define function types */
extern      int         msg_handler();
extern      int         err_handler();
extern      int         batch_size;

#endif /* TPCC_INCLUDED */
••••••••••••••••••
----------------------
---------------
/*************************************************************
*******
*************************************************************
********

Sybase Specific Routines

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

#include <stdio.h>
#include <sys/time.h>
#include <string.h>
#include "loader.h"

datetime(date)
        DBDATETIME *date;
{
```

```
	struct timeval time;
	gettimeofday(&time, NULL);
	date->dtdays = time.tv_sec / (60*60*24)
			+ (1970-1900)*365 + (1970-
1900)/4;
	date->dttime = (time.tv_sec % (60*60*24))*300
			+ time.tv_usec*300/1000000;
}


/* define the type information for each field */
typedef struct
{
	char *terminator;
	int termlen;
	int type;
} bind_parm;

bind_parm parm[MAX_T] =
{
	/* COUNT */		{NULL, 0, SYBINT4},
	/* ID	*/		{NULL, 0, SYBINT4},
	/* MONEY */		{NULL, 0, SYBFLT8},
	/* FLOAT */		{NULL, 0, SYBFLT8},
	/* TEXT */  {"",   1, SYBCHAR},
	/* DATE */  {NULL, 0, SYBDATETIME},
	/* LOGICAL */		{NULL, 0, SYBINT4}
};

#define MAXOPENS 10

DBPROCESS *dbproc[MAXOPENS];
int		count[MAXOPENS];

int bulk_open(database, table, password)
	char database[];
	char table[];
	char password[];
{
	LOGINREC *login;
	int db;

	/* make note we have established a connection */
	for (db=0; db<MAXOPENS; db++)
		if (dbproc[db] == NULL) break;
	count[db] = 0;

	/* Install an error and Message handler */
	dbmsghandle(msg_handler);
	dberrhandle(err_handler);

	/* initialize dblib */
	if (dbinit() != SUCCEED)
		printf("Can't initialize the DB library\n");

	/* allocate a login record and fill it in */
	login = dblogin();
	if (login == NULL)
		printf("Can't allocate a login record.\n");
	DBSETLUSER(login, "sa");

	if(strlen(password) > 0)
	DBSETLPWD(login, password);

	DBSETLAPP(login, table);
	BCP_SETL(login, TRUE);

	/* Set Packet Size to 4096 */
	DBSETLPACKET(login, 4096);

	/* establish a connection with the server specified by
DSQUERY */
	dbproc[db] = dbopen(login, NULL);
	if (dbproc[db] == NULL)
		printf("Can't establish connection. Is DSQUERY
set?\n");

	/* select the database to use */
	if (database != NULL)
		if (dbuse(dbproc[db], database) != SUCCEED)
			printf("Can't select database: %s\n",
database);

	/* release the login record */
	dbloginfree(login);

	/* prepare to do a bulk copy */
	if (bcp_init(dbproc[db], table, NULL, NULL, DB_IN) !=
SUCCEED)
	printf("Can't initialize the bulk copy to table %s\n", table);

	return db;
}



bulk_bind(db, column, name, address, type)
	int db;
	int column;
	char name[];
	void *address;
	int type;
{
	if (bcp_bind(dbproc[db], address, 0, -1,
parm[type].terminator,
			parm[type].termlen,
parm[type].type, column) != SUCCEED)
		printf("Can't bind column %d to 0x%x,
type=%d\n",
			column,address,type);
}

bulk_null(db, column)
	int db;
	int column;
{
	if (bcp_collen(dbproc[db], 0, column) != SUCCEED)
		printf("Can't null column %d\n", column);
}


bulk_non_null(db, column)
	int db;
	int column;
{
	if (bcp_collen(dbproc[db], -1, column) != SUCCEED)
		printf("Can't non-null column %d\n", column);
}


bulk_load(db)
	int db;
{
	count[db]++;
	if (bcp_sendrow(dbproc[db]) != SUCCEED)
		printf("bulk_load: Can't load row\n");
	if (count[db]%batch_size == 0 && (bcp_batch(dbproc[db])
== -1))
	printf("bulk_load: Can't post rows\n");
	if (count[db]%1000 == 0) write(1,".",1);
	if (count[db]%50000 == 0) write(1,"\n",1);

}

bulk_close(db)
	int db;
{
	if (bcp_done(dbproc[db]) == -1)
		printf("Problems completing the bulk copy.\n");
	dbproc[db] = NULL;
	if (count[db] >= 1000) write(1,"\n",1);
}
••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
---------------------
```

Appendix C

RTE Transaction Driver Code

```
!
!                    COPYRIGHT @ 1991
!              DIGITAL EQUIPMENT CORPORATION, MAYNARD
!               MASSACHUSETTS. ALL RIGHTS RESERVED.
!
! Name:          DID.SCR (User device control scripts)
!
! Description:
!      This is the device (teller in TPC-A) script.  Think of this as what
!      each teller would do.  Each teller logs into SUT, get into ACMS,
!      then start doing transactions.
!
! Usage:
!
! Author:        TNSG/Software Performance Group
!
! Creation Date:   Sep. 1, 1991
!
! Limitations:
!
! Revision History:
!   Chen, Yongmin      Sep. 1, 1991
!            Clean up of scripts to remove unused codes and put in
!            more documentation to make it more readable.
!
!*************************************************************
******************
!
!*********************************  DID
*************************************
!
! All users start with this script. They all started with a PAUSE,
! waiting for TPC script to singal continuation.  Each user then
! log into the system and start application (ACMS or USER AGENT),
! do a transaction and PAUSE again.  After they woke up by the TPC
! script the second time, transactions will be generated and send
! to SUT by calling TPCA_TRANSACTION.SCR until the end of the test.
!
!*************************************************************
******************
!
SCRIPT DID
!
DECLARE LOCAL TEMP_STORAGE
                COPY %DID %R10
                MUL DID$_SEED %R10
      COPY %R10 TEMP_STORAGE
      MOD 2 TEMP_STORAGE
      IF TEMP_STORAGE EQ 0 THEN SUB 1 %R10
!SEND CONSOLE %R10
                !
!               CALL INIT_RAND_SEED(%R10)
      COPY %DID %R1
      ADD TPC$_DISTRICT_OFFSET %R1      ! offset for slave RTEs
      SUB 1 %R1
                COPY %R1 %R2                          ! %R1 WAREHOUSE
1-???
      DIV 10 %R1
                ADD 1 %R1
      MOD 10 %R2                    ! %R2 DISTRICT 1-10
                ADD 1 %R2
                !
LABEL DID_START
                PAUSE
!               ENABLE PROMPT "<01>"
                IGNORE
!               PURGE       !   The TPC script will resume this process when
it is ready.
                !
                GOSUB DID_START
                !
      COPY %DID TEMP_STORAGE
      MOD 25 TEMP_STORAGE
!       IF TEMP_STORAGE EQ 0 THEN SEND CONSOLE "Device ready
and waiting..."
                !
                IF TPC$_OPERATING_SYSTEM EQ 0 THEN GOSUB
TPCC_TRANSACTION
                IF TPC$_OPERATING_SYSTEM EQ 1 THEN GOSUB
TPCC_TRANSACTION_OSF
                !
                !
LABEL DID_RUN_DONE
                DELAY 3
                IF TPC$_OPERATING_SYSTEM EQ 1 THEN JUMP
DID_RUN_DONE_OSF
                !
                SEND @CR
                DELAY 2
                SEND @CR
                DELAY 2
                SEND "<11>$EXIT"
                SEND @CR
                SEND "<11>"
                PURGE
                SUB 1 USERS_LOGGED_IN
                JUMP DID_RUN_DONE_END
LABEL DID_RUN_DONE_OSF
                DELAY 3
                !
                SEND "9"
                DELAY 2
                SEND "logout"
                SEND @CR
                PURGE
                SUB 1 USERS_LOGGED_IN
LABEL DID_RUN_DONE_END
END
!
!
SCRIPT DID_START
                !
                ! Script DID_START.SCR
                !
                !       This script handles logging in a user on a ANY terminal
device.
                !       Terminal characteristics must be explicitly set after login.
                !
                !       The device will log in to the front-end processor only.
                !       The SUT number to log in is calculated as:
                !
                !       (%DID mod #front-ends)
                !
                !       If distributed ACMS is not being used, the number of
front-ends
                !       should be set to equal the total number of SUTs.
                !
                !       This assumes that the front-end SUTs are numbered:
                !       SUT1, SUT2, ..., SUT#front-ends
                !
                !       This assumes that the back-end SUTs are numbered:
                !       SUT#front-ends, SUT#front-ends+1, ..., SUTn
                !
                !**************************  SCRIPT START
*****************************
                !
LABEL START_LOGIN
!               PURGE
                !
                !
                ! Call NEWPLOGIN to log the user in. NEWPLOGIN expects
the following
                !  registers to contain the following:
                !
                !       %R4 - the device class
                !       %R5 - the string index of the main SUT name. For TPC-
A, this
```

```
                !           is "SUT"
                !       %R6 - the relative SUT number. Concatenated with the
SUT name
                !           to form the actual SUT name - e.g., SUT1, SUT2,...
                !       %R7 - the string index of the VMS username
                !       %R8 - the string index of the VMS password
                !
                ! Use device class 2
                !
!               COPY %DID %R6
!               MOD TPC$_NUM_FRONT_END %R6
!               ADD 1 %R6
                COPY TPC$_CUR_FRONT_END %R6
                !
                COPY 2 %R4
                !
                ! The string SUT contains "SUT"
                !
                COPY SUT %R5
                COPY DID$_USERNAME %R7
                COPY DID$_PASSWORD %R8
                GOSUB NEWPLOGIN
                !
                ! NEWPLOGIN does a RESTORE. Negate it by doing
IGNORE.
                !
                IGNORE
                !
                !           %R9 contains return status from NEWPLOGIN.
If 0, then it failed.
                !           If 1, then it's a hard-wired line; if 2, then it's a
LAT line.
                !
                !           If failure, send a message to the console, then
pause. If we
                !           get resumed from the console, then try again.
                !
                IF %R9 NE 0 THEN JUMP LOGIN_OK
                !
                SEND CONSOLE "DEVICE FAILED TO LOGIN"
                SEND CONSOLE "DEVICE PAUSING - RESUME TO
RETRY LOGIN"
                PAUSE
                JUMP START_LOGIN
                !
LABEL LOGIN_OK
                IF TPC$_OPERATING_SYSTEM EQ 0 THEN JUMP
LOGIN_OK_VMS
                IF TPC$_OPERATING_SYSTEM EQ 1 THEN JUMP
LOGIN_OK_OSF
                !
LABEL LOGIN_OK_VMS
                !   Set terminal characteristics explicitly for a LAT terminal
                !
                IF %R9 NE 2 THEN JUMP NO_LAT_SETUP_VMS
                CLASS 2
!               DISABLE PROMPT "<01>"
                ENABLE PROMPT "$"
                SEND "SET
TERMINAL/TTSYNC/HOSTSYNC/NOBROAD/PERM/VT100/NOAUT
OBAUD"
                SEND @CR
                MODE OUTPUT
                DISABLE PROMPT "$"
LABEL NO_LAT_SETUP_VMS
                !
                ENABLE PROMPT ":"
                SEND "ACMS/ENTER/NORETURN<11>"
                SEND @CR
                MODE OUTPUT
                CLASS 1
                !
                ADD 1 USERS_LOGGED_IN
                JUMP EOF
!
!
LABEL LOGIN_OK_OSF
                !   Set terminal characteristics explicitly for a LAT terminal
                !
                CLASS 1
!               DISABLE PROMPT "<01>"
                !
                SEND "exec tpcc_client "
                SEND %R1
                SEND " "
                SEND %R2
                ENABLE PROMPT "<01>"
                SEND @CR
                MODE OUTPUT
                !
                ADD 1 USERS_LOGGED_IN

LABEL EOF

END
-----------------------
!************************************************************
**********************
!* COPYRIGHT (C) 1993 BY                                    *
!* Digital Equipment Corporation, Maynard, Massachusetts.   *
!* All Rights Reserved.                                     *
!************************************************************
**********************
!
!
!               TPCC_TRANSACTION.SCR
!             This is the TPC-C Transaction Script
!
!**************************DESCRIPTION*********************
**************
!
! This script, which is called from the DID script, causes the
! device to fill in the fields on the input stream to be sent to
! SUT for processing, and record the response time.
!
! In order to fill in the fields, this script generates the input by
! calling each individual external routine for each transaction type.
!
! Current warehouse number and district numbers are determined in
another
! script and stored in the device specific registers listed below:
!           %R1 - warehouse number
!           %R2 - district number
!
! RESPONSE .. EVENT is used to collect response time, this also allows
for
! the log of one variable along with the response time pairs.  Device
specific
! register, R12, is used to log additional data.  For more detail, see the
! corresponding external routines for documentation.
!
! The encoding for each MENU response time is as follows:
!  MENU:        XXXXXXXXXXXX3210987654321
!                            || 006 NEW ORDER (  6   06)
!                            ||| PAYMENT      ( 14   016)
!                            || || ORDER STATUS  ( 22   026)
!                            |||| DELIVERY     ( 30   036)
!                            | || STOCK LEVEL   ( 38   046)
!
!**********************************SCRIPT
START*********************************
!
SCRIPT TPCC_TRANSACTION_OSF
!
DECLARE LOCAL next_txn

!
!EXTERNAL ROUTINE get_no_data(value, value, value, value,
reference, descriptor, reference)
!EXTERNAL ROUTINE get_pt_data(value, value, value, value, reference,
descriptor, reference)
!EXTERNAL ROUTINE get_os_data(value, value, reference, descriptor,
reference)
!EXTERNAL ROUTINE get_dy_data(reference, descriptor, reference)
```

```
!EXTERNAL ROUTINE get_sl_data(reference, descriptor, reference)
!EXTERNAL ROUTINE NEW_RAND_EXPO
(REFERENCE,REFERENCE,REFERENCE,REFERENCE)
!
 IGNORE
 PURGE
!
 DISABLE PROMPT ":"
 ENABLE PROMPT "<01>"
!
!   start the TPC-C transaction
!
 IF TPC$TEST_TERMINAL_RESP EQ 1 THEN GOSUB
TEST_TERMINAL_RESPONSE
 GOSUB TpccStartup_osf
 PAUSE
 CLASS 3
 PURGE
!
! The maximum value allowed for %RANGE variable in VAXRTE is 256.
By assigning
! 112, 111, 11, 11, 11 for the five transaction types, roughly 43.8%, 43.4%,
! 4.3%, 4.3%, and 4.3% should be achieved with reasonably long enough
test.
!
COPY 243 %RANGE
!
LABEL MAIN_LOOP
 RANDOM %R10 next_txn
 ENABLE PROMPT "<01>"
 CASE next_txn
   0:107    GOSUB NewOrderTxn_OSF
   108:212  GOSUB PaymentTxn_OSF
   213:222  GOSUB OrderStatusTxn_OSF
   223:232  GOSUB DeliveryTxn_OSF
   233:242  GOSUB StockLevelTxn_OSF
   OTHER    SEND CONSOLE "impossible case"
!
 IF EXPERIMENT_DONE EQ 0 THEN JUMP MAIN_LOOP
 CLASS 1
!
END           ! TPCC_TRANSACTION
!

SCRIPT NewOrderTxn_OSF
!
! This script submits a NEW-ORDER transaction and collects response
time.
!
 RESPONSE "<01>" EVENT 6
 SEND "1"
 MODE OUTPUT
!
 COPY FORM_IDX(%DID) %R5
!
!   call external routine to get input data
!
 CALL
get_no_data(%R1,TPC$_NUM_WAREHOUSES,C_ID_CONST,ITEM_C
ONST,%R10,%R5,%R12)
!
!
!   Delay for 0.35 seconds to compensate for delay of transmitting menu
data
!   for NEW-ORDER txn through DECserver at 38.4 K.
!8-Jul-1995 No more terminal server delay
!  DELAY GRAINS 7
 SEND @%R5
!
!   Delay of 18 seconds for keying time
!
 DELAY GRAINS 361
!
 RESPONSE "<01>" EVENT %R12
 SEND @CR
 MODE OUTPUT
!
 CALL NEW_RAND_EXPO(%R10, NO_THINK_TIME, %R11,%DID)
!
!   Delay for additional 0.40 seconds to compensate for delay of
transmitting
!   New Order output data through DECserver at 38.4 K in addition to the
!   think time delay.
!
!8-Jul-1995 No more terminal server delay
 DELAY GRAINS %R11
!
END                          ! NewOrderTxn
!

SCRIPT PaymentTxn_OSF
!
! This script submits a PAYMENT transaction and collectes response time.
!
 RESPONSE "<01>" EVENT 14
 SEND "2"
 MODE OUTPUT
!
 COPY FORM_IDX(%DID) %R5
!
!   call external routine to get input data
!
 CALL get_pt_data(%R1, TPC$_NUM_WAREHOUSES, C_ID_CONST,
C_LAST_CONST, %R10, %R5, %R12)
!
!
!   Delay for 0.20 seconds to compensate for delay of transmitting menu
data
!   for PAYMENT txn through DECserver at 38.4 K.
!
!8-Jul-1995 No more terminal server delay
!  DELAY GRAINS 4
 SEND @%R5
!
!   Delay of 3 seconds for keying time
!
 DELAY GRAINS 61
!
 RESPONSE "<01>" EVENT %R12
 SEND @CR
 MODE OUTPUT
!
 CALL NEW_RAND_EXPO (%R10,PT_THINK_TIME,%R11,%DID)
!
!   Delay for additional 0.30 seconds to compensate for delay of
transmitting
!   Payment output data through DECserver at 38.4 K in addition to the
!   think time delay.
!
!8-Jul-1995 No more terminal server delay
 DELAY GRAINS %R11
!
END                                  ! PaymentTxn
!

SCRIPT OrderStatusTxn_OSF
!
! This script submits a ORDER-STATUS transaction and collectes
response time.
!
 RESPONSE "<01>" EVENT 22
 SEND "3"
 MODE OUTPUT
!
 COPY FORM_IDX(%DID) %R5
!
!   call external routine to get input data
!
 CALL get_os_data(C_ID_CONST, C_LAST_CONST, %R10, %R5,
%R12)
!
!
!   Delay for 0.20 seconds to compensate for delay of transmitting menu
data
```

```
!    for ORDER-STATUS txn through DECserver at 38.4 K.
!
!8-Jul-1995 No more terminal server delay
!  DELAY GRAINS 4
  SEND @%R5
!
!    Delay of 2 seconds for keying time
!
  DELAY GRAINS 41
!
  RESPONSE "<01>" EVENT %R12
  SEND @CR
  MODE OUTPUT
!
  CALL NEW_RAND_EXPO(%R10, OS_THINK_TIME, %R11,%DID)
!
!    Delay for additional 0.40 seconds to compensate for delay of
transmitting
!    Order Status output data through DECserver at 38.4 K in addition to the
!    think time delay.
!
!8-Jul-1995 No more terminal server delay
  DELAY GRAINS %R11
!
END                                     ! OrderStatusTxn
!

SCRIPT DeliveryTxn_OSF
!
! This script submits a DELIVERY transaction and collects response time.
!
  RESPONSE "<01>" EVENT 30
  BREAK
  SEND "4"
!  MODE OUTPUT
  COPY FORM_IDX(%DID) %R5
!
!    call external routine to get input data
!
  CALL get_dy_data(%R10, %R5, %R12)
!
!
!    Delay for 0.15 seconds to compensate for delay of transmitting menu
data
!    for DELIVERY txn through DECserver at 38.4 K.
!
!8-Jul-1995 No more terminal server delay
!  DELAY GRAINS 3
  SEND @%R5
!
!    Delay of 2 seconds for keying time
!
  DELAY GRAINS 41
!
  RESPONSE "<01>" EVENT %R12
  BREAK
  SEND @CR
!  MODE OUTPUT
!
  CALL NEW_RAND_EXPO(%R10, DY_THINK_TIME, %R11,%DID)
!
!    Delay for additional 0.20 seconds to compensate for delay of
transmitting
!    Delivery output data through DECserver at 38.4 K in addition to the
!    think time delay.
!
!8-Jul-1995 No more terminal server delay
  DELAY GRAINS %R11
!
END                                     ! DeliveryTxn
!

SCRIPT StockLevelTxn_OSF
!
! This script submits a STOCK-LEVEL transaction and collectes response
time.
!
!
  RESPONSE "<01>" EVENT 38
  BREAK
  SEND "5"
!
!  MODE OUTPUT
  COPY FORM_IDX(%DID) %R5
!
!    call external routine to get input data
!
  CALL get_sl_data(%R10, %R5, %R12)
!
!
!    Delay for 0.10 seconds to compensate for delay of transmitting menu
data
!    for STOCK-LEVEL txn through DECserver at 38.4 K.
!
!8-Jul-1995 No more terminal server delay
!  DELAY GRAINS 2
  SEND @%R5
!
!    Delay of 2 seconds for keying time
!
  DELAY GRAINS 41
!
  RESPONSE "<01>" EVENT %R12
  BREAK
  SEND @CR
!  MODE OUTPUT
!
  CALL NEW_RAND_EXPO(%R10, SL_THINK_TIME, %R11,%DID)
!
!    Delay for additional 0.15 seconds to compensate for delay of
transmitting
!    Stock Level output data through DECserver at 38.4 K in addition to the
!    think time delay.
!
!8-Jul-1995 No more terminal server delay
  DELAY GRAINS %R11
!
end                                     ! StockLevelTxn

SCRIPT TpccStartup_OSF
!
! Start the TPC-C transaction by selecting the TPC-C task, which in turn
will
! do an ORDER-STATUS txn to initialize the warehouse and district id.
!
   PURGE
  SEND "3"
!
!  RESPONSE "<01>" EVENT 38
!  SEND @CR
  MODE OUTPUT
!
  COPY FORM_IDX(%DID) %R5
  CALL get_os_data(C_ID_CONST, C_LAST_CONST, %R10, %R5,
%R12)
!          send string
!SEND LOGFILE @%R5
  SEND @%R5
  ENABLE PROMPT "<01>"
  SEND @CR
  MODE OUTPUT
  DELAY 5
!
!   SEND "12                567"
!   SEND @CR
!   MODE OUTPUT
!   DELAY 5
!
!
!
END             ! TpccStartup

SCRIPT TEST_TERMINAL_RESPONSE
```

```
DECLARE LOCAL terminal_loop_ct

!Does a new-order screen, entering 1 character at a time and measuring the
!response time for the echo.  The 1 character is "7".

  PURGE
  SEND "1"
  MODE OUTPUT

!District 7
  ENABLE PROMPT "7"
  RESPONSE "7" EVENT 38
  SEND "7"
  MODE OUTPUT
  SEND @TAB

!Customer 123
  ENABLE PROMPT "1"
  RESPONSE "1" EVENT 38
  SEND "1"
  MODE OUTPUT
  ENABLE PROMPT "2"
  RESPONSE "2" EVENT 38
  SEND "2"
  MODE OUTPUT
  ENABLE PROMPT "3"
  RESPONSE "3" EVENT 38
  SEND "3"
  MODE OUTPUT

  SEND @TAB

  COPY 10 terminal_loop_ct

LABEL TERMINAL_LOOP

!OL1 - supply_w 7
  ENABLE PROMPT "7"
  RESPONSE "7" EVENT 38
  SEND "7"
  MODE OUTPUT

  SEND @TAB

!OL1 - item_id 1234
  ENABLE PROMPT "1"
  RESPONSE "1" EVENT 38
  SEND "1"
  MODE OUTPUT
  ENABLE PROMPT "2"
  RESPONSE "2" EVENT 38
  SEND "2"
  MODE OUTPUT
  ENABLE PROMPT "3"
  RESPONSE "3" EVENT 38
  SEND "3"
  MODE OUTPUT
  ENABLE PROMPT "4"
  RESPONSE "4" EVENT 38
  SEND "4"
  MODE OUTPUT

  SEND @TAB

!OL1 - qty     77

  ENABLE PROMPT "7"
  RESPONSE "7" EVENT 38
  SEND "7"
  MODE OUTPUT
  ENABLE PROMPT "7"
  RESPONSE "7" EVENT 38
  SEND "7"
  MODE OUTPUT

  SEND @TAB

  SUB 1 terminal_loop_ct

  IF terminal_loop_ct GT 0 THEN JUMP TERMINAL_LOOP

  ENABLE PROMPT "<01>"
  SEND @CTRLC                    !Exit from screen
  MODE OUTPUT
  SEND "9"                  !Exit from menu
END  !test_terminal_response
----------------------------
```

RTE Data Generation and Random Functions

```
/************************************************************
*********************
 * COPYRIGHT © 1993 BY                                      *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS                *
 * ALL RIGHTS RESERVED.                                     *

************************************************************
******************/
/*
 * These are the five routines to be called by the VAXRTE scripts to
generate
 * input data for the five different TPC-C transaction types.  All input data
 * are generated based on the requirement of the TPC-C spec.
 *
 * When response time is collected on VAXRTE, a variable can be logged
along
 * with the response time.  Taking advantage of this feature, a 32 bit
 * variable, context, is used to encode data that are needed to be collected
 * on the VAXRTE.  The way this variable is encoded is described below:
 *
 * NO_TXN:        XXXXXXXXXXXX3210987654321
 *   last 3 bits for txn type          | | 001 for New Order txn
 *   next 1 bit for illegal item       |   1 if rollback, 0 otherwise
 *   next 4 bits for              | |    # of order lines
 *   next 4 bits for          | |        # of remote order lines
 * PT_TXN:        XXXXXXXXXXXX3210987654321
 *   last 3 bits for txn type          | | 002 for Payment txn
 *   next 1 bit for                  |   1 if non-primary key access
 *   next 1 bit for                  |   1 if remote payment
 * OS_TXN:        XXXXXXXXXXXX3210987654321
 *   last 3 bits for txn type          | | 003 for Order Status txn
 *   next 1 bit for                  |   1 if non-primary key access
 * DY_TXN:        XXXXXXXXXXXX3210987654321
 *   last 3 bits for txn type          | | 004 for Delivery txn (interactive)
 * SL_TXN:        XXXXXXXXXXXX3210987654321
 *   last 3 bits for txn type          | | 005 for Stock Level txn
 *
 * MENU:          XXXXXXXXXXXX3210987654321
 *                                 || 006 NEW ORDER (  6   06)
 *                                ||| PAYMENT      ( 14   016)
 *                               | || ORDER STATUS  ( 22   026)
 *                               | || DELIVERY      ( 38   046)
 *                              |  || STOCK LEVEL   ( 70  0106)
************************************************************
******************/

#include <stdio.h>
#include <math.h>
#include <string.h>
#include <descrip.h>

#define ILLEGAL_ITEM     111111
#define MENU_TXN         0x000
#define NEW_ORDER_TXN    0x001
#define PAYMENT_TXN      0x002
#define ORDER_STATUS_TXN 0x003
#define DELIVERY_TXN     0x004
#define STOCK_LEVEL_TXN  0x005
#define ILLEGAL_ITEM_TXN 0x008
#define NON_KEY_ACCESS   0x008
```

```
#define REMOTE_PAYMENT   0x010

#define USE_DECIMAL 0
#define PCT_C_LAST 60

    /* MTH$RANDOM generate a random number of [0..1) */
float MTH$RANDOM();

    /* rand_mod would generate a random number between [0..modulo-1]
*/
#define rand_mod(modulo, seed)
(int)(MTH$RANDOM(seed)*((float)modulo))

    /* both random and NURand are specified in TPC-C spec */
#define random(x, y, seed) (rand_mod(y-x+1, seed)+x)
#define NURand(A, x, y, C, seed) \
     ((((random(0, A, seed) | random(x, y, seed)) + C) % (y-x+1)) + x)

static char *c_table[10] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
"ESE",
                    "ANTI", "CALLY", "ATION", "EING"};
static int c_length[10] = {3,5,4,3,4,3,4,5,5,4};
struct descriptor {
  unsigned short length;
  char type;
  char class;
  long *address;
};
long int c_id, d_id;



void get_no_data(int w_id,
           int W_MAX,
           int C_ID_CONST,
           int ITEM_CONST,
           long int *seed,
           struct descriptor *string,
           long int *context)
{
 int i, remote_txn=0, length=0, len;
 char *ptr;
 long int ol_supply_w_id, ol_quantity, ol_item_id, ol_count;

  *context = NEW_ORDER_TXN;
  ptr = string->address;

  d_id = random(1, 10, seed);
  len = sprintf(ptr, "%d\t", d_id);
  ptr += len;
  length += len;

  c_id = NURand(1023, 1, 3000, C_ID_CONST, seed);
  len = sprintf(ptr, "%d\t", c_id);
  length += len;
  ptr += len;

  ol_count = random(5, 15, seed);
  *context |= (ol_count << 4);
  for (i=0; i < ol_count; i++) {
       /* ol_supplying warehouse is remote 1% of the time */
    if ((rand_mod(1000, seed) < 990) || (W_MAX == 1))
      ol_supply_w_id = w_id;
    else {
      remote_txn++;
      if ((ol_supply_w_id = random(1, W_MAX-1, seed)) >= w_id)
        ol_supply_w_id++;
     }
    len = sprintf(ptr, "%d\t", ol_supply_w_id);
    ptr += len;
    length += len;

       /* set the last ol_item_id to 111111 if the rollback flag is 1 */
    if (i == ol_count-1) {              /* last order-line */
      if (rand_mod(1000, seed) < 10) {
        ol_item_id = ILLEGAL_ITEM;
        *context |= ILLEGAL_ITEM_TXN;
      }
      else
        ol_item_id = NURand(8191, 1, 100000, ITEM_CONST, seed);
     }
    else
      ol_item_id = NURand(8191, 1, 100000, ITEM_CONST, seed);

    len = sprintf(ptr, "%d\t", ol_item_id);
    ptr += len;
    length += len;

    ol_quantity = random(1, 10, seed);
    len = sprintf(ptr, "%d\t", ol_quantity);
    ptr += len;
    length += len;
   }            /* for loop */
  *context |= (remote_txn << 8);
  string->length = length-1;
}                               /* get_no_data */



void get_pt_data(int w_id,
           int W_MAX,
           int C_ID_CONST,
           int C_LAST_CONST,
           long int *seed,
           struct descriptor *string,
           long int *context)
{
 int tmp_len = 0;
 char *ptr;
 int length = 0, len;
 int index, i1, i2, i3;
 long int h_amount, c_w_id, c_d_id;

  *context = PAYMENT_TXN;
  ptr = string->address;

  d_id = random(1, 10, seed);
  len = sprintf(ptr, "%d\t", d_id);
  ptr += len;
  length += len;

  if (rand_mod(100, seed) < PCT_C_LAST) {         /* use C_LAST */
    *context |= NON_KEY_ACCESS;
    index = NURand(255, 0, 999, C_LAST_CONST, seed);
    i1 = index/100;                                 /* hundreds digit */
    i2 = (index - i1 * 100) / 10;               /* tens digit */
    i3 = index - (i1 * 100) - (i2 * 10);   /* ones digit */

    memcpy(ptr, "\t", 1); /* TAB over c_id field */
    ptr ++;
    memcpy(ptr, c_table[i1], c_length[i1]);
    ptr += c_length[i1];
    memcpy(ptr, c_table[i2], c_length[i2]);
    ptr += c_length[i2];
    memcpy(ptr, c_table[i3], c_length[i3]);
    ptr += c_length[i3];
    memcpy(ptr, "\t", 1); /* TAB to the next field */
    ptr++;
    length += c_length[i1] + c_length[i2] + c_length[i3] + 2;
   }
  else {              /* use c_id within [1..3000] */
    c_id = NURand(1023, 1, 3000, C_ID_CONST, seed);
    len = sprintf(ptr, "%d\t\t", c_id);
    ptr += len;
    length += len;
   }

  if ((rand_mod(1000, seed) < 850) || (W_MAX == 1)) {    /* 15% remote
payment */
    c_w_id = w_id;
    c_d_id = d_id;
   }
  else {
    *context |= REMOTE_PAYMENT;
```

```
    if ((c_w_id = rand_mod(W_MAX-1, seed)+1) >= w_id)
        c_w_id++;
    c_d_id = random(1, 10, seed);
   }

  len = sprintf(ptr, "%d\t", c_w_id);
  ptr += len;
  length += len;
  len = sprintf(ptr, "%d\t", c_d_id);
  ptr += len;
  length += len;

      /* h_amount in the within [1.00..5000.00] */
  h_amount = random(100, 500000, seed);
#if USE_DECIMAL == 1
  len = sprintf(ptr, "%6.2f", h_amount/100.00);
#else
  len = sprintf(ptr, "%d", h_amount);
#endif
  length += len;
  string->length = length;
}                               /* get_pt_data */



void get_os_data(int C_ID_CONST,
           int C_LAST_CONST,
           long int *seed,
           struct descriptor *string,
           long int *context)
{
 char *ptr;
 int length = 0, len, index, i1, i2, i3;

  *context = ORDER_STATUS_TXN;
  ptr = string->address;

  d_id = random(1, 10, seed);
  len = sprintf(ptr, "%d\t", d_id);
  ptr += len;
  length += len;

  if (rand_mod(100, seed) < PCT_C_LAST) {         /* use C_LAST */
     *context |= NON_KEY_ACCESS;
     index = NURand(255, 0, 999, C_LAST_CONST, seed);
     i1 = index/100;                                        /* hundreds digit */
     i2 = (index - i1 * 100) / 10;              /* tens digit */
     i3 = index - (i1 * 100) - (i2 * 10);    /* ones digit */

     memcpy(ptr, "\t", 1); /* TAB over c_id field */
     ptr ++;
     memcpy(ptr, c_table[i1], c_length[i1]);
     ptr += c_length[i1];
     memcpy(ptr, c_table[i2], c_length[i2]);
     ptr += c_length[i2];
     memcpy(ptr, c_table[i3], c_length[i3]);
     ptr += c_length[i3];
     length += c_length[i1] + c_length[i2] + c_length[i3] + 1;
    }
  else {                /* use c_id within [1..3000] */
     c_id = NURand(1023, 1, 3000, C_ID_CONST, seed);
     len = sprintf(ptr, "%d", c_id);
     length += len;
    }

  string->length = length;
}                               /* get_os_data */



void get_dy_data(long int *seed,
           struct descriptor *string,
           long int *context)
{
  *context = DELIVERY_TXN;
              /* generate o_carrier_id in the range [1..10] */
  string->length = sprintf(string->address, "%d", random(1, 10, seed));
}                               /* get_dy_data */


void get_sl_data(long int *seed,
           struct descriptor *string,
           long int *context)
{
  *context = STOCK_LEVEL_TXN;
              /* generate threshold in the range [10..20] */
  string->length=sprintf(string->address, "%d", random(10, 20, seed));

}                               /* get_sl_data */

----------------------
```

Appendix D

Digital UNIX Tunable Parameters: Server

sysconfigtab

```
#
#
*************************************************************
******
# *                                                         *
# *   Copyright (c) Digital Equipment Corporation, 1991, 1995   *
# *                                                         *
# *   All Rights Reserved.  Unpublished rights  reserved  under   *
# *   the copyright laws of the United States.                  *
# *                                                         *
# *   The software contained on this media  is  proprietary  to   *
# *   and  embodies  the  confidential  technology  of  Digital   *
# *   Equipment Corporation.  Possession, use,  duplication  or   *
# *   dissemination of the software and media is authorized only  *
# *   pursuant to a valid written license from Digital Equipment  *
# *   Corporation.                                              *
# *                                                         *
# *   RESTRICTED RIGHTS LEGEND   Use, duplication, or disclosure  *
# *   by the U.S. Government is subject to restrictions  as  set  *
# *   forth in Subparagraph (c)(1)(ii)  of  DFARS  252.227-7013,  *
# *   or  in  FAR 52.227-19, as applicable.                     *
# *                                                         *
#
*************************************************************
******
#
#
# HISTORY
#
# (c) Copyright 1990, 1991, 1992, 1993 OPEN SOFTWARE
FOUNDATION, INC.
# ALL RIGHTS RESERVED
#
#
# OSF/1 1.2

ipc:
        shm-max = 2139095040
proc:
        per-proc-data-size = 3200000000
        max-per-proc-data-size = 3200000000
        per-proc-address-space = 3200000000
        max-per-proc-address-space = 3200000000
rt:
        aio-max-num = 1024
        aio-max-retry = 2
        aio-task-max-num = 1024
```

System Configuration File

```
ident                   "PINHD"

options                 UERF
options                 OSF
options                 _LMF_
options                 BIN_COMPAT
options                 COMPAT_43
options                 MACH
options                 MACH_IPC_TCACHE
options                 MACH_IPC_WWA
options                 MACH_IPC_XXXHACK
options                 BUFCACHE_STATS
options                 INOCACHE_STATS
options                 STAT_TIME
options                 VAGUE_STATS
options                 UFS
options                 NFS
options                 NFS_SERVER
options                 STRKINFO
options                 STREAMS
options                 LDTTY
options                 RPTY
options                 INET
options                 UIPC
options                 SYSV_COFF
options                 QUOTA
options                 LABELS
options         SL
options SNMPINFO
options                 DLI
options                 BSD_TTY
options                 LAT
options                 DLB
options         PROCFS
#
# Standard options.
#
options                 UNIX_LOCKS
options                 SER_COMPAT
options                 RT_PREEMPT
options                 RT_SCHED
options                 RT_SCHED_RQ
options                 RT_PML
options                 RT_TIMER
options                 RT_SEM
options                 RT_CSEM
options                 RT_IPC
#
makeoptions             CDEBUGOPTS="-g3"
makeoptions             CCOMPRESS="-compress"
makeoptions             PROFOPTS="-DPROFILING -DPROFTYPE=4"
#
# Max number of processors in the system (DO NOT CHANGE)
#
processors      16

#
# Special options (see configuring the kernel chapter
# in the Guide to System Administration)
#
timezone        0 dst 0
dfldsiz                 134217728
maxdsiz                 1073741824
dflssiz                 2097152
maxssiz                 33554432
cpu                     "DEC1000A_5"
maxusers        32
machine                 alpha

config                  vmunix       swap generic

bus             pci0       at nexus?
callout after_c "../bin/mkdata pci"
bus             eisa0      at pci0      slot  7
callout after_c "../bin/mkdata eisa"
controller      ace0       at eisa0     slot  8 vector    aceintr
controller      ace1       at eisa0     slot  9 vector    aceintr
controller      lp0        at eisa0     slot  10 vector   lpintr
controller      fdi0       at eisa0     slot  11 vector   fdintr
device disk     fd0        at fdi0      drive     0
controller      vga0       at eisa0     slot  12 vector   vgaintr
controller      gpc0       at eisa0     slot  0 vector    gpcintr
bus             pci2000    at pci0      slot  8
bus             isp0       at pci2000   slot  0 vector    ispintr
controller      scsi0      at isp0      slot  0
```

```
device disk     rz0       at scsi0    drive     0
device disk     rz1       at scsi0    drive     8
device disk     rz2       at scsi0    drive     16
device disk     rz3       at scsi0    drive     24
device disk     rz4       at scsi0    drive     32
device disk     rz6       at scsi0    drive     48
bus             pza0      at pci2000   slot  4 vector    pzaintr
controller      scsi1     at pza0     slot  0
device disk     rz8       at scsi1    drive     64
device disk     rz9       at scsi1    drive     72
device disk     rz10      at scsi1    drive     80
device disk     rz11      at scsi1    drive     88
device disk     rz12      at scsi1    drive     96
device disk     rz13      at scsi1    drive     104
device disk     rz14      at scsi1    drive     112
bus             pza1      at pci0     slot  12 vector    pzaintr
controller      scsi2     at pza1     slot  0
device disk     rz18      at scsi2    drive     144
device disk     rz19      at scsi2    drive     152
device disk     rz20      at scsi2    drive     160
controller      tu0       at pci0     slot  11 vector    tuintr


scs_sysid     1
pseudo-device               sysv_hab
pseudo-device               svid_three_hab
pseudo-device               svr_four_hab
pseudo-device               soe_two_hab
pseudo-device               rt_hab
pseudo-device               ether
pseudo-device               loop
#
# Max number of processors in the system (DO NOT CHANGE)
#
pseudo-device               cpus            16
pseudo-device     ws
```

Digital UNIX Tunable Parameters: Client

sysconfigtab

```
#
#
************************************************************
******
# *                                                        *
# *   Copyright (c) Digital Equipment Corporation, 1991, 1995    *
# *                                                        *
# *   All Rights Reserved.  Unpublished rights  reserved  under   *
# *   the copyright laws of the United States.                   *
# *                                                        *
# *   The software contained on this media  is  proprietary  to   *
# *   and  embodies  the  confidential  technology  of  Digital   *
# *   Equipment Corporation.  Possession, use,  duplication  or   *
# *   dissemination of the software and media is authorized only  *
# *   pursuant to a valid written license from Digital Equipment  *
# *   Corporation.                                               *
# *                                                        *
# *   RESTRICTED RIGHTS LEGEND   Use, duplication, or disclosure  *
# *   by the U.S. Government is subject to restrictions  as  set  *
# *   forth in Subparagraph (c)(1)(ii)  of  DFARS  252.227-7013,  *
# *   or  in  FAR 52.227-19, as applicable.                      *
# *                                                        *
#
************************************************************
******
#
#
# HISTORY
#
# (c) Copyright 1990, 1991, 1992, 1993 OPEN SOFTWARE
FOUNDATION, INC.
# ALL RIGHTS RESERVED
#
#
# OSF/1 1.2

ipc:

shm-max = 100000000
shm-seg = 32
msg-max = 32768
msg-mnb = 32768
#msg-mni = 1024
msg-mni = 1700
msg-tql = 4096
sem-mni = 10
#sem-msl = 150
sem-msl=230
sem-opm = 2000
sem-ume = 2000
sem-vmx = 32767
sem-aem = 16384
max-kernel-ports = 38615
port-hash-max-num = 1930750
port-reserved-max-num = 38615
set-max-num = 6405
proc:
max-proc-per-user = 2048
```

System Configuration File

```
ident                   "SUT21"

options                 UERF
options                 OSF
options                 _LMF_
options                 BIN_COMPAT
options                 COMPAT_43
options                 MACH
options                 MACH_IPC_TCACHE
options                 MACH_IPC_WWA
options                 MACH_IPC_XXXHACK
options                 BUFCACHE_STATS
options                 INOCACHE_STATS
options                 STAT_TIME
options                 VAGUE_STATS
options                 UFS
options                 NFS
options                 NFS_SERVER
options                 STRKINFO
options                 STREAMS
options                 LDTTY
options                 RPTY
options                 INET
options                 UIPC
options                 SYSV_COFF
options                 QUOTA
options                 LABELS
options       SL
options SNMPINFO
options                 DLI
options                 BSD_TTY
options                 LAT
options                 DLB
options       PROCFS
options          ATM
options                 MSFS
options       FFM_FS
options       KDEBUG
options       PACKETFILTER
options   PCKT
options       TIRDWR
options       TIMOD
```

```
options                 XTISO
options                 FFM_FS
options                 CDFS
options         DEC_AUDIT
#
# Standard options.
#
options                 UNIX_LOCKS
options                 SER_COMPAT
options                 RT_PREEMPT
options                 RT_SCHED
options                 RT_SCHED_RQ
options                 RT_PML
options                 RT_TIMER
options                 RT_SEM
options                 RT_CSEM
options                 RT_IPC
#
makeoptions             CDEBUGOPTS="-g3"
makeoptions             CCOMPRESS="-compress"
makeoptions             PROFOPTS="-DPROFILING -DPROFTYPE=4"
#
# Max number of processors in the system (DO NOT CHANGE)
#
processors      16

#
# Special options (see configuring the kernel chapter
# in the Guide to System Administration)
#
timezone        0 dst 0
dfldsiz                 134217728
maxdsiz                 1073741824
dflssiz                 2097152
maxssiz                 33554432
cpu                     "DEC1000A"
maxusers        1024
machine                 alpha

config                  vmunix          swap generic

bus             pci0        at nexus?
callout after_c "../bin/mkdata pci"
bus             eisa0       at pci0     slot  7
callout after_c "../bin/mkdata eisa"
controller      ace0        at eisa0    slot  9 vector    aceintr
controller      ace1        at eisa0    slot  10 vector   aceintr
controller      lp0         at eisa0    slot  11 vector   lpintr
controller      fdi0        at eisa0    slot  12 vector   fdintr
device disk     fd0         at fdi0     drive     0
controller      vga0        at eisa0    slot  13 vector   vgaintr
controller      gpc0        at eisa0    slot  0 vector    gpcintr
bus             pci1000     at pci0     slot  8
bus             isp0        at pci1000  slot  0 vector    ispintr
controller      scsi0       at isp0     slot  0
device disk     rz0         at scsi0    drive     0
device disk     rz1         at scsi0    drive     8
device disk     rz2         at scsi0    drive     16
device disk     rz3         at scsi0    drive     24
device disk     rz4         at scsi0    drive     32
controller      tu0         at pci0     slot  11 vector    tuintr
controller      tu1         at pci0     slot  13 vector    tuintr


scs_sysid       1
pseudo-device           sysv_hab
pseudo-device           svid_three_hab
pseudo-device           svr_four_hab
pseudo-device           soe_two_hab
pseudo-device           rt_hab
pseudo-device           ether
pseudo-device           loop
pseudo-device   prf     6
pseudo-device           lv              2
pseudo-device           lsm_ted         0
pseudo-device           lsm             1
#
# Max number of processors in the system (DO NOT CHANGE)
#
pseudo-device           cpus            16
pseudo-device           ws
```

Sybase Tunable Parameter/Configuration File

```
#########################################################
##################
#
#                       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 = 32000
        print recovery information = DEFAULT
        tape retention in days = DEFAULT

[Cache Manager]
        number of oam trips = DEFAULT
        number of index trips = DEFAULT
        procedure cache percent = 2
        memory alignment boundary = DEFAULT

[Named Cache:c_customer]
        cache size = 8M
        cache status = mixed cache

[2K I/O Buffer Pool]
        pool size = 8M
        wash size = 6144 K

[Named Cache:c_customer_index]
        cache size = 40M
        cache status = mixed cache

[2K I/O Buffer Pool]
        pool size = 40M
        wash size = 512 K

[Named Cache:c_log]
        cache size = 6M
        cache status = log only

[2K I/O Buffer Pool]
        pool size = 1M
        wash size = 256 K

[4K I/O Buffer Pool]
        pool size = 5M
        wash size = 256 K

[Named Cache:c_no_ol]
        cache size = 25M
        cache status = mixed cache

[2K I/O Buffer Pool]
        pool size = 25M
        wash size = 4096 K

[Named Cache:c_ol_index]
```

```
        cache size = 25M
        cache status = mixed cache

[2K I/O Buffer Pool]
        pool size = 25M
        wash size = 4096 K

[Named Cache:c_orders]
        cache size = 20M
        cache status = mixed cache

[2K I/O Buffer Pool]
        pool size = 15M
        wash size = 512 K

[16K I/O Buffer Pool]
        pool size = 5M
        wash size = 512 K

[Named Cache:c_stock]
        cache size = 215M
        cache status = mixed cache

[2K I/O Buffer Pool]
        pool size = 215M
        wash size = 4096 K

[Named Cache:c_stock_index]
        cache size = 45M
        cache status = mixed cache

[2K I/O Buffer Pool]
        pool size = 45M
        wash size = 512 K

[Named Cache:c_tinyhot]
        cache size = 15M
        cache status = mixed cache

[2K I/O Buffer Pool]
        pool size = 15M
        wash size = 512 K

[Named Cache:default data cache]
        cache size = 15M
        cache status = default data cache

[2K I/O Buffer Pool]
        pool size = 12M
        wash size = 2048 K

[16K I/O Buffer Pool]
        pool size = 3M
        wash size = 2048 K

[Disk I/O]
        allow sql server async i/o = DEFAULT
        disk i/o structures = DEFAULT
        page utilization percent = DEFAULT
        number of devices = 46

[Network Communication]
        default network packet size = DEFAULT
        max network packet size = 4096
        remote server pre-read packets = DEFAULT
        number of remote connections = DEFAULT
        allow remote access = DEFAULT
        number of remote logins = DEFAULT
        number of remote sites = DEFAULT
        max number network listeners = DEFAULT
        tcp no delay = DEFAULT

[O/S Resources]
        max async i/os per engine = 1024
        max async i/os per server = 1024

[Physical Resources]

[Physical Memory]
        total memory = 250880
        additional network memory = 1228800
        lock shared memory = DEFAULT
        shared memory starting address = DEFAULT

[Processors]
        max online engines = DEFAULT
        min online engines = DEFAULT

[SQL Server Administration]
        number of open objects = DEFAULT
        number of open databases = DEFAULT
        audit queue size = DEFAULT
        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
        number of mailboxes = DEFAULT
        number of messages = DEFAULT
        number of alarms = DEFAULT
        number of pre-allocated extents = DEFAULT
        event buffers per engine = DEFAULT
        cpu accounting flush interval = 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 = 5000
        cpu grace time = DEFAULT
        number of sort buffers = DEFAULT
        sort page count = DEFAULT
        number of extent i/o buffers = DEFAULT
        size of auto identity column = DEFAULT
        identity grab size = DEFAULT
        lock promotion HWM = DEFAULT
        lock promotion LWM = DEFAULT
        lock promotion PCT = DEFAULT
        housekeeper free write percent = 0
        partition groups = DEFAULT
        partition spinlock ratio = DEFAULT

[User Environment]
        number of user connections = 100
        stack size = DEFAULT
        stack guard size = DEFAULT
        systemwide password expiration = DEFAULT
        permission cache entries = DEFAULT
        user log cache size = 4096
        user log cache spinlock ratio = DEFAULT

[Lock Manager]
        number of locks = 10000
        deadlock checking period = DEFAULT
        freelock transfer block size = DEFAULT
        max engine freelocks = 25
        address lock spinlock ratio = DEFAULT
        page lock spinlock ratio = DEFAULT
        table lock spinlock ratio = DEFAULT
----------------------
```

Tuxedo Tunable Parameters

```
*RESOURCES
IPCKEY          64000
MASTER          SERVER1
PERM            0666
MAXACCESSERS    1670
MAXSERVERS      50
MAXSERVICES     50
MODEL           SHM
```

```
LDBAL           Y
MAXGTT          300
MAXBUFTYPE      16
MAXBUFSTYPE     32
SCANUNIT        60
SANITYSCAN      5
DBBLWAIT        1
BBLQUERY        30
BLOCKTIME       5

*MACHINES
sut21       LMID=SERVER1
            TUXCONFIG="/usr/users/tuxedo/kits/tpcc/tuxconfig"
            TUXOFFSET=0
            ROOTDIR="/usr/opt/tuxedo"
            APPDIR="/usr/users/tuxedo/kits/tpcc"
            TLOGOFFSET=0
            TLOGNAME=TLOG
            TLOGSIZE=100

*GROUPS
TPCC1       LMID=SERVER1        GRPNO=1
            TMSCOUNT=0

*SERVERS

DEFAULT:SRVGRP=TPCC1 REPLYQ=N RESTART=N

tpcc_server    SRVID=100 RQADDR = "DY0" CLOPT="-s
DELIVERY_SVC -o stdout.0"
tpcc_server    SRVID=101 RQADDR = "DY1" CLOPT="-s
DELIVERY_SVC -o stdout.1"
tpcc_server    SRVID=102 RQADDR = "DY2" CLOPT="-s
DELIVERY_SVC -o stdout.2"
tpcc_server    SRVID=103 RQADDR = "DY3" CLOPT="-s
DELIVERY_SVC -o stdout.3"
tpcc_server    SRVID=104 RQADDR = "DY4" CLOPT="-s
DELIVERY_SVC -o stdout.4"
tpcc_server    SRVID=105 RQADDR = "DY5" CLOPT="-s
DELIVERY_SVC -o stdout.5"
tpcc_server    SRVID=106 RQADDR = "DY6" CLOPT="-s
DELIVERY_SVC -o stdout.6"

tpcc_server    SRVID=200 RQADDR = "NO0" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=201 RQADDR = "NO1" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=202 RQADDR = "NO2" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=203 RQADDR = "NO3" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=204 RQADDR = "NO4" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=205 RQADDR = "NO5" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=206 RQADDR = "NO6" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=207 RQADDR = "NO7" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=208 RQADDR = "NO8" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=209 RQADDR = "NO9" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=210 RQADDR = "NO10" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=211 RQADDR = "NO11" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=212 RQADDR = "NO12" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=213 RQADDR = "NO13" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=214 RQADDR = "NO14" CLOPT="-s
NEWORDER_SVC"
tpcc_server    SRVID=215 RQADDR = "NO15" CLOPT="-s
NEWORDER_SVC"

tpcc_server    SRVID=300 RQADDR = "OS0" CLOPT="-s
ORDSTAT_SVC"
tpcc_server    SRVID=301 RQADDR = "OS1" CLOPT="-s
ORDSTAT_SVC"
tpcc_server    SRVID=302 RQADDR = "OS2" CLOPT="-s
ORDSTAT_SVC"

tpcc_server    SRVID=400 RQADDR = "PT0" CLOPT="-s
PAYMENT_SVC"
tpcc_server    SRVID=401 RQADDR = "PT1" CLOPT="-s
PAYMENT_SVC"
tpcc_server    SRVID=402 RQADDR = "PT2" CLOPT="-s
PAYMENT_SVC"
tpcc_server    SRVID=403 RQADDR = "PT3" CLOPT="-s
PAYMENT_SVC"
tpcc_server    SRVID=404 RQADDR = "PT4" CLOPT="-s
PAYMENT_SVC"
tpcc_server    SRVID=405 RQADDR = "PT5" CLOPT="-s
PAYMENT_SVC"
tpcc_server    SRVID=406 RQADDR = "PT6" CLOPT="-s
PAYMENT_SVC"
tpcc_server    SRVID=407 RQADDR = "PT7" CLOPT="-s
PAYMENT_SVC"

tpcc_server    SRVID=500 RQADDR = "SL0" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=501 RQADDR = "SL1" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=502 RQADDR = "SL2" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=503 RQADDR = "SL3" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=504 RQADDR = "SL4" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=505 RQADDR = "SL5" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=506 RQADDR = "SL6" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=507 RQADDR = "SL7" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=508 RQADDR = "SL8" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=509 RQADDR = "SL9" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=510 RQADDR = "SL10" CLOPT="-s
STOCKLEV_SVC"
tpcc_server    SRVID=511 RQADDR = "SL11" CLOPT="-s
STOCKLEV_SVC"

*SERVICES

neworder_svc
            LOAD=1              PRIO=1
            BUFTYPE="CARRAY"
            TRANTIME=900
            AUTOTRAN=N
payment_svc
            LOAD=1              PRIO=1
            BUFTYPE="CARRAY"
            TRANTIME=900
            AUTOTRAN=N
ordstat_svc
            LOAD=1              PRIO=1
            BUFTYPE="CARRAY"
            TRANTIME=900
            AUTOTRAN=N
delivery_svc
            LOAD=1              PRIO=1
            BUFTYPE="CARRAY"
            TRANTIME=900
            AUTOTRAN=N
stocklev_svc
            LOAD=1              PRIO=1
            BUFTYPE="CARRAY"
            TRANTIME=900
            AUTOTRAN=N
```

Appendix E

Auditor Attestation

Appendix F

Price Quotations