hp server rp8400
using
HP-UX 11i 64-bit Base 0S, September 2001
and
Sybase Adaptive Server Enterprise v12.5

TPC Benchmark® C
Full Disclosure Report

Fifth Edition

Submitted for Review
February 22, 2002

Hewlett-Packard Company believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. Hewlett-Packard Company assumes no responsibility for any errors that may appear in this document.
The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, Hewlett-Packard Company provides no warranty of the pricing information in this document.
Benchmark results are highly dependent upon workload, specific application requirements, and system design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC Benchmark® C should not be used as a substitute for a specific customer application benchmark when critical capacity planning and/or product evaluation decisions are contemplated.
All performance data contained in this report was obtained in a rigorously controlled environment. Results obtained in other operating environments may vary significantly. Hewlett-Packard Company does not warrant or represent that a user can or will achieve similar performance expressed in transactions per minute (tpmC®) or normalized price/performance ($/tpmC®). No warranty of system performance or price/performance is expressed or implied in this report.

Printed in U.S.A., February 22, 2002.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark® C test conducted on the hp server rp8400 in a client/server configuration, using Sybase Adaptive Server Enterprise v12.5 and the BEA TUXEDO 6.4 transaction monitor. The operating system used for the benchmark was Hewlett-Packard's HP-UX 11i 64-bit Base 0S, September 2001. The application was written in C and compiled using HP C/ANSI C/HP-UX.

TPC Benchmark C Metrics

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

Standard and Executive Summary Statements

Page *iii* contains the standard system summary and pages ***iv-vi*** contain the executive summary of the benchmark results for the hp server rp8400 .

Auditor

The benchmark configuration, environment and methodology used to produce and validate the test results, and the pricing model used to calculate the price/performance, were audited by Lorna Livingtree for Performance Metrics, Inc. to verify compliance with the relevant TPC specifications.

Standard System Summary

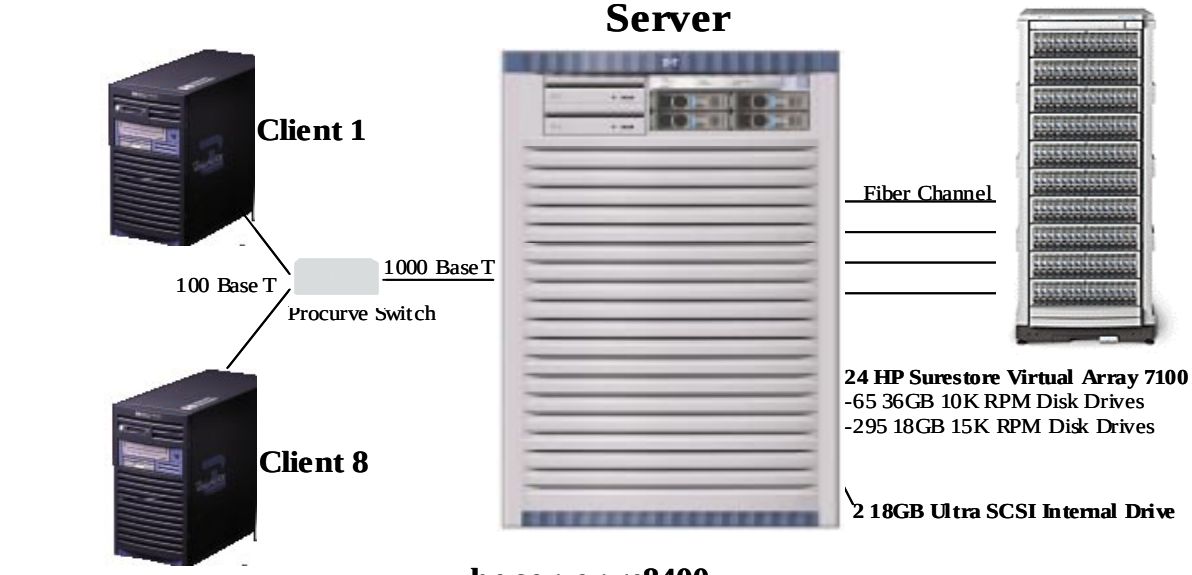
Company Name	**System Name**	**Database Software**	**Operating System Software**
Hewlett-Packard Company	hp server rp8400	Sybase Adaptive Server Enterprise v12.5	HP-UX 11i 64-bit Base 0S, September 2001
HP H/W Availability Date - September 18, 2001 S/W Availability Date - Now			
Total System Cost	**TPC-C® Throughput**	**Price/Performance**	
Hardware Software 5-year maintenance	Sustained maximum throughput of System running TPC-C® expressed in transactions per minute	Total system cost/tpmC ($2,015,289/140239.97)	
$2,015,289	**140,239.97 tpmC**	**$14.37 per tpmC**	

hp server rp8400

TPC-C Revision 5

Report Date: February 22, 2002

Total System Cost	TPC Throughput	Price/Performance		Availability Date
$2,015,289	**140,239.97 tpmC**	**$14.37/tpmC**		**September 18, 2001**
Processors	Database Manager	Operating System	Other Software	Number of Users
16 PA-RISC 8700 750MHz	**Sybase Adaptive Server Enterprise v12.5**	**HP-UX 11i 64-bit Base 0S, September 2001**	**BEA TUXEDO 6.4**	**114,400**



System Components	*Server (hp server rp8400)*		*each Client (8 hp workstation c3700)*	
	Qty	*Type*	*Qty*	*Type*
Processors	16	750MHz PA-RISC 8700	1	750MHz PA-RISC 8700
Cache Memory	each	.75 MB I-cache, 1.5 MB D-cache	each	.75MB I-cache/1.5MB D-cache
Memory	64	GB	1	8 GB
Disk Controllers	8	PCI Fibre Channel 2X	1	Ultra2 SCSI LVD
Disk Drives	24	HP Surestore Virtual Array 7100 with 295 18GB 15K RPM drive and 65 36GB 10K RPM drives	1	18 GB disk
Internal Drive	2	18GB Ultra SCSI drive		
Total Storage	3,498	GB		
Tape Drives	1	DVD ROM		
Terminals	1	Console Terminal	1	Console Terminal

hp server rp8400

TPC-C Rev 5

Report Date: February 22, 2002

Description	Part Number	Brand	Price Key	Unit Price	Qty	Price	3Year Main.Price
Server Hardware							
hp server rp8400 Enterprise Server	A6093A		1	40000	1	40,000	49,218
750MHz PA_RISC 8700 CPU-Dual	A6444A, Opt. 0D1		1	46000	8	368,000	69,216
Processor/Memory Cell Board	A6094A, Opt. 0D1		1	7000	4	28,000	
Core I/O	A6096A, Opt, 0D1		1	5000	1	5,000	
4GB SyncDRAM Memory Modules	A6098A, Opt. 0D1		1	28000	16	448,000	
18GB Ultra SCSI Internal Drive	A6723A, Opt 0D1		1	1764	2	3,528	
DVD Drive	A6180A, Opt. 0D1		1	460	1	460	
PCI Fibre Channel 2X	A5158A, Opt 0D1		1	2240	8	17,920	
PCI 1000BT Lan Adapter	A4926A, Opt. 0D1		1	2135	1	2,135	
Rack System/E Cabinet	J1528AZ		1	510	1	510	
4.5m, Server to UPS Power Cord	J1528AZ, Opt. A5F		1	0	1	0	
System Hotswap Power Supply	A6099A, Opt 0D1		1	560	4	2,240	
HPUX Opertating Environment	B9088AC		1	520	1	520	
6.5kW/9kVA HP UPS Rackmount	A6584A		1	8799	5	43,995	
Surestore VA7100 w/dual controllers 512/MB cache	A6262A		1	44250	24	1,062,000	101,064
18GB 15K RPM FC HDD	A6191A		1	914	295	269,630	
36GB 10K RPM FC HDD	A6192		1	1026	65	66,690	
2 meter Fibre Optic Cable	A3583A		1	175	24	4,200	
10 Port Short Wave Fibre Channel Hub	A3724A		1	9690	8	77,520	
16 meter Fibre Optic Cable	92227C		1	200	8	1,600	
HP9000 Std. Rack System E41	A4902AZ		1	2350	4	9,400	
Modular Power Dist.	A5137AZ		1	155	8	1,240	
200-240 Volts	A5137AZ, Opt AW4		1	115	8	920	
				Subtotal		**2,453,508**	**219,498**
Server Software							
Sybase Adaptive Server Enterprise 12.5	Runtime	Sybase	2	295,000	1	295,000	194,700
Software Developers Kit	License	Sybase	2	995	1	995	480
				Subtotal		**295,995**	**195,180**
Client Hardware							
hp workstation c3700 with 2GB Memory	A6057B		1	13390	8	107,120	31,048
1 GB Memory Moudle	A6016A, Opt. OD1		1	1395	48	66,960	
700/96 Console	C1064GX		1	550	1	550	
100BaseT 4-Port PCI Lan Adapter	A5506B		1	1830	8	14,640	
18 GB LVD 10K RPM Disk	A4998A, Opt. OD1		1	595	8	4,760	
				Subtotal		**194,030**	**31,048**
Client Software							
HP C/ANSI C Compiler	B3901BA, Option AH0		1	1,600	1	1,600	170
BEA Tuxedo 6.4		Bea Sys.	3	2,850	8	22,800	16,560
				Subtotal		**24,400**	**16,730**
User Connectivity							
HP ProCurve Switch 8000M	J4110A		1	1,699	1	1,699	821
HP ProCurve Switch 10/100Base-T Module	J4111A		1	509	1	509	
HP ProCurve Switch Gigabit-SX Module	J4113A		1	1,189	1	1,189	
				Subtotal		**3,397**	**821**
	Large configuration Discount and Support Prepayment*					(1,326,268)	(93,050)
* All discounts are based on US list prices and for similar quantities and configurations				**Total**		**1,645,063**	**370,227**

Notes:1=HP, 2=Sybase 3=BEA Systems

Note: 3 of the hp workstations c3700 clients were emulated, see FDR for further documentation

Audited by Lorna Livingtree for Performance Metrics, Inc.

Five Year Cost of Ownership:	**$2,015,289**
tpmC Rating:	**140,240**
$/tpmC:	**$14.37**

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

Numerical Quantities Summary for hp server rp8400

MQTH, Computed Maximum Qualified Throughput **140,239.97 tpmC**

Response Times (in seconds)

	90th %-ile	Maximum	Average
New-Order	1.63s	8.17s	0.88s
Payment	1.59s	8.13s	0.84s
Order-Status	1.67s	8.12s	0.92s
Delivery (interactive portion)	0.15s	0.70s	0.09s
Delivery (deferred portion)	1.80s	7.54s	1.04s
Stock-Level	1.95s	8.03s	1.09s
Menu	0.10s	1.04s	0.003s

Transaction Mix, in percent of total transactions

New-Order	44.80%
Payment	43.05%
Order-Status	4.05%
Delivery	4.05%
Stock-Level	4.05%

Keying/Think Times

	Keying Time			Think Time		
	Min	Avg	Max	Min	Avg	Max
New-Order	18.02s	18.03s	18.05s	0.01s	12.14s	191.15s
Payment	3.01s	3.02s	3.04s	0.01s	12.07s	201.55s
Order-Status	2.01s	2.02s	2.04s	0.01s	10.13s	154.64s
Delivery (interactive)	2.01s	2.02s	2.04s	0.01s	5.07s	77.46s
Stock-Level	2.01s	2.02s	2.04s	0.01s	5.06s	76.21s

Test Duration

Ramp up time	44 minutes
Measurement interval	120 minutes
Transactions during measurement interval	37,565,265
Ramp down time	6.45 minutes

Checkpointing

Number of checkpoints in measurement interval	4
Checkpoint Interval	30 minutes

TPC Benchmark C Overview

This is the full disclosure report for a benchmark test of the hp server rp8400 using Sybase Adaptive Server Enterprise v12.5 . It meets the requirements of the TPC Benchmark® C Standard Specification, Revision 5 dated March, 2001.

TPC Benchmark® C was developed by the **T**ransaction Processing **P**erformance **C**ouncil (TPC). It is the intent of this group to develop a suite of benchmarks to measure the performance of computer systems executing a wide range of applications. Hewlett-Packard Company Sybase Inc. are active participants in the TPC.

*TPC Benchmark® C is an **O**n **L**ine **T**ransaction **P**rocessing (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 of data access and update

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

Despite the fact that this benchmark offers a rich environment that emulates many OLTP applications, this benchmark does not reflect the entire range of OLTP requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C® approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to other environments are not recommended.

Hewlett-Packard Company does not warrant or represent that a user can or will achieve performance similar to the benchmark results contained in this report. No warranty of system performance or price/performance is expressed or implied by this report.

1 General Items

1.1 Application Code and Definition Statements

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

Appendix A contains the HP C/ANSI C/HP-UX application code used in this TPC-C® test.

1.2 Test Sponsor

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

The AlwaysOn Infrastructure Solutions Division of Hewlett-Packard Company and Sybase Inc. are the test sponsors of this TPC Benchmark® C.

1.3 Parameter Settings

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

- Database options
- Recover/commit options
- Consistency/locking options
- Operating system and application configuration parameter
- Compilation and linkage options and run-time optimizations used to create/install applications, OS, and/or databases

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

The intent of the above clause is that anyone attempting to recreate the benchmark environment has sufficient information to compile, link, optimize, and execute all software used to produce the disclosed benchmark result.

Appendix A contains the application "make" files. Appendix C contains the HP-UX operating system parameters used to generate the kernel for the configuration used in this benchmark. Also included are all of the Sybase Adaptive Server Enterprise v12.5 database parameters and the BEA TUXEDO 6.4 transaction monitor parameters used.

1.4 Configuration Diagrams

Diagrams of both measured and priced configurations must be provided, 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 of memory unique to the test*
- *Number and type of disk units (and controllers, if applicable)*
- *Number of channels or bus connections to disk units, including the protocol type*
- *Number of LAN (e.g. Ethernet) connections, including routers, work stations, terminals, etc, that were physically used in the test or are incorporated into the pricing structure (See Clause 8.1.8)*
- *Type and run-time execution location of software components (e.g. DBMS, client processes, transaction monitors, software drivers, etc)*

The server System Under Test, an hp server rp8400 depicted in Figure 1.1, consisted of:

- 16 750MHz PA-RISC 8700 System Processors
- 64 GB of memory

- 8 PCI Fibre Channel 2X Adapters
- 24 HP Surestore Virtual Array 7100 (with 65 36GB 10K RPM disks and 295 15K RPM disks)
- One 1GB Ethernet LAN interface

As indicated in Figure 1.2, this benchmark configuration used Remote Terminal Emulator (RTE) programs that executed on 2 N4000 Enterprise Server drivers partitioned into 4 virtual environments each to emulate TPC-C user sessions. The emulated users on the driver systems were directly connected to the client systems under test via 8 separate 100 Base-T local area network (LAN) and communicated using TCP/IP. The clients were connected to the SUT via one HP Procurve 100BT/ 1000BT switch.

The priced configuration for the hp server rp8400 is shown in Figure 1.1. In the priced configuration, the RTE shown in the benchmark configuration is replaced by the appropriate number of workstations (emulating ANSI terminals) connected via hubs/switches.

Figure 1.1: hp server rp8400 Priced Configuration

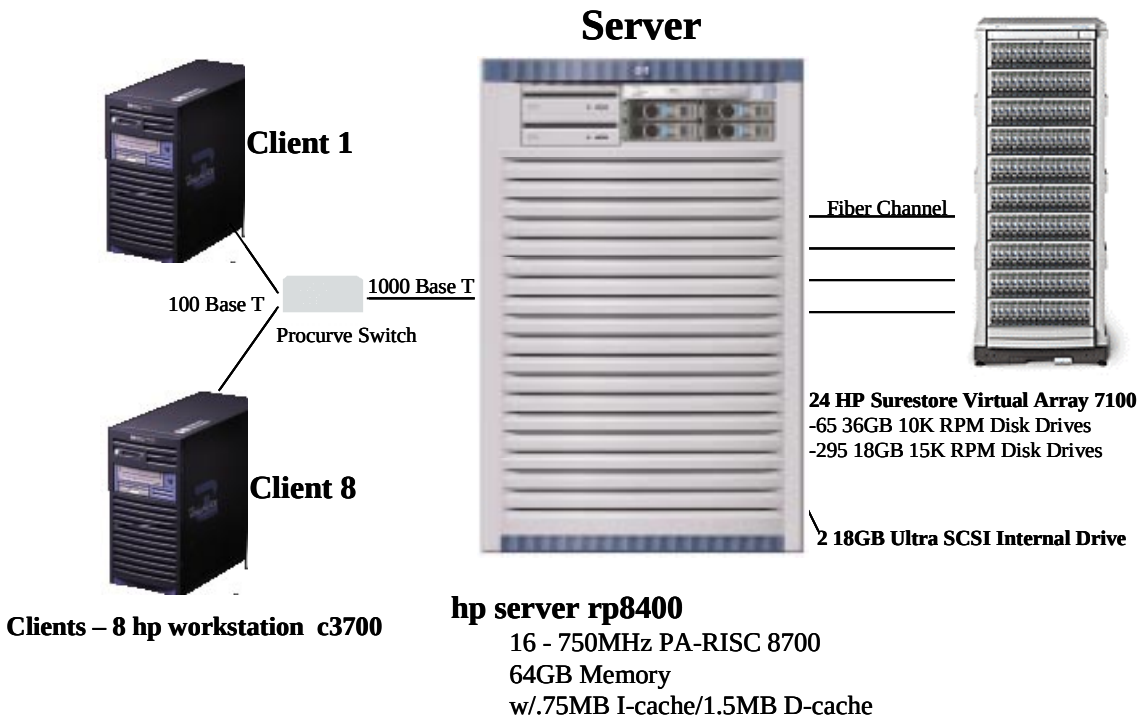
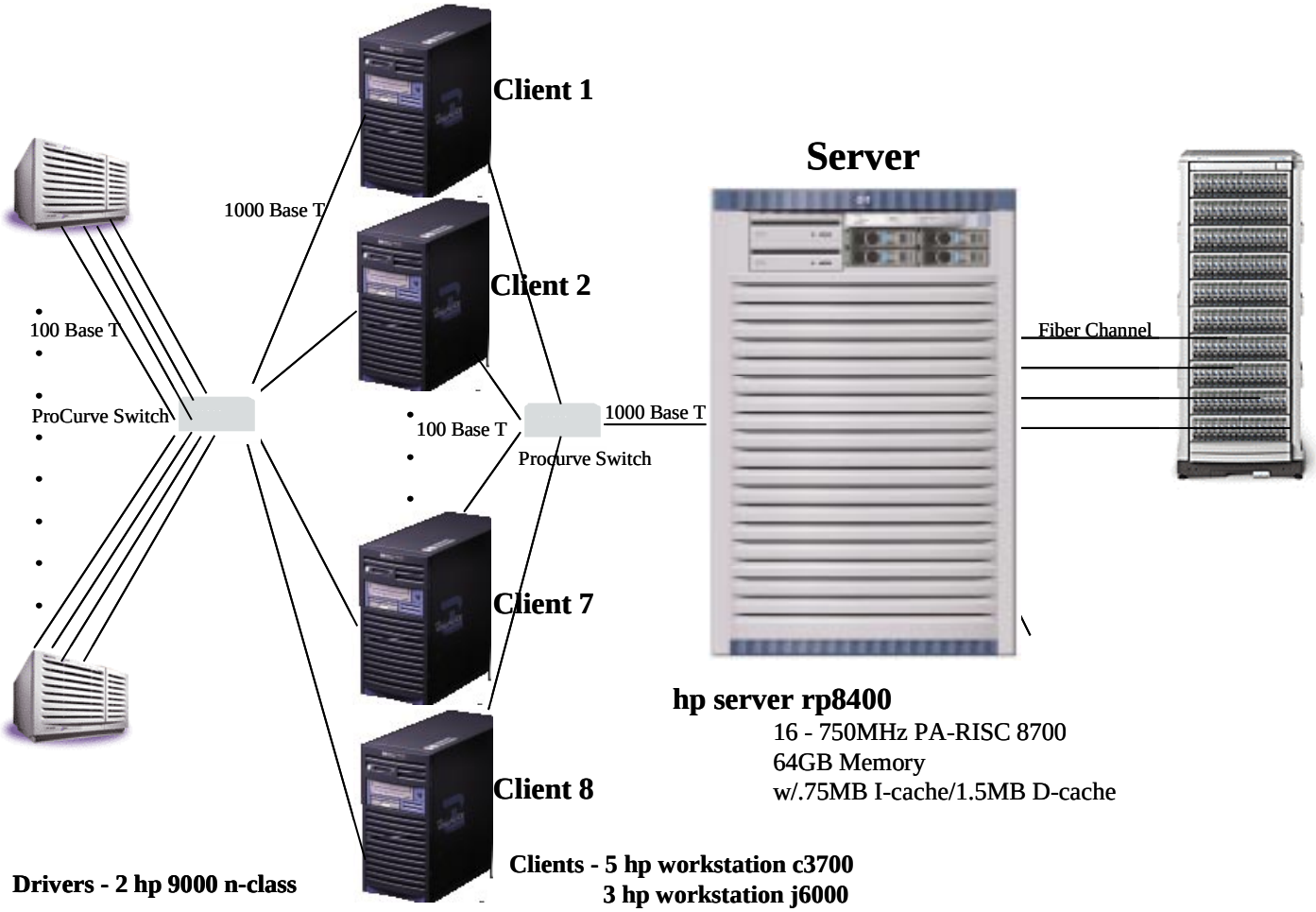


Figure 1.2: hp server rp8400 Benchmark Configuration



2 Clause 1 Related Items

2.1 Table Definitions

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

Appendix B describes the programs that define, create, and populate the Sybase Adaptive Server Enterprise v12.5 database for TPC-C® testing.

2.2 Physical Organization of Database

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

Space was allocated to Sybase Adaptive Server Enterprise v12.5 according to the data in section 5.2. The size of the database table space on each disk drive was calculated to provide even distribution of load across the disk drives.

2.3 Insert and Delete Operations

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

There were no restrictions on insert and delete operations to any tables.

2.4 Partitioning

While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C® benchmark, any such partitioning must be disclosed. Replication of tables, if used, must be disclosed. Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.

Partitioning, replication, and additional or duplicated attributes were not used in this implementation.

3 Clause 2 Related Items

3.1 Random Number Generation

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

The random number generator used can be found in the source appendix. It is from the book "The Art of Computer Systems Performance Analysis" by Raj Jain, page 443. The properties of this random number generator are documented in the book. It is a full-period multiplicative linear-congruential random number generator.

3.2 Input/Output Screen Layout

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

The screen layouts corresponded exactly to those in Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3, and 2.8.3 of the TPC-C® Standard Specification.

3.3 Priced Terminal Feature Verification

The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained. Although not specifically priced, the type and model of the terminals used for the demonstration in 8.1.3.3 must be disclosed and commercially available (including supporting software and maintenance).

The terminal features were verified by manually exercising each specification on an HP 712/80 workstation running an ANSI terminal emulator.

3.4 Presentation Manager or Intelligent Terminal

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

Application code running on the client implemented the TPC-C user interface. A listing of this code is included in Appendix A. Used capabilities of the terminal beyond basic ASCII entry and display were restricted to cursor positioning.

A presentation manager was not used.

Table 3.1: Transaction Statistics

Type	Item	Value
New Order	Home warehouse items	99.00%
	Remote warehouse items	1.00%
	Rolled back transactions	1.01%
	Average items per order	10.00
Payment	Home warehouse	85.00%
	Remote warehouse	15.00%
	Non primary key access	59.98%
Order Status	Non primary key access	59.97%
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.80%
	Payment	43.05%
	Order Status	4.05%
	Delivery	4.05%
	Stock Level	4.05%

3.5 Transaction Statistics

Table 3.1 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 require.

3.6 Queuing Mechanism

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

Delivery transactions were submitted to servers using the same TUXEDO mechanism that other transactions used. The only difference was that the call was asynchronous, i.e., control would return to the client process immediately and the deferred delivery part would complete asynchronously.

4 Clause 3 Related Items

4.1 Transaction System Properties (ACID)

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

The TPC Benchmark® C Standard Specification defines a set of transaction processing system properties that a system under test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation, and Durability (ACID). This section quotes the specification definition of each of these properties and describes the tests done as specified and monitored by the auditor to demonstrate compliance.

4.2 Atomicity

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

4.2.1 Completed Transaction

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

The values of w_ytd, d_ytd, c_balance, c_ytd_payment, and c_payment_cnt of a randomly selected warehouse, district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was committed. The values w_ytd, d_ytd, c_balance, c_ytd_payment, and c_payment_cnt were retrieved again. It was verified that all values had been changed appropriately.

4.2.2 Aborted Transaction

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

The values of w_ytd, d_ytd, c_balance, c_ytd_payment and c_payment_cnt of a randomly selected warehouse, district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was rolled back. The values of w_ytd, d_ytd, c_balance, c_ytd_payment, c_payment_cnt were retrieved again. It was verified that none of the values had changed.

4.3 Consistency

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

The TPC Benchmark C standard requires the System Under Test to meet the following 12 consistency conditions (c.f. *TPC Standard Specification, Clauses 3.3.2.1 to 3.3.2.12*):

1. the sum of the district balances in a warehouse is equal to the warehouse balance;
2. for each district, the next order-id minus one is equal to maximum order-id in the ORDER table and equal to the maximum new-order-id in the NEW-ORDER table;
3. for each district, the maximum order-id minus minimum order-id in the ORDER table plus one equals the number of rows in the NEW-ORDER table for that district;

4. for each district, the sum of the order-line counts equals the number of rows in the ORDER-LINE table for that district;
5. for each row in the ORDER table, the carrier-id is set to a null value only if there is a corresponding row in the NEW-ORDER table;
6. for each row in the ORDER table, the order-line count must equal the number of rows in the ORDER-LINE table for that order;
7. for any row in the ORDER-LINE table, the delivery date/time is set to a null value only if the corresponding row in the ORDER table has the carrier-id set to a null value;
8. for each warehouse, the year-to-date amount must equal the sum of the amounts in the HISTORY table for that warehouse;
9. for each district, the year-to-date amount must equal the sum of the amounts in the HISTORY table for that district;
10. for each customer, the balance must equal the sum of the order-line amount minus the sum of the history amount for that customer;
11. for each district, the total orders minus the total new-orders must equal the sum of the customer delivery count;
12. for any randomly selected customer, the balance plus the year-to-date payment must equal the sum of the order-line amount.

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

To demonstrate that consistency is maintained, conditions 1-4 were verified for a sample of warehouses before and after the durability tests.

4.4 Isolation

Operations of concurrent transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order.

*This property is commonly called **serializability**. Sufficient conditions must be enabled at either the system or application level to ensure serializability of transactions under any arbitrary mix of TPC-C transactions, unless otherwise specified by the transaction profile. The system or application must have full serializability enabled (i.e., repeated reads of the same rows within any committed transaction must return identical data when run concurrently with any arbitrary mix of TPC-C transactions), except in the case of Stock-Level transaction. For the Stock-Level transaction, the isolation requirement is relaxed to simply require that the transaction see only committed data.*

The TPC Benchmark C Standard (Revision 3) defines nine required tests to be performed to demonstrate that the required levels of transaction isolation are met.

For conventional locking schemes, isolation should be tested as described below. Systems that implement other isolation schemes may require different validation techniques. It is the responsibility of the test sponsor to disclose those techniques and the tests for them. If isolation schemes other than conventional locking are used, it is permissible to implement these tests differently provided full details are disclosed. (Examples of different validation techniques are shown in Isolation Test 7, Clause 3.4.2.7).

4.4.1 Isolation Test 1

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

The execution of the above test proceeded as follows:

1. An Order-Status transaction T0 was executed for a randomly selected customer, and the order returned was noted. T0 was committed
2. A New-Order transaction T1 was started for the same customer used in T0. T1 was stopped prior to COMMIT.
3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was allowed to complete and was committed.
5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the order inserted by T1.

This outcome demonstrates serialization of T2 before T1. It has equivalent validity to the outcome specified in the Standard which supposes T1 to be serialized before T2.

4.4.2 Isolation Test 2

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

The execution of the above test proceeded as follows:

1. An Order-Status transaction T0 was executed for a randomly selected customer and the order returned was noted. T0 was committed.
2. A New-Order transaction T1 with an invalid item number, was started for the same customer used in T0. T1 was stopped immediately prior to ROLLBACK.
3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was allowed to ROLLBACK.
5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the same order that T0 had returned.

4.4.3 Isolation Test 3

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

The execution of the above test proceeded as follows:

1. The D_NEXT_O_ID of a randomly selected district was retrieved.
2. A New-Order transaction T1 was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to COMMIT.
3. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.
4. T1 was allowed to complete. T2 completed and was committed.
5. The order number returned by T1 was the same as the D_NEXT_O_ID retrieved in step 1. The order number returned by T2 was one greater than the order number returned by T1.

6. The D_NEXT_O_ID of the same district was retrieved again. It had been incremented by two (i.e. it was one greater than the order number returned by T2).

4.4.4 Isolation Test 4

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

The execution of the above test proceeded as follows:

1. The D_NEXT_O_ID of a randomly selected district was retrieved.
2. A New-Order transaction T1, with an invalid item number, was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to ROLLBACK.
3. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.
4. T1 was allowed to roll back, and T2 completed and was committed.
5. The order number returned by T2 was the same as the D_NEXT_O_ID retrieved in step 1.
6. The D_NEXT_O_ID of the same district was retrieved again. It had been incremented by one (i.e. one greater than the order number returned by T2).

4.4.5 Isolation Test 5

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

The execution of the above test proceeded as follows:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C_BALANCE of the customer found in step 1 was retrieved.
3. A Delivery business transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the COMMIT of the database transaction corresponding to the district used in step 1.
4. A Payment transaction T2 was started for the same customer found in step 1. T2 waited.
5. T1 was allowed to complete. T2 completed and was committed.
6. The C_BALANCE of the customer found in step 1 was retrieved again. The C_BALANCE reflected the results of both T1 and T2.

4.4.6 Isolation Test 6

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

The execution of the above test proceeded as follows:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C_BALANCE of the customer found in step 1 was retrieved.

3. A Delivery business transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the ROLLBACK of the database transaction corresponding to the district used in step 1.
4. A Payment transaction T2 was started for the same customer found in step 1. T2 waited.
5. T1 was allowed to ROLLBACK. T2 completed and was committed.

The C_BALANCE of the customer found in step 1 was retrieved again. The C_BALANCE reflected the results of only T2.

4.4.7 Isolation Test 7

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

The execution of the above test proceeded as follows:

1. The I_PRICE of two randomly selected items X and Y were retrieved.
2. A New-Order transaction T2 with a group of items including items X and Y was started. T2 was stopped immediately after retrieving the prices of all items. The prices of items X and Y retrieved matched those retrieved in step 1.
3. A transaction T3 was started to increase the price of items X and Y by 10%.
4. T3 did not stall and no transaction was rolled back. T3 was committed.
5. T2 was resumed, and the prices of all items were retrieved again within T2. The prices of items X and Y matched those retrieved in step 1.
6. T2 was committed.
7. The prices of items X and Y were retrieved again. The values matched the values set by T3.

- Execution followed *Case D* of *Clause 3.4.2.7*.

4.4.8 Isolation Test 8

This test demonstrates isolation for phantom protection between New-Order and Order-Status transactions.

The execution of the above test proceeded as follows:

1. An Order-Status transaction T1 was started for a randomly selected customer.
2. T1 was stopped immediately after reading the order table for the selected customer. The most recent order for that customer was found.
3. A New-Order transaction T2 was started for the same customer. T2 completed and was committed without being blocked by T1.
4. T1 was resumed and the ORDER table was read again to determine the most recent order for the same customer. The order found was the same as the one found in step 2.
5. T1 completed and was committed.

4.4.9 Isolation Test 9

This test demonstrates isolation for phantom protection between New-Order and Delivery transactions.

The execution of the above test proceeded as follows:

1. The NO_D_ID of all new_ORDER rows for a randomly selected warehouse and district was changed to 11. The changes were committed.
2. A Delivery transaction T1 was started for the selected warehouse.
3. T1 was stopped immediately after reading the new_ORDER table for the selected warehouse and district. No qualifying row was found.
4. A New-Order transaction T2 was started for the same warehouse and district. T2 completed and was committed without being blocked by T1.
5. T1 was resumed and the new_ORDER table was read again. No qualifying row was found.
6. T1 completed and was committed.
7. The NO_D_ID of all new_ORDER rows for the selected warehouse and district was restored to the original value. The changes were committed.

4.5 Durability

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

List of single failures:

- *Permanent irrecoverable failure of any single durable medium containing TPC-C database tables or recovery log data.*
- *Instantaneous interruption (system crash / system hang) in processing which requires system reboot to recover.*
- *Failure of all or part of memory (loss of contents)…*

Specified durability tests were executed to demonstrate satisfaction of the durability requirements for this implementation of TPC Benchmark C. One durability test, described below, covering the following failure situations was performed:

- *Permanent irrecoverable failure of any single durable medium containing TPC-C database tables or recovery log data (Clause 3.5.3.1).*

This test was performed under a load of 114,400 users on the full-scale database built for 114,400 users. Another durability test, described below, combining the following failure situations was performed:

- *instantaneous interruption which requires system reboot [of processors] to recover. (Clause 3.5.3.2*)
- *failure of all or part of memory. (Clause 3.5.3.3).*

This test was performed under the full performance-measurement load of 114,400 users.

4.5.1 Loss of Data Disk or Log Disk

Because the log and data-storage devices are Redundant Disk Arrays which each function independently of the rest of the system in ensuring data integrity under loss and/or replacement of any individual disk drive (and other failures as well), integrity under such failure and replacement does not entail any interruption in processing. The test below validates the durability by demonstrating persistence of the results of transactions processed both before and during these failures, validating the durability upon database recovery (in this instance, forced) of transactions which completed before the failure and the non-effect of transactions which did not complete.

1. The D_NEXT_O_ID fields for all rows in the DISTRICT table were summed up to determine the initial count of the total number of orders (count1).
2. A test was initiated with 114,400 terminals. On the driver system, completed/rolled-back transactions (including New-Orders) were recorded in a "success" file.
3. After 15 minutes, one of the individual disks containing the Sybase recovery logs was unplugged from its array. After unplugging the disk, system processing continued normally given the array was configured with redundancy. After another 5 minutes, a disk was pulled from an array containing data. After unplugging the disk, system processing continued normally given the array was configured with redundancy.
4. The test was halted.
5. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 was 2 greater than the number of records for successful New Orders in the RTE “success” file which is within the “in-flight” limits allowed given 114,400 users.
6. Consistency checks 1-4 were run before and after the benchmark run and the results were verified.

4.5.2 Instantaneous Interruption and Loss of Memory

Instantaneous interruption and loss of memory tests were combined because the loss of power erases the contents of memory. This failure was induced while the benchmark was running by turning off the power supplies to the server.

1. The D_NEXT_O_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).
2. Transactions were started at full load. On the driver system, completed/rolled-back transactions (including New-Orders) were recorded in a "success" file.
3. After fifteen minutes the server system was de-powered.
4. The test was aborted on the driver.
5. The server system was restarted.
6. The database was restarted and a recovery performed using the transaction log.
7. The contents of the "success" file on the driver and the orders table were spot-compared to verify that records in the "success" file for completed New-Order transactions had corresponding records in the orders table.
8. The auditor verified that the total number of orders committed by the database was within the “in-flight” limit allowed based on the number of New Orders in the RTE “success” file.
9. Consistency checks 1-4 were run before and after the benchmark run and the results were verified.

5 Clause 4 Related Items

5.1 Initial Cardinality of Tables

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

The TPC-C database for this test was configured with 11,500 warehouses.

Table	Occurrences
Warehouse	11,500
District	115,000
Customer	345,000,000
History	345,000,000
Orders	345,000,000
New Orders	103,500,000
Order Line	3,450,159,276
Item	100,000
Stock	1,150,000,000

During the measurement only 11,440 warehouses and their associated data were accessed. This was confirmed using D_NEXT_O_1D and W_YTD as described in *Clause 4.2.2 Comment (2).*

5.2 Database and Growth Layout

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

Table 5.2 indicates the distribution of the database tables over the disks of the tested and priced systems**.**

Table 5.2: Disk Usage in Tested System

Disc layout:
==========

root, swap:

Use Disk Size (MB)
---
root /dev/vg00/lvol3 3666
project /dev/vg00/lvol4 4007
stand /dev/vg00/lvol1 292
tmp /dev/vg00/lvol5 961
swap /dev/vg00/lvol2 8192
swap /dev/vgtpcc/swap 57408

Dataspace for:

(NOTE: /dev/vgtpcc consists of 23 discs, 23-way striped
/dev/vg00 consists of disk c1t6d0, one 18GB spindle):

c4t0d2, c5t0d2, c6t0d2, c16t0d2, c17t0d2, c18t0d2, c12t0d2, c13t0d2,
c14t0d2, c8t0d2, c9t0d2, c10t0d2, c20t0d2, c21t0d2, c22t0d2, c24t0d2,
c25t0d2, c26t0d2, c32t0d2, c33t0d2, c34t0d2, c28t0d2, c29t0d2

master	/dev/vgtpcc/rmaster	1000
tmpdb1	/dev/vgtpcc/rtmpdb1	59
log1	/dev/rdsk/c30t0d0	32000
log2	/dev/rdsk/c30t0d1	32000
log3	/dev/rdsk/c30t0d2	32000
log4	/dev/rdsk/c30t0d3	32000
log5	/dev/rdsk/c30t0d4	32000
log6	/dev/rdsk/c30t0d5	32000
log7	/dev/rdsk/c30t0d6	32000
log8	/dev/rdsk/c30t0d7	23476
w_d_I	/dev/vgtpcc/rw_d_I	200
new_order1	/dev/vgtpcc/rnew_order1	4000
history1	/dev/vgtpcc/rhistory1	24000
order_line1	/dev/vtpcc/rorder_line1	12060
order_line2	/dev/vtpcc/rorder_line2	12060
order_line3	/dev/vtpcc/rorder_line3	12060
order_line4	/dev/vtpcc/rorder_line4	12060
order_line5	/dev/vtpcc/rorder_line5	12060
order_line6	/dev/vtpcc/rorder_line6	12060
order_line7	/dev/vtpcc/rorder_line7	12060
order_line8	/dev/vtpcc/rorder_line8	12060
order_line9	/dev/vtpcc/rorder_line9	12060
order_line10	/dev/vtpcc/rorder_line10	12060
order_line11	/dev/vtpcc/rorder_line11	12060
order_line12	/dev/vtpcc/rorder_line12	12060
order_line13	/dev/vtpcc/rorder_line13	12060
order_line14	/dev/vtpcc/rorder_line14	12060
order_line15	/dev/vtpcc/rorder_line15	12060

order_line16	/dev/vtpcc/rorder_line16	12060
order_line17	/dev/vtpcc/rorder_line17	12060
order_line18	/dev/vtpcc/rorder_line18	12060
order_line19	/dev/vtpcc/rorder_line19	12060
order_line20	/dev/vtpcc/rorder_line20	12060
order_line21	/dev/vtpcc/rorder_line21	12060
order_line22	/dev/vtpcc/rorder_line22	12060
order_line23	/dev/vtpcc/rorder_line23	12060
customer1	/dev/rdsk/c29t0d1	10800
customer2	/dev/rdsk/c28t0d1	10800
customer3	/dev/rdsk/c34t0d1	10800
customer4	/dev/rdsk/c33t0d1	10800
customer5	/dev/rdsk/c32t0d1	10800
customer6	/dev/rdsk/c26t0d1	10800
customer7	/dev/rdsk/c25t0d1	10800
customer8	/dev/rdsk/c24t0d1	10800
customer9	/dev/rdsk/c22t0d1	10800
customer10	/dev/rdsk/c21t0d1	10800
customer11	/dev/rdsk/c20t0d1	10800
customer12	/dev/rdsk/c10t0d1	10800
customer13	/dev/rdsk/c9t0d1	10800
customer14	/dev/rdsk/c8t0d1	10800
customer15	/dev/rdsk/c14t0d1	10800
customer16	/dev/rdsk/c13t0d1	10800
customer17	/dev/rdsk/c12t0d	10800
customer18	/dev/rdsk/c18t0d1	10800
customer19	/dev/rdsk/c17t0d1	10800
customer20	/dev/rdsk/c16t0d1	10800
customer21	/dev/rdsk/c6t0d1	10800
customer22	/dev/rdsk/c5t0d1	10800
customer23	/dev/rdsk/c4t0d1	10800
c_index1	/dev/vgtpcc/rc_index1	24000
orders1	/dev/vgtpcc/rorders1	16000
stock1	/dev/rdsk/c4t0d0	17430
stock2	/dev/rdsk/c5t0d0	17430
stock3	/dev/rdsk/c18t0d0	17430
stock4	/dev/rdsk/c16t0d0	17430
stock5	/dev/rdsk/c14t0d0	17430
stock6	/dev/rdsk/c6t0d0	17430
stock7	/dev/rdsk/c12t0d0	17430
stock8	/dev/rdsk/c33t0d0	17430
stock9	/dev/rdsk/c17t0d0	17430
stock10	/dev/rdsk/c8t0d0	17430
stock11	/dev/rdsk/c9t0d0	17430
stock12	/dev/rdsk/c10t0d0	17430
stock13	/dev/rdsk/c20t0d0	17430
stock14	/dev/rdsk/c21t0d0	17430
stock15	/dev/rdsk/c22t0d0	17430
stock16	/dev/rdsk/c24t0d0	17430
stock17	/dev/rdsk/c25t0d0	17430
stock18	/dev/rdsk/c26t0d0	17430
stock19	/dev/rdsk/c32t0d0	17430

stock20	/dev/rdsk/c13t0d0	17430
stock21	/dev/rdsk/c34t0d0	17430
stock22	/dev/rdsk/c28t0d0	17430
stock23	/dev/rdsk/c29t0d0	17430

The distribution of the database tables over the disk arrays of the priced system is an extension of the distribution described in Table 5.2; some ancillary details are mentioned in Appendix E. 60-day storage growth requirements are met with the unused space of this configuration. Figure 1.2 shows the configuration of the priced-system disks.

5.3 Data Model & Interfaces

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

Sybase Adaptive Server Enterprise v12.5 is a relational DBMS. SQL stored procedures were used, invoked through the Sybase Open Client DB-Library; the application code appears in Appendix A.

5.4 Partitions/Replications

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

No partitioning or replication was used.

5.5 Growth Requirements

Details of the 60 day space computations along with proof that the database is configured to sustain 8 hours for the dynamic tables (Order, Order-Line, and History) must be disclosed.

See Appendix E.

6 Clause 5 Related Items

6.1 Throughput

Measured tpmC must be reported.

Table 6.1: Measured tpmC

tpmC®	140,239.97

6.2 Response Time

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

Table 6.2: Response Times

Response Times	Average	90th %-ile	Maximum
New-Order	0.88s	1.63s	8.17s
Payment	0.84s	1.59s	8.13s
Order-Status	0.92s	1.67s	8.12s
Delivery (interactive portion)	0.09s	0.15s	0.70s
Delivery (deferred portion)	1.04s	1.80s	7.54s
Stock-Level	1.09s	1.95s	8.03s
Menu	0.003s	0.10s	1.04s

6.3 Keying and Think Times

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

Table 6.3: Keying Times

Keying Times	Minimum	Average	Maximum
New Order	18.02s	18.03s	18.05s
Payment	3.01s	3.02s	3.04s
Order Status	2.01s	2.02s	2.04s
Interactive Delivery	2.01s	2.02s	2.04s
Stock Level	2.01s	2.02s	2.04s

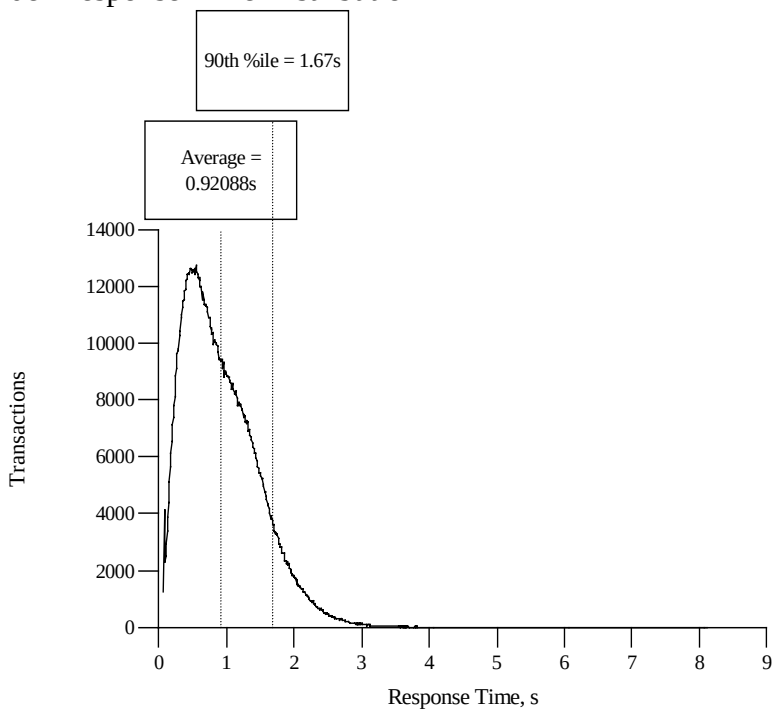
Table 6.4: Think Times

Think Times	Minimum	Average	Maximum
New Order	0.01s	12.14s	191.15s
Payment	0.01s	12.07s	201.55s
Order Status	0.01s	10.13s	154.64s
Interactive Delivery	0.01s	5.07s	77.46s
Stock Level	0.01s	5.06s	76.21s

6.4 Response Time Frequency Distribution Curves and Other Graphs

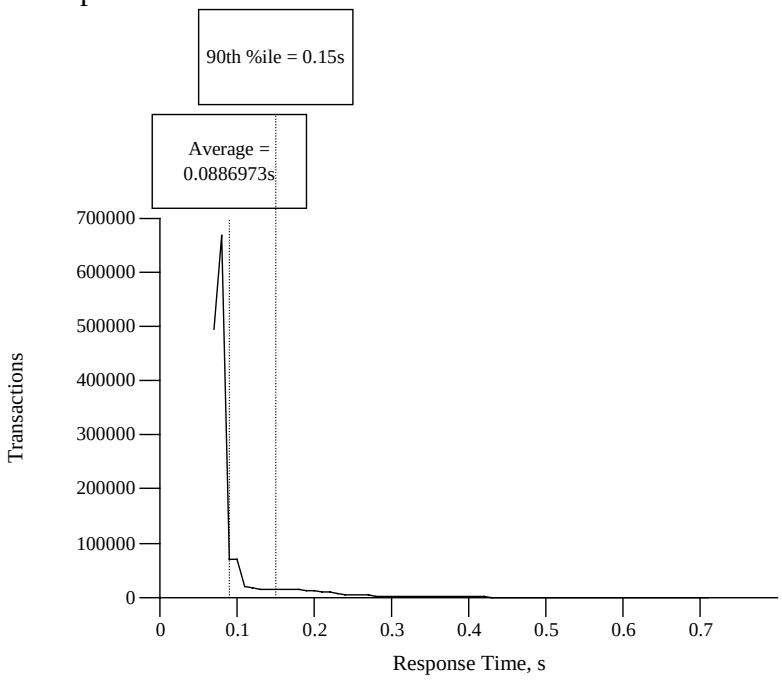
Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type. The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction. The Think Time frequency distribution curve (see Clause 5.6.3) must be reported for the New-Order transaction. A graph of throughput versus elapsed time (see Clause 5.6.5) must be reported for the New-Order transaction, and the measurement interval indicated.

Figure 6.1: New Order Response Time Distribution



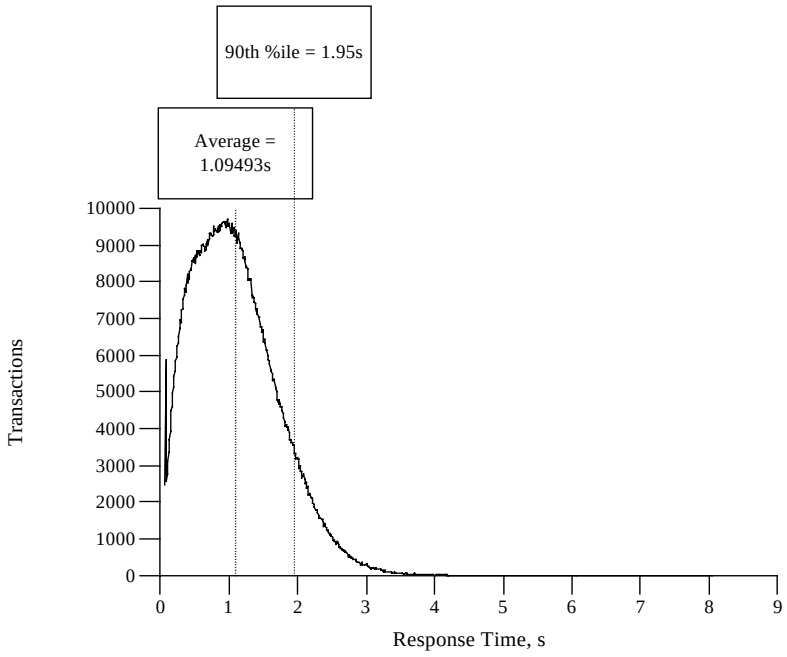
Response time frequency distribution for Order Status transaction

Figure 6.2: Payment Response Time Distribution



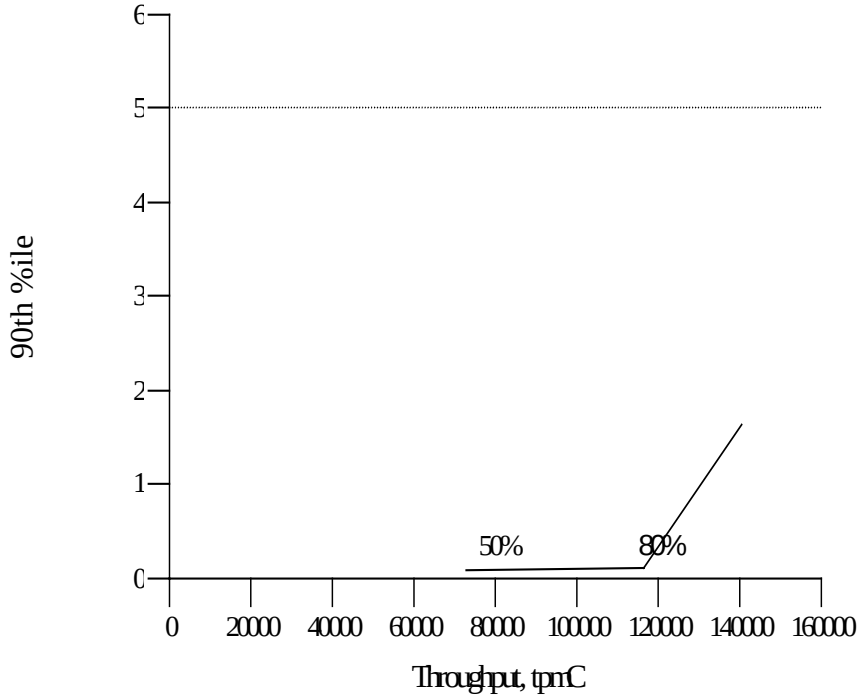
Response time frequency distribution for Delivery transaction

Figure 6.3: Order Status Response Time Distribution



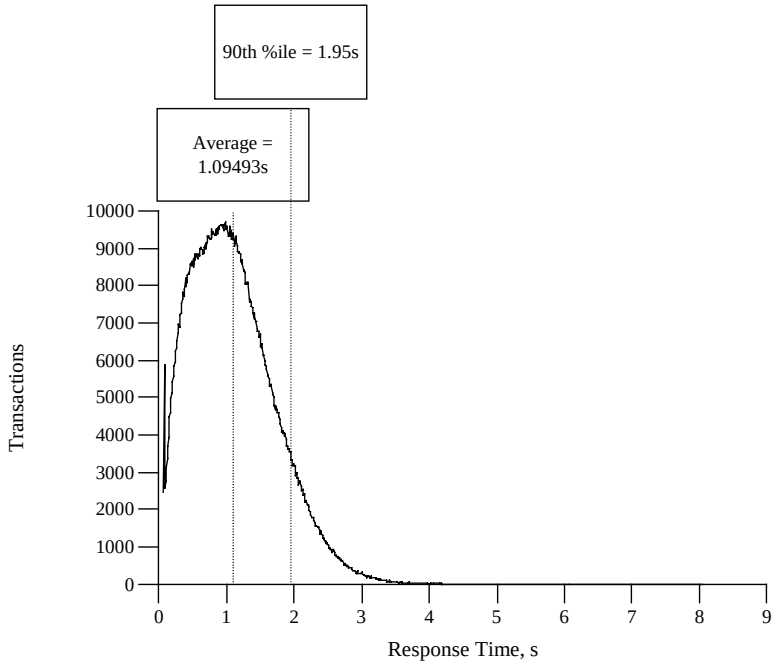
Response time frequency distribution for Stock Level transaction

Figure 6.4: (Interactive) Delivery Response Time Distribution



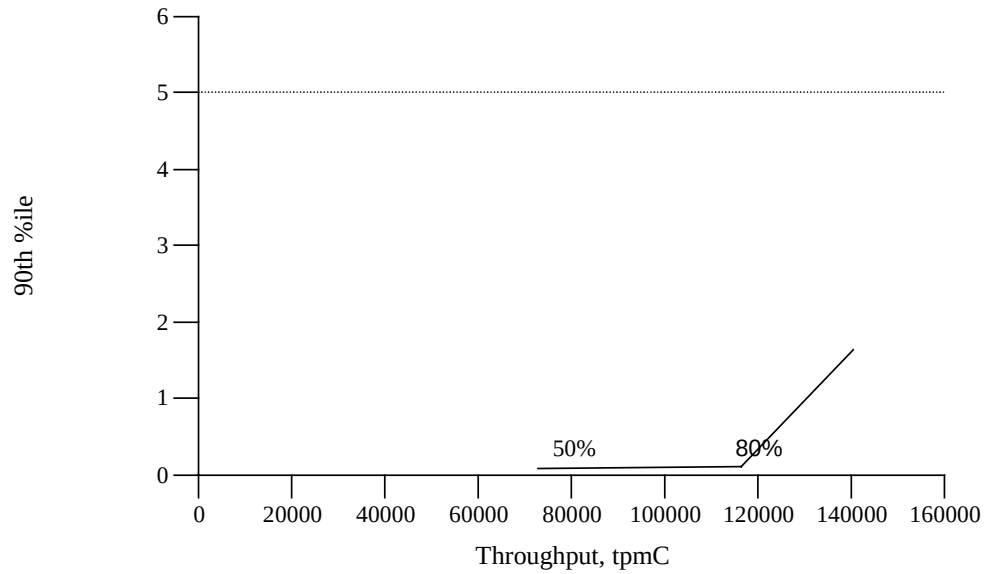
New Order response time versus Throughput

Figure 6.5: Stock Level Response Time Distribution



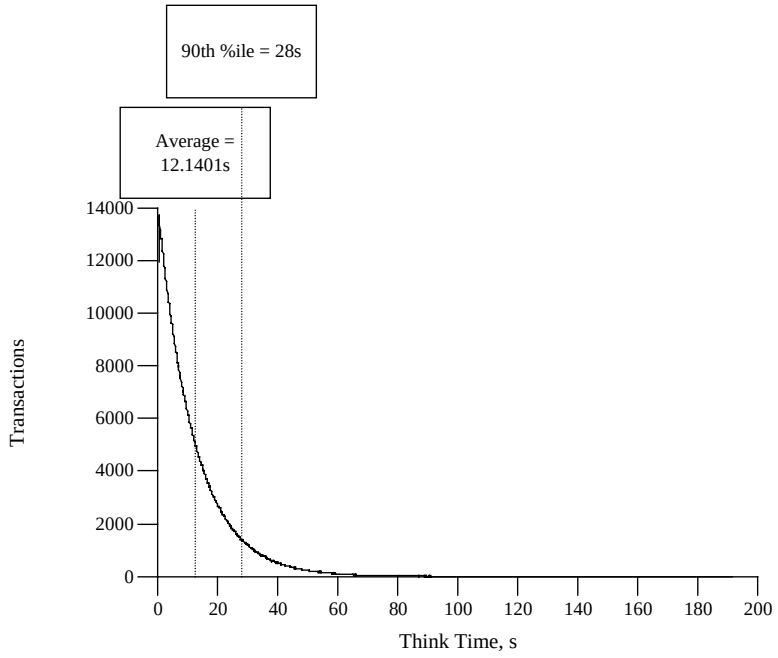
Response time frequency distribution for Stock Level transaction

Figure 6.6: Response Time Versus Throughput



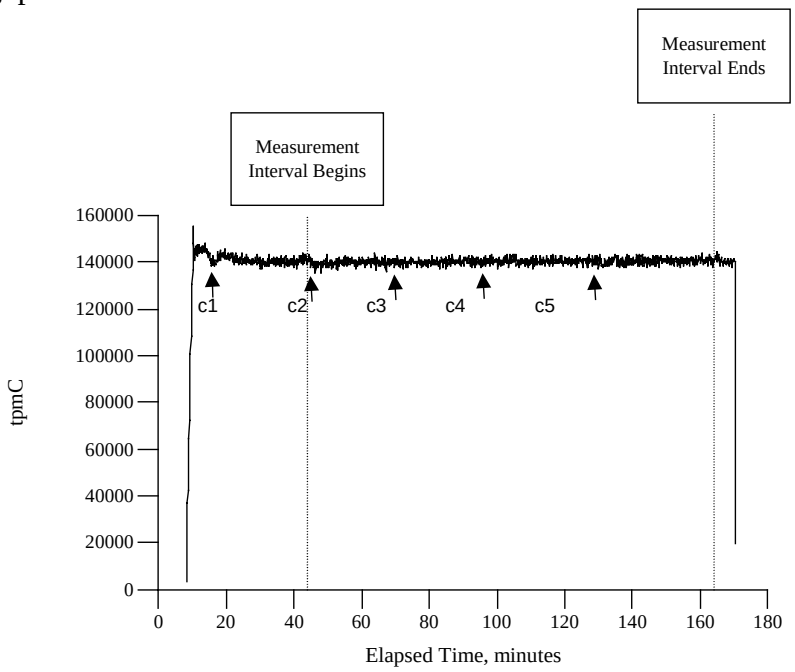
New Order response time versus Throughput

Figure 6.7: New Order Think Time Distribution



Think time frequency distribution for New Order transaction

Figure 6.8: Throughput Versus Time



Throughput of the New-Order transaction versus elapsed time

6.5 Steady State Determination

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval must be disclosed.

Synchronization techniques employed in the benchmark process ensure that all emulated users are logged into the client and have opened the application before submitting transactions. Once all users are connected, each pauses a random amount of time before submitting transactions. The pause time distribution is controlled by a benchmark input parameter. The ramp-up interval is discernible in the graph of throughput over time. The data reduction also tracks the user load and indicates the point in time at which all users have submitted at least one transaction. The throughput is observed to be steady within the systematic and statistical variability of the measurement after all users are submitting transactions. A checkpoint is initiated upon the end of ramp-up.

6.6 Work Performed During Steady State

A description of how the work normally performed during a sustained test (for example checkpointing, writing redo/undo log records, etc.), actually occurred during the measurement interval must be reported.

Modified database buffers migrated to disk on a least-recently-used basis independent of transaction commits. In addition, every block modification was protected by redo log records. These redo log records were written to the redo log buffer (in memory) and were flushed to a redo log file on disk either when the transaction committed or when the redo log buffer became full. However, due to the rapid commit during this benchmark, the redo log buffer was always flushed by a commit before it became full. Also, because many transactions were committing in a short period of time, a single flush of the redo log buffer resulted in many transactions' redo log data being written to disk. This is called group commit.

6.6.1 Checkpoint

During an Sybase Adaptive Server Enterprise v12.5 checkpoint, all modified blocks in the shared buffer cache which had not been written to disk since the last checkpoint are written to disk.

6.6.2 Checkpoint Conditions

Sybase Adaptive Server Enterprise v12.5 performs a checkpoint for the following conditions:

1. Automatically, at an interval calculated by Sybase Adaptive Server Enterprise on the basis of system activity and the recovery interval value in the system table *syscurconfigs.* The recovery interval determines checkpoint frequency by specifying the amount of time it should take the system to recover.
2. Upon an explicit **checkpoint** request in Transact-SQL

6.6.3 Checkpoint Implementation

For each benchmark measurement after all users are active, the script checkpoints issues a checkpoint and starts a background process, which sleeps and performs another checkpoint every 30 minutes. The recovery interval is configured large enough that no other checkpoints occur during the measurement.

6.7 Reproducibility

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

No reproducibility run needed in this revision of the benchmark.

6.8 Measurement Period Duration

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

The measurement interval was 120 minutes.

6.9 Regulation of Transaction Mix

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

The weighted selection method of *Clause 5.2.4.1* was used. The weights were not adjusted during the run.

6.10 Transaction Mix

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

Table 6.5: Transaction Mix

Type	Percentage
New Order	44.80%
Payment	43.05%
Order Status	4.05%
Delivery	4.05%
Stock Level	4.05%

6.11 Transaction Statistics

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

See Table 3.1

6.12 Checkpoint Count and Location

The number of checkpoints in the measurement interval, the time in seconds from the start of the measurement interval to the first checkpoint, and the Checkpoint Interval must be disclosed.

The number of checkpoints in the measurement interval was 4. These was one checkpoint during the rampup period. A new checkpoint happened every 30 minutes. The duration of the checkpoints were:

25 minutes, 58 seconds
29 minutes, 47 seconds
28 minutes, 36 seconds
29 minutes, 10 seconds
28 minutes, 56 seconds

7 Clause 6 Related Items

7.1 RTE Description

If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of what inputs (e.g., scripts) to the RTE had been used. The RTE input parameters, code fragments, functions, et cetera used to generate each transaction input field must be disclosed. Comment: The intent is to demonstrate the RTE was configured to generate transaction input data as specified in Clause 2.

The RTE (Remote Terminal Emulator) on the driver system was developed at Hewlett-Packard and is not commercially available. Appendix D lists RTE input parameters and code fragments used to generate each transaction input field.

For this instance of the TPC-C benchmark, 8 drivers and 8 clients were used. The drivers emulated users logged in to the clients. An overview of the benchmark software on the drivers, clients and server is shown in Figure 7.1.

The benchmark is started with the **run** command on the driver system. **Run** controls the overall execution of the benchmark. After reading a configuration file, **run** starts TUXEDO on the client, collects pre-benchmark audit information and inserts a timestamp into a database audit table. When all the initial steps are completed, **run** invokes another program, **driver,** to start the benchmark. As the benchmark completes, **run** shuts down TUXEDO and collects the benchmark results into a single location.

Driver is the heart of the benchmark software. It simulates users as they log in, execute transactions and view results. **Driver** collects response times for each transaction and saves them in a file for future analysis.

Qualify is the post-processing analysis program. It produces the numerical summaries and histograms needed for the disclosure report.

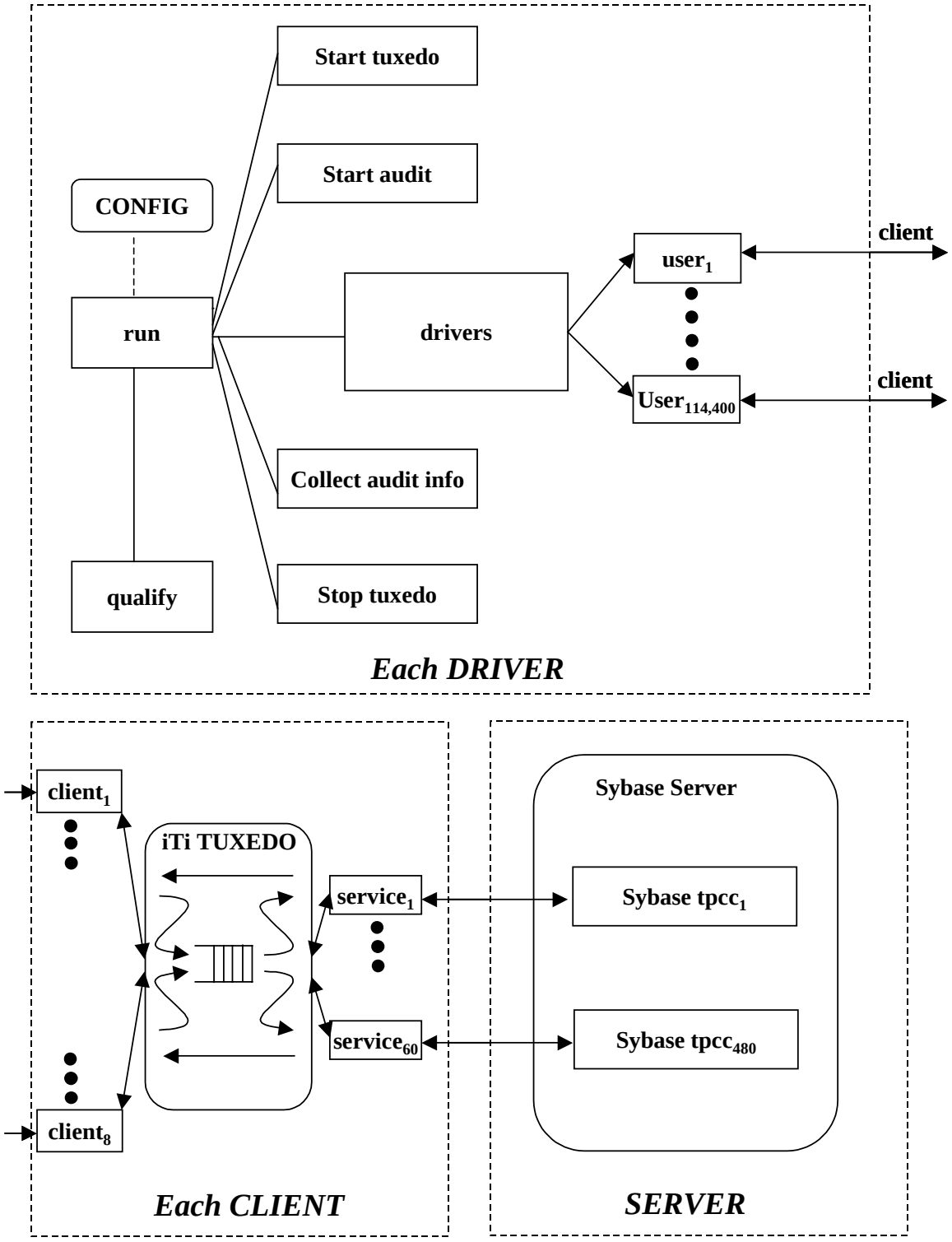


Figure 7.1: Benchmark Software

7.2 Emulated Components

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to the priced system.

In the benchmark configuration, the 114,400 simulated workstations connected to the clients over 8 100BT lans through a single hp procurve switch. In the priced system, the 114,400 workstations would connect to the clients via a combination of hubs and switches which eventually multiplexed down to 8 100BT lans.

7.3 Functional Diagrams

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

Figures 1.1 and 1.2 (in Chapter 1) show functional diagrams of the benchmark and configured systems. A description of the RTE and benchmark software is provided above.

7.4 Networks

The network configuration of both the tested and proposed services which are being represented and a thorough explanation of exactly which parts are being replaced with the Driver System must be disclosed.

Figures 1.1 and 1.2 (in Chapter 1) diagram the network configurations of the benchmark and configured systems, and represent the Driver connected via a 100BT switch replacing the workstations and HUBs/switches. The clients are connected via 100 Base-T to an 100BT/1000BT-Ethernet switch which in turn is connected via 1000BT-Ethernet to the SUT.

The bandwidth of the networks used in the tested/priced configurations must be disclosed.

Ethernet and 100 Base-T local area networks (LAN) with a bandwidth of 100 megabits per second are used in the tested/priced configurations. The 1000BT used has a bandwidth of 1000 megabits per second.

7.5 Client Substitution

Control Set Client Results

	New_Order	Payment	Order-Status	Delivery	Stock-Level
Client-1: (control set)					
average response time	0.45	0.41	0.49	0.07	0.64
90th percentile rsp.	0.81	0.78	0.87	0.08	1.21
transaction mix in %	44.77%	43.09%	4.04%	4.05%	4.05%
(count# out of 4428262)	1982438	1908105	179121	179444	179154
Menu response time	0	0	0	0	0
keying/think times	18.03/12.14	3.02/12.06	2.02/10.15	2.02/5.05	2.02/5.05
Client-2: (control set)					
average response time	0.85	0.82	0.89	0.08	1.06
90th percentile rsp.	1.4	1.37	1.45	0.1	1.76
transaction mix in %	44.72%	43.09%	4.06%	4.06%	4.08%
(count# out of 4350452)	1945336	1874716	176554	176421	177425
Menu response time	0	0	0	0	0
keying/think times	18.03/12.14	3.02/12.07	2.02/10.13	2.02/5.09	2.02/5.06
Client-3: (control set)					
average response time	0.31	0.28	0.35	0.07	0.51
90th percentile rsp.	0.59	0.56	0.65	0.08	1.03
transaction mix in %	44.75%	43.13%	4.05%	4.04%	4.03%
(count# out of 4457297)	1994717	1922271	180357	180135	179817
Menu response time	0	0	0	0	0
keying/think times	18.03/12.13	3.02/12.06	2.02/10.08	2.02/5.08	2.02/5.07
Client-4: (control set)					
average response time	1.36	1.32	1.4	0.07	1.57
90th percentile rsp.	2.08	2.04	2.13	0.08	2.43
transaction mix in %	44.73%	43.10%	4.05%	4.06%	4.05%
(count# out of 4258018)	1904698	1835270	172525	172962	172563
Menu response time	0	0	0	0	0
keying/think times	18.03/12.14	3.02/12.06	2.02/10.14	2.02/5.05	2.02/5.08
Client-5: (control set)					
average response time	0.98	0.94	1.02	0.07	1.19
90th percentile rsp.	1.57	1.54	1.62	0.08	1.93
transaction mix in %	44.76%	43.07%	4.06%	4.07%	4.05%
(count# out of 4329500)	1937746	1864843	175568	176102	175241
Menu response time	0	0	0	0	0
keying/think times	18.03/12.14	3.02/12.04	2.02/10.15	2.02/5.08	2.02/5.08
Control Set Aggregate:					
average response time	0.78	0.75	0.82	0.08	0.99
90th percentile rsp.	1.57	1.53	1.61	0.09	1.89
transaction mix in %	44.74%	43.10%	4.05%	4.06%	4.05%
(count# out of 21823529)	9764935	9405205	884125	885064	884200
Menu response time	0	0	0	0	0
keying/think times	18.03/12.14	3.02/12.06	2.02/10.13	2.02/5.07	2.02/5.07

Non-Priced Client Results

	New_Order	Payment	Order-Status	Delivery	Stock-Level
Client-6: (non-priced)					
average response time	0.73	0.68	0.75	0.19	0.92
90th percentile rsp.	1.23	1.18	1.26	0.3	1.56
transaction mix in %	44.75%	43.07%	4.07%	4.05%	4.05%
(count# out of 4376884)	1958780	1885322	177971	177414	177397
Menu response time	0	0	0	0	0
keying/think times	18.03/12.14	3.02/12.08	2.02/10.14	2.02/5.06	2.02/5.06
Client-7: (non-priced)					
average response time	1.41	1.36	1.43	0.13	1.61
90th percentile rsp.	2.1	2.05	2.13	0.14	2.42
transaction mix in %	44.75%	43.10%	4.06%	4.05%	4.05%
(count# out of 4251443)	1902359	1832431	172430	172214	172009
Menu response time	0	0	0	0	0
keying/think times	18.03/12.14	3.02/12.06	2.02/10.12	2.02/5.06	2.02/5.04
Client-8: (non-priced)					
average response time	0.69	0.64	0.71	0.19	0.87
90th percentile rsp.	1.18	1.13	1.21	0.31	1.51
transaction mix in %	44.76%	43.06%	4.05%	4.07%	4.06%
(count# out of 4386424)	1963233	1888845	177659	178439	178248
Menu response time	0	0	0	0	0
keying/think times	18.03/12.15	3.02/12.07	2.02/10.10	2.02/5.08	2.02/5.10
Non-Priced Aggregate:					
average response time	0.94	0.89	0.96	0.17	1.13
90th percentile rsp.	1.72	1.67	1.75	0.26	2.02
transaction mix in %	44.75%	43.08%	4.06%	4.06%	4.05%
(count# out of 13014751)	5824372	5606598	528060	528067	527654
Menu response time	0	0	0	0	0
keying/think times	18.03/12.14	3.02/12.07	2.02/10.12	2.02/5.07	2.02/5.07

	New_Order	Users	New_Orders/User	Ratio Deviation (%)
Client-1: (control set)	1982438	14300	138.63	
client-2: (control set)	1945336	14300	136.04	
Client-3: (control set)	1994717	14300	139.49	
Client-4: (control set)	1904698	14300	133.2	
Client-5: (control set)	1937746	14300	135.51	
(control set)	Average		136.573	0.000 (0.000%)
Client-6: (non-priced)	1958780	14300	136.98	0.405 (0.003%)
client-7: (non-priced)	1902359	14300	133.03	-3.540 (-0.026%)
client-8: (non-priced)	1963233	14300	137.29	0.717 (0.005%)
(non-priced)	Average		135.766	-0.806(-0.006%)

8 Clause 7 Related Items

8.1 System Pricing

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

The total 3-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.

Each priced configuration consists of an integrated system package, additional options, and components. Prices for all Hewlett-Packard products that are not provided by a third party quote are HP's US list prices. A one (1) year warranty is standard with all Hewlett-Packard products.

8.2 Support Pricing

The three year support pricing for Hewlett-Packard products is based on twenty-four (24) months of monthly support costs; thirty-six (36) months minus the twelve month warranty period. The Sybase Inc. support pricing is based on thirty-six (36) months of monthly support costs. The following support products were priced in the benchmark:

- HP four-hour on-site repair hardware support,
- HP telephone support for software and updates
- Sybase Inc. Standard Technical Support and,
- BEA TUXEDO Standard Technical Support

8.2.1 HP Hardware Support

HP's on-site support for hardware provides service 24 hour, seven day support.

8.2.2 HP Software Support

HP Software Support provides the following:

- Access to the HP Response Centers for fault isolation and problem solving assistance,
- Guaranteed two (2) hour call return, immediate response for critical calls,
- Electronic access to product and support information,
- Electronic access to software patches,
- Right-to-use and copy software updates.

8.3 Sybase Inc. Standard Technical Support

Sybase Inc. Standard Technical Support includes:

Product updates,

- A regular technical publication,
- Unlimited, toll-free telephone service to assist in product installation, syntax, and usage that is available 24 hours, seven days a week.

8.4 Availability

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

see below

8.5 Priced System Configuration

The hardware, software, and support/maintenance products priced in this benchmark are detailed on page *v*.

8.6 Throughput, Price/Performance, and Availability Date

A statement of the measured tpmC® *as well as the respective calculations for the 5-year pricing, price/performance (price/tpmC®).*

For Throughput and Price/Performance, please see page iv and v. The Price/Performance calculation spreadsheet appears on page v.

All hardware components in this test of the hp server rp8400 system are available September 18, 2001. HP-UX 11i 64-bit Base 0S, September 2001 is available now. Sybase Adaptive Server Enterprise v12.5 is available now.

9 Clause 9 Related Items

9.1 Auditor's Report

If the benchmark has been independently audited, then 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.

If audited, the auditor's attestation letter must be made readily available to the public as part of the Full Disclosure Report, but a detailed report from the auditor is not required.

This implementation of the TPC Benchmark® C on the hp server rp8400 was audited by Lorna Livingtree for Performance Metrics, Inc..

Lorna Livingtree
Performance Metrics, Inc.
137 Yankton Street, Suite 101
Folsom, CA 95630
U.S.A.
Phone: 916 985-1131
Fax: 916 985-1185

The attestation letter is shown on the following pages.

September 13, 2001

Andreas Hotea
AlwaysOn Infrastructure Solutions Division
Hewlett-Packard Company
19111 Pruneridge Avenue
Cupertino, CA 95014

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

Platform:	hp rp8400 Enterprise Server
Database Manager:	Sybase Adaptive Server Enterprise 12.5
Operating System:	HP-UX 11i 64-bit Base OS, September 2001
Transaction Manager:	Tuxedo 6.4

Server: hp rp8400				
CPU's	Memory	Disks	90% Response	TpmC
16 PA8700 @ 750 MHz	Main: 64 GB I-cache 768 KB D-cache 1536 KB	325 @ 18GB 35 @ 36GB	1.63 sec	140,239.97
8 Clients: 5 Controlled, 3 emulated				
PA8700 @750 Mhz	Main: 8 GB I-cache 768 KB D-cache 1536 KB	2 @ 18GB	na	Na
PA-8600 @552Mhz	Main: 8 GB I-cache 512 KB D-cache 1024 KB	1 @ 18 GB	Na	Na

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

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

- The database was properly scaled with 11,500 warehouses of which only 11,440 were active during the Measured Interval.
- The ACID properties were met.
- The durability data loss and log loss tests were performed on a fully scaled database with 114,400 users active.
- Input data was generated according to the specified percentages.
- Eight hours of mirrored log space was configured on the measured system.
- Eight hours of dynamic table growth space was configured on the measured system.
- The 60-day space calculation was verified. The measured configuration has sufficient storage to satisfy this requirement.
- Each emulated user started with a different random number seed.
- The NURand constants used for database load and at run time were 123 and 208.
- The steady state portion of the test was 2 hours (7200 seconds).
- One checkpoint was taken before the measured interval.
- Four checkpoints were taken during the measured interval.
- The system pricing was checked for major components and maintenance.

Auditor Notes:

1. All requirements of TAB ID 334 for client substitution were met and verified.
2. All requirements of TAB ID 309 for disk substitution were met and verified.
3. The database had been loaded with random alpha strings that were only lower case alphas. The error was corrected and a performance run demonstrated there was no significant impact on performance. The demonstration run was within 0.5% of the reported run and complaint in all respects.

Sincerely,

Lorna Livingtree
Auditor

10 Report Availability

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

Transaction Processing
Performance Council
c/o Shanley Public Relations
650 N. Winchester Blvd.
Suite 1
San Jose, CA 95128

or your local Hewlett-Packard sales office.

Appendix A Client/Server Source

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

A.1 Client Front-End

client/Makefile

```
#*****************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:23:52 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************

include ../buildenv.mk

#
# Makefile for compiling the client, batch-tpcc, and service code
#

TOB             = ${TUXDIR}/udataobj
P       = ${WORK_DIR}/src
I       = $(P)/lib
L       = $(P)/lib
D       = $(P)/driver
Q       = $(P)/que
S       = $(P)/client

SH_OPT      = -Wl,-a,shared
#OPT        =  -Wl,-a,archive_shared
OPT         = -Wl,-E -Wl,-a,archive_shared
#LDOPTS      = -ldld -a archive_shared
LDOPTS      = -E -ldld -a archive_shared

#TUXEDO     = -D_HPUX_SOURCE ${ROOTDIR}/include ${OPT}

SYB_INCLUDE= -I${SYBASE}/${SYBASE_ASE}/include -I${SYBASE}/${SYBASE_OCS}/include
VIS_INCLUDE= -I ${VISIGENIC}/include
TUX_INCLUDE= -I${ROOTDIR}/include
INCLUDE     = -I. -I$L

SH_CFLAGS  = $(BUILDFLAGS) ${SH_OPT} ${INCLUDE} ${TUX_INCLUDE}
CFLAGS          = $(BUILDFLAGS) ${OPT} ${INCLUDE} ${TUX_INCLUDE}
CFLAGS_SYB = $(BUILDFLAGS) ${OPT} ${INCLUDE} ${TUX_INCLUDE} ${SYB_INCLUDE}
CFLAGS_SQL = -Dunix -D_HPUX_SOURCE -DVG_UNIX ${OPT} ${INCLUDE} \
                        ${TUX_INCLUDE} ${SQL_INCLUDE} ${VIS_INCLUDE}


#`cat ${OL}/sysliblist`\
#

LDFLAGS_SYB= ${OPT} ${L}/tpc_lib.a -L${SYBASE}/${SYBASE_OCS}/lib -lsybdb -lm
LDFLAGS_SQL= ${OPT} ${L}/tpc_lib.a -L/opt/odbc/lib -lodbc -lm

PROGRAMS     = client service startup client_batch msg_server raw

all: ${PROGRAMS}

tpcc_client: client
                $(MV) client ${WORK_DIR}/bin/
others_sybase: raw startup client_batch_syb msg_server_syb
                $(MV) raw startup client_batch msg_server ${WORK_DIR}/bin
others_sqlserver: raw startup client_batch_sql msg_server_sql
                $(MV) raw startup client_batch msg_server ${WORK_DIR}/bin
service_sybase: service_syb
                $(MV) service ${WORK_DIR}/bin/
service_sqlserver: service_sql
                $(MV) service ${WORK_DIR}/bin/

${S}/sybase/transaction.o: ${S}/sybase/transaction.c
                $(CC) ${CFLAGS_SYB} $(L)/tpc_lib.a -c ${S}/sybase/transaction.c;
${S}/sqlserver/transactionb.o: ${S}/sqlserver/transactionb.c
                $(CC) ${CFLAGS_SQL} $(L)/tpc_lib.a -c ${S}/sqlserver/transactionb.c;


raw: raw.o
                cc ${CFLAGS} raw.o $(L)/tpc_lib.a -o raw

startup: startup.o $(L)/tpc_lib.a
                cc ${SH_CFLAGS} startup.o $(L)/tpc_lib.a -o startup
                chmod a+rw startup

# Warning: can't use +pd because kernel will use this size to extend
# DATA pregion when break/sbreak is called.
# Force shared libc to avoid to both libc data from the archived and shared version.
client: client.o  tux_transaction.o $(L)/tpc_lib.a
                cp ${TOB}/lic.sdk ${TOB}/lic.txt
                ${ROOTDIR}/bin/buildclient -v -o client \
        -f "$(BUILDFLAGS) $(OPT) -Wl,+pi 256K -Wl,+pd 4K -Wl,-R 0x100000 \
           client.o tux_transaction.o $(L)/tpc_lib.a" \
                -l "-lnsl  -lm -Wl,-ashared -lc"
                cp ${TOB}/lic.sdk ${TOB}/lic.txt

service_syb: service.o ${S}/sybase/transaction.o $(L)/tpc_lib.a
                cp ${TOB}/lic.sdk ${TOB}/lic.txt
                if [ -f /project/iti/lib/libgp.a ] ; then \
           mv /project/iti/lib/libgp.a /project/iti/lib/libgp.a.cc_service ; \
        fi
                ${ROOTDIR}/bin/buildserver -v -b shm \
              -s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC \
              -o service \
              -f "-Wl,+pi 4M -Wl,+pd 256K \
                 -Wl,-R 0x400000 -Wl,-D 0x40100000 \
                 service.o transaction.o $L/tpc_lib.a \
                 ${SYBASE}/${SYBASE_OCS}/lib/libsybdb.a -lm" \
              -l "-lnsl ";
                 if [ -f /project/iti/lib/libgp.a.cc_service ] ; then \
                    cp /project/iti/lib/libgp.a.cc_service /project/iti/lib/libgp.a ; \
                 fi
                 cp ${TOB}/lic.sdk ${TOB}/lic.txt

service_sql: service.o ${S}/sqlserver/transactionb.o $(L)/tpc_lib.a
                cp ${TOB}/lic.sdk ${TOB}/lic.txt
                ${ROOTDIR}/bin/buildserver -v -b shm \
              -s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC \
              -o service \
              -f "service.o transactionb.o $L/tpc_lib.a \
                /vsbuild/v1.10/build/com/obj/inst/libodbc.sl"
                 cp ${TOB}/lic.sdk ${TOB}/lic.txt

client_batch_syb: $(D)/driver.o $(D)/generate.o ${S}/sybase/transaction.o \
                        $(Q)/dummy_que.o $(L)/tpc_lib.a $(L)/server_default.o
                 $(CC) $(D)/driver.o $(D)/generate.o transaction.o \
                   $(Q)/dummy_que.o $(L)/server_default.o $(L)/tpc_lib.a \
                   ${LDFLAGS_SYB} -o client_batch;

client_batch_sql: $(D)/driver.o $(D)/generate.o ${S}/sqlserver/transactionb.o \
                          $(Q)/dummy_que.o $(L)/tpc_lib.a $(L)/server_default.o
```

```
                        $(CC) $(D)/driver.o $(D)/generate.o transactionb.o \
                $(Q)/dummy_que.o $(L)/server_default.o $(L)/tpc_lib.a \
                ${LDFLAGS_SQL} -o client_batch;

msg_server_syb: $(Q)/msg_server.o  ${S}/sybase/transaction.o $(L)/tpc_lib.a
                $(CC) $(Q)/msg_server.o transaction.o ${LDFLAGS_SYB} -o msg_server;

msg_server_sql: $(Q)/msg_server.o  ${S}/sqlserver/transactionb.o $(L)/tpc_lib.a
                $(CC) $(Q)/msg_server.o transactionb.o ${LDFLAGS_SQL} -o msg_server;

clean:
                rm -f *.o

clobber: clean
                rm -f ${PROGRAMS}

install: ${PROGRAMS}
                cp ${PROGRAMS} ${WORK_DIR}/bin
```

client/make_no_pbo

```
#!/usr/bin/csh

setenv CCOPTS "-Ae +DA2.0 +DS2.0 +pgm client"
make -f Makefile tpcc_client

if (${DATABASE} == "sybase") then
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 +pgm service"
   make -f Makefile service_sybase
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 -DHPUX -D_REENTRANT"
   make -f Makefile others_sybase
else if (${DATABASE} == "oracle") then
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 +pgm service"
   make -f Makefile service_oracle
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 -DHPUX -D_REENTRANT"
   make -f Makefile others_oracle
else if (${DATABASE} == "sqlserver") then
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 +pgm service"
   make -f Makefile service_sqlserver
   setenv CCOPTS "-Ae +DA2.0 +DS2.0 -DHPUX -D_REENTRANT"
   make -f Makefile others_sqlserver
endif
```

client/client.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 16:52:43 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
/*****************************************************************
History
941101 JVM Fixed login screen to detect broken connection (used to loop)
941013 JVM Added audit strings to the login form
941013 VM modified the getfield procedure to add digit and char check
       according to the field type.
941014 VM added the status_msg routine to display transaction results.
941015 VM added zip routine to format zip codes and phone routine
       to format phone numbers.
*****************************************************************/
```

```
#include "iobuf.h"
#include "tpcc.h"
#include <signal.h>

#define until(c)  while(!(c))


/* a generic transaction variable. */
generic_trans generic_transaction;
generic_trans *trans=&generic_transaction;

/* global variables set up during initialization */
int user;
ID warehouse;
ID district;

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

   /* setup the transactions */
   key = setup(argc, argv);

   /* repeat until done */
   while (key != '9' && key != EOF)
     {

      /* get the menu choice */
      key = menu_read();

      /* process according to the choice */
      switch(key)
         {
         case '1': key = neworder(&trans->neworder); break;
         case '2': key = payment(&trans->payment); break;
         case '3': key = ordstat(&trans->ordstat); break;
         case '4': key = delivery(&trans->delivery); break;
         case '5': key = stocklev(&trans->stocklev); break;
         case EOF: break;
         case '9': break;
         default:    msgline("Please enter a valid menu choice");
         }
      }

   /* done */
   cleanup();
   }
```

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

Neworder form processing

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

define_iobuf(neworder_form, 900);

int neworder(trans)
   neworder_trans *trans;
   {
   int key;
   display(neworder_form);
   key = neworder_read(trans);
   if (key != ENTER) return key;
   neworder_transaction(trans);
```

```
    neworder_write(trans);
    return key;
    }


int neworder_read(trans)
    neworder_trans *trans;
    {
    int i;
    int field;
    int key;
    int ol;

    /* Our warehouse number is fixed */
    trans->W_ID = warehouse;
    trans->D_ID = EMPTY_NUM;

    /* assume nothing set yet */
    trans->C_ID = EMPTY_NUM;
    for (i=0; i<15; i++)
        {
        trans->item[i].OL_I_ID = EMPTY_NUM;
        trans->item[i].OL_QUANTITY = EMPTY_NUM;
        trans->item[i].OL_SUPPLY_W_ID = EMPTY_NUM;
        }

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 47))
        retry: switch (field)
            {

            case 1: key = read_number(4, 29, &trans->D_ID, 2);
                    break;

            case 2: key = read_number(5, 12, &trans->C_ID, 4);
                    break;

            case 3: case 6: case 9: case 12: case 15:
            case 18: case 21: case 24: case 27: case 30:
            case 33: case 36: case 39: case 42: case 45:
                ol = (field - 3) / 3;
                key = read_number(9+ol, 2, &trans->item[ol].OL_SUPPLY_W_ID,6);
                break;

            case 4: case 7: case 10: case 13: case 16:
            case 19: case 22: case 25: case 28: case 31:
            case 34: case 37: case 40: case 43: case 46:
                ol = (field - 3) / 3;
                key = read_number(9+ol,10, &trans->item[ol].OL_I_ID, 6);
                break;

            case 5: case 8: case 11: case 14: case 17:
            case 20: case 23: case 26: case 29: case 32:
            case 35: case 38: case 41: case 44: case 47:
                ol = (field - 3) / 3;
                key = read_number(9+ol, 45, &trans->item[ol].OL_QUANTITY, 2);
                break;
            }

    /* abort the screen if requested */
    if (key != ENTER)
        return key;

    /* calculate how many items were entered */
    for (i=15; i>0; i--)
        if ((trans->item[i-1].OL_I_ID != EMPTY_NUM) ||
                    (trans->item[i-1].OL_SUPPLY_W_ID != EMPTY_NUM) ||
                    (trans->item[i-1].OL_QUANTITY != EMPTY_NUM)) break;
    trans->O_OL_CNT = i;

    /* make sure all necessary fields are filled in */
    if (trans->D_ID == EMPTY_NUM)
        {field=1; msgline("Please specify district"); goto retry;}
    if (trans->C_ID == EMPTY_NUM)
        {field=2; msgline("Please specify customer id"); goto retry;}
    if (trans->O_OL_CNT == 0)
        {field=3; msgline("Please enter at least one orderline"); goto retry;}
    for (i=0; i<trans->O_OL_CNT; i++)
        {
        if (trans->item[i].OL_SUPPLY_W_ID == EMPTY_NUM)
            {field=i*3+3; msgline("Please enter supply warehouse"); goto retry;}
        if (trans->item[i].OL_I_ID == EMPTY_NUM)
            {field=i*3+4; msgline("Please enter Item id"); goto retry;}
        if (trans->item[i].OL_QUANTITY == EMPTY_NUM
         || trans->item[i].OL_QUANTITY <= 0)
            {field=i*3+5; msgline("Please enter quantity > 0"); goto retry;}
        }

    /* decide if they were all local */
    for (i=0; i<trans->O_OL_CNT; i++)
        if (trans->item[i].OL_SUPPLY_W_ID != trans->W_ID) break;
    trans->all_local = (i == trans->O_OL_CNT);

    /* display number of order lines */
    number(6, 42, trans->O_OL_CNT, 2);

    msgline("");
    flush();
    return key;
    }



neworder_write(t)
    neworder_trans *t;
    {
    int i;
    MONEY amount, total_amount, cost;

    /* Rev. 3.3 error checking: both of the following branches are
     * skipped.  We'll go to status and print an error message.
     */

    /* CASE: invalid item, display only these values */
    if (t->status == E_INVALID_ITEM)
        {
        text(5, 25, t->C_LAST);
        text(5,52,  t->C_CREDIT);
        number(6, 15, t->O_ID, 8);
        }

    /* CASE: everything OK, display everything */
    else if (t->status == OK)
        {
        text(5, 25, t->C_LAST);
        text(5,52,  t->C_CREDIT);
        number(6, 15, t->O_ID, 8);
        date(4, 61, t->O_ENTRY_D);
        real(5, 64, t->C_DISCOUNT * 100, 5, 2);
        real(6, 59, t->W_TAX*100, 5, 2);
        real(6, 74, t->D_TAX*100, 5, 2);

        total_amount = 0;
        for (i=0; i < t->O_OL_CNT; i++)
            {

            /* keep track of amount of each line and total */
            amount = t->item[i].I_PRICE * t->item[i].OL_QUANTITY;
            total_amount += amount;
```

```
        /* display the item line */
        text(9+i, 19, t->item[i].I_NAME);
        number(9+i, 51, t->item[i].S_QUANTITY, 3);
        position(9+i, 58);   pushc(t->item[i].brand_generic);
        money(9+i, 62, t->item[i].I_PRICE, 7);
        money(9+i, 71, amount, 8);
        }

    /* Clear the screen of any empty input fields */
    clear_screen();

    /* display the total cost */
    text(24, 63, "Total:");
    cost = total_amount * (1 - t->C_DISCOUNT) * (1 + t->W_TAX + t->D_TAX);
    money(24, 71, cost, 9);
    }

  /* display the status message */
  status(24, 1, t->status);
  }


neworder_setup()
  {
  int item;
  iobuf *old;

  /* start with an empty form */
  reset(neworder_form);

  /* redirect the data to a special menu buffer */
  old = out_buf;  out_buf = neworder_form;

  /* clear the iobuf below the menu */
  position(3,1);
  clear_screen();

  /* set up all the field labels */
  text(3, 36, "New Order");
  text(4, 1, "Warehouse:");
  number(4, 12, warehouse, 6);
  text(4, 19, "District:");
  empty(4, 29, 2);
  text(4, 55, "Date:");
  text(5, 1, "Customer:");
  empty(5, 12, 4);
  text(5, 19, "Name:");
  text(5, 44, "Credit:");
  text(5, 57, "Disc.:");
  text(6, 1, "Order Number:");
  text(6, 25, "Number of Lines:");
  text(6, 52, "W_Tax:");
  text(6, 67, "D_Tax:");
  text(8, 2, "Supp_W Item_Num   Item_Name");
  text(8, 45, "Qty Stock   B/G     Price  Amount");

  /* display blank fields for each item */
  for (item = 1; item <= 15; item++)
    {
    empty(8+item, 2, 6);
    empty(8+item, 10, 6);
    empty(8+item, 45, 2);
    }

  trigger();

  /* restore to the previous I/O buffer */
  out_buf = old;
  }
```

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

Payment form processing

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

define_iobuf(payment_form, 400);


int payment(trans)
  payment_trans *trans;
  {
  int key;
  display(payment_form);
  key = payment_read(trans);
  if (key != ENTER)  return key;
  payment_transaction(trans);
  payment_write(trans);
  return key;
  }


payment_setup()
  {
  int item;
  iobuf *old;

  /* start with an empty form */
  reset(payment_form);

  /* redirect the data to a special menu buffer */
  old = out_buf;  out_buf = payment_form;

  /* clear the iobuf below the menu */
  position(3,1);
  clear_screen();

  /* set up all the field labels */
  text(3, 38, "Payment");
  text(4, 1, "Date:");
  text(6, 1, "Warehouse:");
  number(6, 12, warehouse, 6);
  text(6, 42, "District:");
  empty(6, 52, 2);
  text(11, 1, "Customer:");
  empty(11, 11, 4);
  text(11, 17, "Cust-Warehouse:");
  empty(11, 33, 6);
  text(11, 40, "Cust-District:");
  empty(11, 54, 2);
  text(12, 1, "Name:");
  empty(12, 29, 16);
  text(12, 50, "Since:");
  text(13, 50, "Credit:");
  text(14, 50, "%Disc:");
  text(15, 50, "Phone:");
  text(17, 1, "Amount Paid:");
  empty(17, 23, 8);
  text(17, 37, "New Cust-Balance:");
  text(18, 1, "Credit Limit:");
  text(20, 1, "Cust-Data:");
  trigger();

  out_buf = old;
```

```
    }


int payment_read(t)
    payment_trans *t;
    {
    int i;
    int field;
    int key;

    /* Our warehouse number is fixed */
    t->W_ID = warehouse;
    t->C_ID = EMPTY_NUM;
    t->D_ID = EMPTY_NUM;
    t->C_W_ID = EMPTY_NUM;
    t->C_D_ID = EMPTY_NUM;
    t->H_AMOUNT = EMPTY_FLT;
    t->C_LAST[0] = '\0';

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 6))
        retry: switch (field)
            {

            case 1: key = read_number(6, 52, &t->D_ID, 2);
                    break;

            case 2:
                /* if last name specified, skip this field */
                if (t->C_LAST[0] != '\0')
                    break;

                /* read in the customer id */
                key = read_number(11, 11, &t->C_ID, 4);

                /* if specified, don't allow last name to be entered */
                if (t->C_ID != EMPTY_NUM)
                    {
                    blanks(12, 29, 16);
                    t->C_LAST[0] = '\0';
                    }

                /* refresh the C_LAST underlines, if possibly needed */
                else if (t->C_LAST[0] == '\0')
                    empty(12, 29, 16);
                break;

            case 3: key = read_number(11, 33,  &t->C_W_ID, 6);
                    break;

            case 4: key = read_number(11, 55, &t->C_D_ID, 2);
                    break;

            case 5:
                /* skip this field if C_ID was already specified */
                if (t->C_ID != EMPTY_NUM)
                    break;

                /* read in the customer last name */
                key = read_text(12, 29, t->C_LAST, 16);

                /* if specified, don't allow c_id to be entered */
                if (t->C_LAST[0] != '\0')
                    {
                    blanks(11, 11, 4);
                    t->C_ID = EMPTY_NUM;
                    }

                /* refresh the C_ID underlines, if possibly needed */
                else if (t->C_ID == EMPTY_NUM)
                    empty(11, 11, 4);
                break;

            case 6: key = read_money(17, 23, &t->H_AMOUNT, 8);
                    break;
            }

    /* if Aborted, then done */
    if (key != ENTER)
        return key;


    /* Make sure all the fields were entered */
    if (t->D_ID == EMPTY_NUM)
        {field=1; msgline("Please enter district id"); goto retry;}
    if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
        {field=2; msgline("C_ID or C_LAST must be entered"); goto retry;}
    if (t->C_W_ID == EMPTY_NUM)
        {field=3; msgline("Please enter customer's warehouse"); goto retry;}
    if (t->C_D_ID == EMPTY_NUM)
        {field=4; msgline("please enter customer's district"); goto retry;}
    if (t->H_AMOUNT == EMPTY_FLT)
        {field=6; msgline("Please enter payment amount"); goto retry;}
    if (t->H_AMOUNT <= 0)
                {field=6; msgline("Please enter a positive payment"); goto retry;}


    t->byname = (t->C_ID == EMPTY_NUM);
    msgline("");
    flush();
    return key;
    }


payment_write(t)
    payment_trans *t;
    {

    /* if errors, display a message and quit */
    if (t->status != OK)
        {
        status(24, 1, t->status);
        return;
        }

    /* display the screen */
    date(4, 7, t->H_DATE);
    text(7, 1, t->W_STREET_1);
    text(7, 42, t->D_STREET_1);
    text(8, 1, t->W_STREET_2);
    text(8, 42, t->D_STREET_2);
    text(9, 1, t->W_CITY);
    text(9, 22, t->W_STATE);
    zip(9, 25, t->W_ZIP);
    text(9, 42, t->D_CITY);
    text(9, 63, t->D_STATE);
    zip(9, 66, t->D_ZIP);
    number(11, 11, t->C_ID, 4);
    text(12, 9, t->C_FIRST);
    text(12, 26, t->C_MIDDLE);
    text(12, 29, t->C_LAST);
    date_only(12, 58, t->C_SINCE);
    text(13, 9, t->C_STREET_1);
    text(13, 58, t->C_CREDIT);
    text(14, 9, t->C_STREET_2);
    real(14, 58, t->C_DISCOUNT*100, 5, 2);  /* percentage or fraction? */
    text(15, 9, t->C_CITY);
    text(15, 30, t->C_STATE);
    zip(15, 33, t->C_ZIP);
```

```
    phone(15, 58, t->C_PHONE);
    money(17, 17, t->H_AMOUNT,14);
    money(17, 55, t->C_BALANCE, 15);
    money(18, 17, t->C_CREDIT_LIM, 14);

    /* Display cust data if bad credit. */
    if (t->C_CREDIT[0] == 'B' && t->C_CREDIT[1] == 'C')
        long_text(20, 12, t->C_DATA, 50);
    }



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

ORDSTAT form processing

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

define_iobuf(ordstat_form, 300);



int ordstat(t)
    ordstat_trans *t;
    {
    int key;
    display(ordstat_form);
    key = ordstat_read(trans);
    if (key != ENTER)  return key;
    ordstat_transaction(trans);
    ordstat_write(trans);
    return key;
    }


ordstat_setup()
    {
    int item;
    iobuf *old;

    /* start with an empty form */
    reset(ordstat_form);

    /* redirect the data to a special menu buffer */
    old = out_buf;  out_buf = ordstat_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 35, "Order-Status");
    text(4, 1, "Warehouse:");
    number(4, 12, warehouse, 6);
    text(4, 19, "District:");
    empty(4, 29, 2);
    text(5, 1, "Customer:");
    empty(5, 11, 4);
    text(5, 18, "Name:");
    empty(5, 44, 16);
    text(6, 1, "Cust-Balance:");
    text(8, 1, "Order-Number");
    text(8, 26, "Entry-Date:");
    text(8, 60, "Carrier-Number:");
    text(9, 1, "Supply-W");
    text(9, 14, "Item-Num");
    text(9, 25, "Qty");
    text(9, 33,"Amount");
    text(9, 45, "Delivery-Date");

    trigger();

    /* done */
    out_buf = old;
    }



int ordstat_read(t)
    ordstat_trans *t;
    {
    int i;
    int field;
    int key;

    /* Our warehouse number is fixed */
    t->W_ID = warehouse;
    t->C_ID = EMPTY_NUM;
    t->D_ID = EMPTY_NUM;
    t->C_LAST[0] = '\0';

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 3))
        retry: switch (field)
            {

            case 1: key = read_number(4, 29, &t->D_ID, 2);
                    break;

            case 2:
                /* if last name specified, skip this field */
                if (t->C_LAST[0] != '\0')
                    break;

                /* read in the customer id */
                key = read_number(5, 11, &t->C_ID, 4);

                /* if specified, don't allow last name to be entered */
                if (t->C_ID != EMPTY_NUM)
                    {
                    blanks(5, 44, 16);
                    t->C_LAST[0] = '\0';
                    }

                /* refresh the C_LAST underlines, if possibly needed */
                else if (t->C_LAST[0] == '\0')
                    empty(5, 44, 16);
                break;

            case 3:
                /* skip this field if C_ID was already specified */
                if (t->C_ID != EMPTY_NUM)
                    break;

                /* read in the customer last name */
                key = read_text(5, 44, t->C_LAST, 16);

                /* if specified, don't allow c_id to be entered */
                if (t->C_LAST[0] != '\0')
                    {
                    blanks(5, 11, 4);
                    t->C_ID = EMPTY_NUM;
                    }

                /* refresh the C_ID underlines, if possibly needed */
```

```
            else if (t->C_ID == EMPTY_NUM)
                empty(5, 11, 4);
            break;
        }

    /* if Aborted, then done */
    if (key != ENTER)
        return key;


    /* ensure all the necessary fields were entered */
    if (t->D_ID == EMPTY_NUM)
        {field=1; msgline("Please enter district id"); goto retry;}
    if (t->C_ID == EMPTY_NUM && t->C_LAST[0] == '\0')
        {field=2;  msgline("C_ID or C_LAST must be entered"); goto retry;}


    t->byname = (t->C_ID == EMPTY_NUM);
    msgline("");
    flush();
    return key;
    }


ordstat_write(t)
    ordstat_trans *t;
    {
    int i;

    /* if errors, display a status message and quit */
    if (t->status != OK)
        {
        status(24, 1, t->status);
        return;
        }

    /* display the results */
    number(5, 11, t->C_ID, 4);
    text(5, 24, t->C_FIRST);
    text(5, 41, t->C_MIDDLE);
    text(5, 44, t->C_LAST);
    money(6, 15, t->C_BALANCE, 10);
    number(8, 15, t->O_ID, 8);
    date(8, 38, t->O_ENTRY_DATE);
    if (t->O_CARRIER_ID > 0)
        number(8, 76, t->O_CARRIER_ID, 2);

    for (i=0; i< t->ol_cnt; i++)
        {
        number(i+10, 3, t->item[i].OL_SUPPLY_W_ID, 6);
        number(i+10, 14, t->item[i].OL_I_ID, 6);
        number(i+10, 25, t->item[i].OL_QUANTITY, 2);
        money(i+10, 32, t->item[i].OL_AMOUNT, 9);
        date_only(i+10, 47, t->item[i].OL_DELIVERY_DATE);
        }
    }




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

delivery form processing

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

```
define_iobuf(delivery_form, 300);



int delivery(t)
    delivery_trans *t;
    {
    int key;
    display(delivery_form);
    key = delivery_read(trans);
    if (key != ENTER)  return key;
    delivery_enque(trans);
    delivery_write(trans);
    return key;
    }


delivery_setup()
    {
    int item;
    iobuf *old;

    /* start with an empty form */
    reset(delivery_form);

    /* redirect the data to a special menu buffer */
    old = out_buf;  out_buf = delivery_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 38, "Delivery");
    text(4, 1, "Warehouse:");
    number(4, 12, warehouse, 6);
    text(6, 1, "Carrier Number:");
    empty(6, 17, 2);

    trigger();

    /* done */
    out_buf = old;
    }



int delivery_read(t)
    delivery_trans *t;
    {
    int i;
    int field;
    int key;

    /* Our warehouse number is fixed */
    t->W_ID = warehouse;
    t->O_CARRIER_ID = EMPTY_NUM;

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 1))
        retry: switch (field)
            {
            case 1: key = read_number(6, 17, &t->O_CARRIER_ID, 2);
                    break;
            }

    /* if Aborted, then done */
    if (key != ENTER)
        return key;
```

```
    /* Must enter the carrier id */
    if ((t->O_CARRIER_ID == EMPTY_NUM) ||
                (t->O_CARRIER_ID < 1) ||
                (t->O_CARRIER_ID > 10))
        {field=1; msgline("Please enter a Carrier Number within 1 and 10"); goto retry; }

    /* clear the message line */
    msgline("");
    flush();
    return key;
    }


delivery_write(t)
    delivery_trans *t;
    {
    if (t->status == OK)
        text(8, 1, "Execution Status: Delivery has been queued");
    else
        status(8, 1, t->status);
    }



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

stocklev form processing

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

define_iobuf(stocklev_form, 300);


int stocklev(t)
    stocklev_trans *t;
    {
    int key;
    display(stocklev_form);
    key = stocklev_read(trans);
    if (key != ENTER)  return key;
    stocklev_transaction(trans);
    stocklev_write(trans);
    return key;
    }


stocklev_setup()
    {
    int item;
    iobuf *old;

    /* start with an empty form */
    reset(stocklev_form);

    /* redirect the data to a special menu buffer */
    old = out_buf;  out_buf = stocklev_form;

    /* clear the iobuf below the menu */
    position(3,1);
    clear_screen();

    /* set up all the field labels */
    text(3, 35, "Stock-Level");
    text(4, 1, "Warehouse:");
    number(4, 12, warehouse, 6);
    text(4, 19, "District:");
    number(4, 29, district, 2);
    text(6, 1, "Stock Level Threshold:");
    empty(6, 24, 2);
    text(8, 1, "low stock");

    trigger();

    /* done */
    out_buf = old;
    }



int stocklev_read(t)
    stocklev_trans *t;
    {
    int field;
    int key;

    t->W_ID = warehouse;
    t->D_ID = district;
    t->threshold = EMPTY_NUM;

    /* Process fields until done */
    for (field = 1; field > 0; field = next_field(field, key, 1))
        retry: switch (field)
            {
            case 1: key = read_number(6, 24, &t->threshold, 2);
                    break;
            }

    /* if Aborted, then done */
    if (key != ENTER)
        return key;


    /* make sure the necessary fields were entered */
    if ((t->threshold == EMPTY_NUM) ||
        (t->threshold < 10) ||
                    (t->threshold > 20))

        {field=1; msgline("Please enter a threshold within 10 and 20"); goto retry; }

    /* clear the message line */
    msgline("");
    flush();
    return key;
    }


stocklev_write(t)
    stocklev_trans *t;
    {
    if (t->status == OK)
        number(8, 12, t->low_stock, 3);
    else
      status(10, 1, t->status);
    }



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

login form processing

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

```
int login()
    {
    int field;
    int key;
    char auditstr[21];
    int w_id, d_id;

    /* assume the default values */
    w_id = warehouse;
    d_id = district;
    auditstr[0] = '\0';

    /* display the login menu */
    position(1,1);  clear_screen();
    text(3, 30, "Please login.");
    text(5,5,"Warehouse:");
    number(5, 16, w_id, 6);
    text(5, 24, "District:");
    number(5, 34, d_id, 2);
    text(15, 5, "Audit String:");
    text(15, 19, CLIENT_AUDIT_STRING);
    empty(16, 19, 20);
    trigger();

    /* Get values until done */
    for (field = 1; field > 0;  field = next_field(field, key, 3))
        retry: switch (field)
            {
            case 1:
                key = read_number(5, 16, &w_id, 6, Num);
                break;

            case 2:
                key = read_number(5, 34, &d_id, 2, Num);
                break;

            case 3:
                key = read_text(16, 19, auditstr, 20);
                break;
            }

    if (key != ENTER)
        return EOF;

    if (w_id == EMPTY_NUM && warehouse == EMPTY_NUM)
        {
        msgline("You must enter a warehouse id");
        field =1;
        goto retry;
        }

    if (d_id == EMPTY_NUM && district == EMPTY_NUM)
        {
        msgline("You must enter a district id");
        field = 2;
        goto retry;
        }

    if (w_id != EMPTY_NUM)
        warehouse = w_id;
    if (d_id != EMPTY_NUM)
        district = d_id;

    /* done */
    flush();
    return key;
    }
```

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

menu form processing

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

menu_setup()
    {

    /* display the menu on the iobuf -- never erased */
    position(1, 1);
    clear_screen();
    string("(1)New-Order (2)Payment (3)Order-Status ");
    string("(4)Delivery (5)StockLevel (9)Exit");
    }

int menu_read()
    {
    position(1, 1);
    trigger();
    return getkey();
    }



int next_field(current, key, max)
    int current;
    int key;
    int max;
    {

    if (key == BACKTAB)
        if (current == 1)    return max;
        else                 return current-1;
    else if (key == TAB)
        if (current == max)  return 1;
        else                 return current+1;
    else
        return 0;
    }



msgline(str)
    char *str;
    {
    position(24, 1);
    clear_screen();
    string(str);
    flush();  /* Needed? */
    }



int setup(argc, argv)
    int argc;
    char **argv;
    {
    int key;
```

```
    /* Ignore SIGPIPE, since they occur normally */
    signal(SIGPIPE, SIG_IGN);

    /* get the user, warehouse and district numbers */
    warehouse = EMPTY_NUM;
    district = EMPTY_NUM;
    key = login();
    user = warehouse*DIST_PER_WARE + district + 1;

    /* set up the forms */
    menu_setup();
    neworder_setup();
    payment_setup();
    ordstat_setup();
    delivery_setup();
    stocklev_setup();

    /* connect to the delivery queue */
    delivery_init(user);

    /* connect to the transaction processor */
    transaction_begin(user);

    return key;
    }


cleanup()
    {

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

    /* detach from the delivery queue */
    delivery_done();

    /* clear the screen */
    position(1, 1);
    clear_screen();
    flush();
    }


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

Screen Output Routines

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


number(row, col, n, width)
    int row;
    int col;
    int n;
    int width;
    {
    char str[81];
    fmt_num(str, n, width);
    text(row, col, str);
    }



real(row, col, x, width, dec)
    int row;
    int col;
    double x;
    int width;
    int dec;
    {
    char str[81];
    fmt_flt(str, x, width, dec);
    text(row, col, str);
    }

date(row, col, date_str)
    int row;
    int col;
    char *date_str;
    {
    text(row, col, date_str);
    }


date_only(row, col, date_str)
    int row;
    int col;
    char *date_str;
    {
    date_str[10] = '\0';
    text(row, col, date_str);
    }

money(row, col, x, width)
    int row;
    int col;
    double x;
    int width;
    {
    char str[81];
    fmt_money(str, x, width);
    text(row, col, str);
    }


long_text(row, col, str, width)
    int row, col, width;
    char *str;
    {
    int pos;

    /* repeat until the entire string is written out */
    for (pos = width; *str != '\0';  str++, pos++)
        {

        /* if at end of line, position the cursor to next line */
        if (pos >= width)
            {
            position(row, col);
            pos = 0;
            row++;
            }

        /* output the next character */
        pushc(*str);
        }
    }


text(row, col, str)
    int row;
    int col;
    char str[];
    {
    position(row, col);
    string(str);
```

```
    }


phone(row, col,str)
      int row;
      int col;
      char *str;
      {

       char temp[30];

       fmt_phone(temp,str);
       text(row,col,temp);

       }


zip(row, col, str)
    int row;
    int col;
    char *str;
    {
    char temp[30];

    fmt_zip(temp,str);
    text(row,col,temp);
    }

empty(row, col, len)
    int row;
    int col;
    int len;
    {
    position(row, col);
    while (len-- > 0)
        pushc('_');
    }


blanks(row, col, len)
    int row, col, len;
    {
    position(row, col);
    while (len-- > 0)
        pushc(' ');
    }


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

   if (status == OK)
      string("Transaction Committed");
   else if (status == E_INVALID_ITEM)
      string("Item number is not valid");
   /* Do the rev. 3.3 error checking here.       */
   else if (status == E_INVALID_INPUT)
      string("Invalid input, transaction not executed");
   else
      {
      string("Rollback -- ");
      number(row, col+30, status, 5);
      }
   }

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

ASCII terminal control

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


trigger()
/*************************************************************************
trigger sends a turnaround sequence to let the driver know to send input
*************************************************************************/
   {
   pushc(TRIGGER);
   }


position(row, col)
/*************************************************************************
position positions the cursor at the given row and column
*************************************************************************/
   int row;
   int col;
   {
   pushc(ESCAPE);
   pushc('[');
   if (row >= 10)
      pushc('0' + row/10);
   pushc('0' + row%10);
   pushc(';');
   if (col >= 10)
      pushc('0' + col/10);
   pushc('0' + col%10);
   pushc('H');
   }

clear_screen()
/*************************************************************************
clear_screen clears the iobuf from cursor position to end of iobuf
*************************************************************************/
   {
   pushc(ESCAPE);
   pushc('[');
   pushc('J');
   }




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

Screen Input Routines

*************************************************************************
*************************************************************************/
#define funny(key)  (key != ENTER && key !=TAB && key != BACKTAB)


read_number(row, col, n, width)
/*************************************************************************
read_number reads an integer field
*************************************************************************/
   int row;
   int col;
   int *n;
```

```
    int width;
    {
    char temp[81];
    int key;
    int err;
    debug("read_number: row=%d  col=%d  width=%d n=%d \n",row, col,width,*n);

    /* generate the current characters */
    fmt_num(temp, *n, width);
    err = NO;

    /* repeat until a valid number or a funny key is pressed */
    for (;;)
        {
        /* Let the user edit the field */
        key = getfield(row, col, temp, width, Num);
                    if (funny(key)) return key;

        /* convert the field to a number */
        *n = cvt_num(temp);
        if (*n != INVALID_NUM) break;

        msgline("Invalid digit entered");
                    pushc(BELL);
        err = YES;
        }

    /* display the new number */
    number(row, col, *n, width);
    if (err) msgline("");
    debug("read_number: n=%d  key=%d\n", *n, key);
    return key;
    }


int read_money(row, col, m, width)
    int row;
    int col;
    double *m;
    int width;
    {
    char temp[81];
    int key;
    int err;

    err = NO;
    fmt_money(temp, *m, width);

    /* repeat until a valid number or a funny key is pressed */
    for (;;)
        {
        key = getfield(row, col, temp, width, Money);
                    if (funny(key)) return key;

        *m = cvt_money(temp);
        if (*m != INVALID_FLT)  break;

        msgline("Please enter amount $99999.99");
        pushc(BELL);
        err = YES;
                    }

    money(row, col, *m, width);
    if (err) msgline("");
    return key;
    }


int read_real(row, col, x, width, dec)
    int row, col, width;
    double *x;
    {
    char temp[81];
    int key;
    int err;

    /* generate the current characters */
    fmt_flt(temp, *x, width, dec);
    err = NO;

    /* repeat until a valid number or a funny key is pressed */
    for (;;)
                {
        key = getfield(row, col, temp, width);
                    if (funny(key)) return key;

        /* convert the field to a number */
        *x = cvt_flt(temp);
                    if (*x != INVALID_FLT) break;

        msgline("Please enter a valid floating pt number");
        pushc(BELL);
        err = YES;
        }

    /* display the new number */
    real(row, col, *x, width, dec);
    if (err) msgline("");

    return key;
    }


int read_text(row, col, s, width)
    int row, col, width;
    char *s;
    {
    char temp[81];
    int key;
    int i;

    /* generate the current characters */
    fmt_text(temp, s, width);

    /* let the user edit the field */
    key = getfield(row, col, temp, width, Text);
    if (funny(key)) return key;

    /* Strip off leading and trailing space characters */
    cvt_text(temp, s);

    /* redisplay the current text */
    fmt_text(temp, s, width);
    text(row, col, temp);

    return key;
    }


int getfield(row, col, buf, width,ftype)
    int row, col, width;
    char buf[];
    FIELD_TYPE ftype;

    {
    int pos, key;
```

```
    debug("getfield: width=%d  buf=%*s\n", width, width, buf);

    /* go to the beginning of the field */
    position(row, col);
    pos = 0;

    /* repeat until a special control character is pressed */
    for (;;)
        {

        /* get the next character */
        key = getkey();

        /* CASE: Add to buf if it fits and Is it a valid character ? */
        if (pos < width  && valid_char(key, ftype))
            {
            buf[pos] = key;
            pos++;
            pushc(key);
            }

        /* CASE: char is BACKSPACE.  Erase last character. */
        else if (key == BACKSPACE && pos > 0)
            {
            pos--;
            buf[pos] = '_';
            pushc(BACKSPACE);
            pushc('_');
            pushc(BACKSPACE);
            }

        /* CASE: enter, tab, backtab, ^c.  Exit loop */
        else if (key==ENTER || key==TAB || key==BACKTAB || key==CNTRLC
                || key == EOF)
            break;

        else if (key=='\031')   /* for debugging, let ^X == ENTER */
            {key=ENTER; break;}

        /* Otherwise, ignore the character and beep */
        else
            pushc(BELL);
        }

    debug("getfield: final key: %d  buf=%*s\n", key, width, buf);
    return key;
    }




int valid_char(key, ftype)
/*************************************************************************
valid_char is true if the key is valid for this type of field
*************************************************************************/
    int key;
    FIELD_TYPE ftype;
    {
    int valid;
    switch(ftype)
                    {
                    case Num : valid = (isdigit(key) || key == '-' || key == '.');
                                    break;

        case Text : valid = (isprint(key) || key == '.');
                                    break;

        case Money : valid = (isdigit(key) || key == '-' || key == '.'
                                                || key == '$' || key == '.');
                                    break;

        default     : valid = NO;
                                    break;
        }

    return valid;
    }
```

A.2 Tpc_lib Source

lib/tpcc.h

```
/*************************************************************************
 @(#) Version: A.10.10 $Date: 97/12/15 14:01:49 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.

  History
  @022801 ML  Added Client Substitution Report for TPC-C TAB ID 334.

*************************************************************************/
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED
#include <values.h>


/* The auditor can define these 20 char strings to be anything */
#define DRIVER_AUDIT_STRING  "driver audit string"
#define CLIENT_AUDIT_STRING  "client audit string"

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

#include <stdio.h>

typedef int ID;          /* All id's */
typedef double MONEY;    /* Large integer number of cents */
typedef char TEXT;       /* Add an extra byte for null terminator */
typedef double TIME;     /* Elapsed seconds from start of run (float?) */
typedef int COUNT;       /* integer numbers of things */
typedef double REAL;     /* real numbers */
typedef int LOGICAL;     /* YES or NO */
typedef struct {         /* days and seconds since Jan 1, 1900 */
    int day;             /* NULL represented by negative day */
    int sec;
    } DATE;

/* Macro to convert time of day to TIME */
#include <time.h>
extern struct timeval start_time;
#define elapsed_time(t) ( ((t)->tv_sec - start_time.tv_sec) +  \
                ((t)->tv_usec - start_time.tv_usec) / 1000000.0 )

typedef enum {Num,Money,Text,Time,Real,Date} FIELD_TYPE;   /* screen field types */


/* Various TPCC constants */
#define W_ID_LEN        4
#define D_ID_LEN        2
```

```
#define C_ID_LEN        4
#define I_ID_LEN        6
#define OL_QTY_LEN      2
#define PMT_LEN         7
#define C_ID_LEN        4
#define C_LAST_LEN      16
#define CARRIER_LEN     2
#define THRESHOLD_LEN   2
#define DIST_PER_WARE   10
#define CUST_PER_DIST   3000
#define ORD_PER_DIST    3000
#define MAXITEMS        100000
#define MAX_DIGITS      3       /* # of digits of the NURand number selected
                                   to generate the customer last name */
#define MAXWAREHOUSE    2000    /* maximum # of warehouses - scaling factor */
#define LOADSEED        42 /* # of digits of the NURand number selected

/*************************************************************************/
/* database identifiers and populations                                  */
/*************************************************************************/

  int no_warehouse;                     /*  scaling factor    */
  int no_item;                          /*  100000            */
  int no_dist_pw;                       /*      10            */
  int no_cust_pd;                       /*    3000            */
  int no_ord_pd;                        /*    3000            */
  int no_new_pd;                        /*     900            */
  int tpcc_load_seed;                   /*     900            */

/* fields to add to each transaction for acid testing */
#define ACID_STUFF      \
    char acid_txn[2];  \
    int  acid_timing;  \
    int  acid_action;  \
    FILE *acid_res

typedef struct {
    ID OL_SUPPLY_W_ID;
    ID OL_I_ID;
    TEXT I_NAME[24+1];
    COUNT OL_QUANTITY;
    COUNT S_QUANTITY;
    MONEY I_PRICE;
    char brand_generic;
    } neworder_item;

typedef struct {
    int status;
    LOGICAL all_local;
    ID W_ID;
    ID D_ID;
    ID C_ID;
    TEXT C_LAST[C_LAST_LEN+1];
    TEXT C_CREDIT[2+1];
    REAL C_DISCOUNT;
    COUNT O_OL_CNT;
    ID O_ID;
    TEXT O_ENTRY_D[20];  /* dates as text fields */
    REAL W_TAX;
    REAL D_TAX;
    neworder_item item[15];
    ACID_STUFF;
    } neworder_trans;



typedef struct {
    int status;
    LOGICAL byname;
    ID W_ID;
    ID D_ID;
    ID C_ID;
    ID C_D_ID;
    ID C_W_ID;
    MONEY H_AMOUNT;
    TEXT H_DATE[20];  /* date as text field */
    TEXT W_STREET_1[20+1];
    TEXT W_STREET_2[20+1];
    TEXT W_CITY[20+1];
    TEXT W_STATE[2+1];
    TEXT W_ZIP[9+1];
    TEXT D_STREET_1[20+1];
    TEXT D_STREET_2[20+1];
    TEXT D_CITY[20+1];
    TEXT D_STATE[2+1];
    TEXT D_ZIP[9+1];
    TEXT C_FIRST[16+1];
    TEXT C_MIDDLE[2+1];
    TEXT C_LAST[16+1];
    TEXT C_STREET_1[20+1];
    TEXT C_STREET_2[20+1];
    TEXT C_CITY[20+1];
    TEXT C_STATE[2+1];
    TEXT C_ZIP[9+1];
    TEXT C_PHONE[16+1];
    TEXT C_SINCE[20]; /* date as text field */
    TEXT C_CREDIT[2+1];
    MONEY C_CREDIT_LIM;
    REAL C_DISCOUNT;
    REAL C_BALANCE;
    TEXT  C_DATA[200+1];
    ACID_STUFF;
    } payment_trans;


typedef struct {
    int status;
    LOGICAL byname;
    ID W_ID;
    ID D_ID;
    ID C_ID;
    TEXT C_FIRST[16+1];
    TEXT C_MIDDLE[2+1];
    TEXT C_LAST[16+1];
    MONEY C_BALANCE;
    ID O_ID;
    TEXT O_ENTRY_DATE[20];   /* date as text field */
    ID O_CARRIER_ID;
    COUNT ol_cnt;
    struct {
        ID OL_SUPPLY_W_ID;
        ID OL_I_ID;
        COUNT OL_QUANTITY;
        MONEY OL_AMOUNT;
        TEXT OL_DELIVERY_DATE[20]; /* date as text field */
        } item[15];
    ACID_STUFF;
    } ordstat_trans;


typedef struct {
    int status;
    ID W_ID;
    ID D_ID;
    COUNT threshold;
    COUNT low_stock;
    ACID_STUFF;
    } stocklev_trans;

typedef struct {
```

```
    int status;
    ID W_ID;
    ID O_CARRIER_ID;
    struct {
        ID O_ID;
                int status;
        } order[10];
    struct timeval enque[1];
    struct timeval deque[1];
    struct timeval complete[1];
    ACID_STUFF;
    } delivery_trans;


typedef union {
    neworder_trans neworder;
    payment_trans payment;
    ordstat_trans ordstat;
    delivery_trans delivery;
    stocklev_trans stocklev;
    int status;
    } generic_trans;


/*************************************************************
Record formats for results
*************************************************************/

#ifdef NOTYET
typedef struct
    {
    float t1, t2, t3, t4, t5;
    int status          :8;
    unsigned int type    :3;
    unsigned int ol_cnt  :4;
    unsigned int remote_ol_cnt :4;
    unsigned int byname :1;
    unsigned int remote  :1;
    unsigned int skipped  :4;
    int clnt_no;                        /* @022801 ML */
    int userid;                         /* @022801 ML */
    } success_t;
#endif

typedef struct
    {
    TIME t1, t2, t3, t4, t5;
    int status;
    unsigned int type    :3;
    unsigned int ol_cnt  :4;
    unsigned int remote_ol_cnt :4;
    unsigned int byname :1;
    unsigned int remote  :1;
    unsigned int skipped  :4;
    int clnt_no;                        /* @022801 ML */
    int userid;                         /* @022801 ML */
    } success_t;

typedef struct
    {
    struct timeval start_time;
    } success_header_t;


/*************************************************************************
Record formats for loading routines.  (DB's have own internal formats
*************************************************************************/
typedef struct
    {
    ID W_ID;
    TEXT W_NAME[10+1];
    TEXT W_STREET_1[20+1];
    TEXT W_STREET_2[20+1];
    TEXT W_CITY[20+1];
    TEXT W_STATE[2+1];
    TEXT W_ZIP[9+1];
    REAL W_TAX;
    MONEY W_YTD;
    } warehouse_row;

typedef struct
    {
    ID D_ID;
    ID D_W_ID;
    TEXT D_NAME[10+1];
    TEXT D_STREET_1[20+1];
    TEXT D_STREET_2[20+1];
    TEXT D_CITY[20+1];
    TEXT D_STATE[2+1];
    TEXT D_ZIP[9+1];
    REAL D_TAX;
    MONEY D_YTD;
    ID D_NEXT_O_ID;
    } district_row;


typedef struct
    {
    ID C_ID;
    ID C_D_ID;
    ID C_W_ID;
    TEXT C_FIRST[16+1];
    TEXT C_MIDDLE[2+1];
    TEXT C_LAST[16+1];
    TEXT C_STREET_1[20+1];
    TEXT C_STREET_2[20+1];
    TEXT C_CITY[20+1];
    TEXT C_STATE[2+1];
    TEXT C_ZIP[9+1];
    TEXT C_PHONE[16+1];
    DATE C_SINCE;
    TEXT C_CREDIT[2+1];
    MONEY C_CREDIT_LIM;
    REAL C_DISCOUNT;
    MONEY C_BALANCE;
    MONEY C_YTD_PAYMENT;
    COUNT C_PAYMENT_CNT;
    COUNT C_DELIVERY_CNT;
    TEXT  C_DATA[500+1];
    } customer_row;

typedef struct
    {
    ID H_C_ID;
    ID H_C_D_ID;
    ID H_C_W_ID;
    ID H_D_ID;
    ID H_W_ID;
    DATE H_DATE;
    MONEY H_AMOUNT;
    TEXT H_DATA[24+1];
    } history_row;

typedef struct
    {
    ID NO_O_ID;
    ID NO_D_ID;
    ID NO_W_ID;
    } neworder_row;
```

```
typedef struct
    {
    ID O_ID;
    ID O_D_ID;
    ID O_W_ID;
    ID O_C_ID;
    DATE O_ENTRY_D;
    ID O_CARRIER_ID;
    COUNT O_OL_CNT;
    LOGICAL O_ALL_LOCAL;
    } order_row;

typedef struct
    {
    ID OL_O_ID;
    ID OL_D_ID;
    ID OL_W_ID;
    ID OL_NUMBER;
    ID OL_I_ID;
    ID OL_SUPPLY_W_ID;
    DATE OL_DELIVERY_D;
    COUNT OL_QUANTITY;
    MONEY OL_AMOUNT;
    TEXT OL_DIST_INFO[24+1];
    } orderline_row;

typedef struct
    {
    ID I_ID;
    ID I_IM_ID;
    TEXT I_NAME[24+1];
    MONEY I_PRICE;
    TEXT I_DATA[50+1];
    } item_row;

typedef struct
    {
    ID S_I_ID;
    ID S_W_ID;
    COUNT S_QUANTITY;
    TEXT S_DIST_01[24+1];
    TEXT S_DIST_02[24+1];
    TEXT S_DIST_03[24+1];
    TEXT S_DIST_04[24+1];
    TEXT S_DIST_05[24+1];
    TEXT S_DIST_06[24+1];
    TEXT S_DIST_07[24+1];
    TEXT S_DIST_08[24+1];
    TEXT S_DIST_09[24+1];
    TEXT S_DIST_10[24+1];
    COUNT S_YTD;
    COUNT S_ORDER_CNT;
    COUNT S_REMOTE_CNT;
    TEXT S_DATA[50+1];
    } stock_row;


/* Empty field values */
#define EMPTY_NUM (MAXINT-1)
#define INVALID_NUM (MAXINT)
#define EMPTY_FLT   (MAXDOUBLE)
#define INVALID_FLT (MINDOUBLE)

/* Status conditions */
#define OK 0
#define E 1
#define E_INVALID_ITEM 2
#define E_NOT_ENOUGH_ORDERS 3
#define E_DB_ERROR 4
#define E_INVALID_INPUT 5

/* Error message strings */
static char *e_mesg[]={"Transaction complete.","Error","Invalid item number.",
                "Not enough orders.", "Database ERROR !!!!"};



#define YES 1
#define NO 0



double cvt_flt();
double cvt_money();
TIME getclock();
TIME getlocalclock();

#define TPC_MSG_QUE  150


/*****************************************
Transaction specific stuff
*****************************************/

/* types of transactions */
#define NEWORDER 1
#define PAYMENT  2
#define ORDSTAT  3
#define DELIVERY 4
#define STOCKLEV 5
#define DEFERRED 6   /* deferred portion of delivery */

/* the name of each transaction */
static char *transaction_name[] =
    {"", "New_Order", "Payment", "Order-Status",
                "Delivery", "Stock-Level", "Deferred-Delivery"};

/* size of each transaction record */
static int transaction_size[] = {0,
     sizeof(neworder_trans),
     sizeof(payment_trans),
     sizeof(ordstat_trans),
     sizeof(delivery_trans),
     sizeof(stocklev_trans),
     sizeof(delivery_trans),
     0};

/* valid response time for each transaction */
static TIME valid_response[] = {0, 5, 5, 5, 5, 20};


#endif /* TPCC_INCLUDED */
```

lib/date.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

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

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

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

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

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

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

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


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

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


#define Feb29 (31+29-1)

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

    int quad, year, month, day;
    int hour, minute, sec;

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

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

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

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

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

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

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

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

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

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

lib/errlog.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

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

int userid;

static msg_buf();

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

    msg_buf("error \n", strlen("error \n"));
```

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

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

    /* done */
    va_end(argptr);

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


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

    msg_buf("syserror \n", strlen("syserror \n"));
    /* point to the list of arguments */
    va_start(argptr);

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

    /* done */
    va_end(argptr);

    /* display the system error message */
    message("    System error message: %d %s\n", save_errno, strerror(save_errno));

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



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

    msg_buf("message \n", strlen("message \n"));
    /* point to the list of arguments */
    va_start(argptr);

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

    /* done */
    va_end(argptr);
    }



vmessage(format, argptr)
/*************************************************
*************************************************/
    char *format;
    va_list argptr;
{
  char buf[3*1024];

  /* format a message id */
  sprintf(buf, "Host %-8s User %-6d Pid %-6d ", getenv("HOST_NAME"), userid, getpid());

  /* format the string and print it */
  vsprintf(buf+strlen(buf), format, argptr);
  if (getenv("NO_ERROR_LOG") == NULL)
      msg_buf(buf, strlen(buf));
  if (getenv("NO_STDERR") == NULL)
      write(2, buf, strlen(buf));
}



static msg_buf(buf, size)
  char *buf;
  int size;
  {
  int               fd;
  char  *fname;
  time_t tepoch = time(NULL);
  char timestamp[16];
  int ltimestamp;

  ltimestamp = strftime(timestamp, sizeof(timestamp), "%m/%d %T ", localtime(&tepoch));

  /* get the file name to use */
  fname = getenv("ERROR_LOG");
  if (fname == NULL)
    fname = "/tmp/ERROR_LOG";

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

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

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



console_error(str)
   char *str;
   {
   int fd = open("/dev/tty", O_WRONLY);
   write(fd, str, strlen(str));
   close(fd);
   exit(1);
   }
```

lib/fmt.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#include "tpcc.h"
#include "iobuf.h"
#include <math.h>    /* needed for ceil (VM) */
#include <strings.h>

/* formatting routines.  */

/* Note: Currently use integer routines to format and convert.  Need to
   modify the code for cases when integers don't work.  */

fmt_money(str, m, width)
    char *str;
    MONEY m;
    int width;
    {

    if (m == EMPTY_FLT)
        {
        memset(str, '_', width);
        str[width] = '\0';
        return;
        }

    /* format it as a number with a leading blank */
    *str = ' ';
    fmt_flt(str+1, m/100, width-1, 2);

    /* fill in a leading dollar */
    while (*(str+1) == ' ')
        str++;
    *str = '$';
    }


double cvt_money(str)
    char *str;
    {
    char temp[81], *t, *s;
    double cvt_flt(), f;

    /* skip leading and trailing blanks */
    cvt_text(str, temp);

    /* remove leading $ */
    if (*temp == '$')   t = temp + 1;
    else                t = temp;

    /* start scan at current character */
    s = t;

    /* allow leading minus sign */
    if (*s == '-')
                    s++;

    /* allow leading digits */
    while (isdigit(*s))
                    s++;

    /* allow decimal pt and two decimal digits */
    if (*s == '.')    s++;
    if (isdigit(*s))  s++;
    if (isdigit(*s))  s++;

    /* There should be no more characters */
    if (*s != '\0') return INVALID_FLT;

    /* convert the floating pt number */
    f = cvt_flt(t);
    if (f == EMPTY_FLT)        return EMPTY_FLT;
    else if (f == INVALID_FLT) return INVALID_FLT;
    else                       return rint(f*100);
    }


fmt_num(str, n, width)
    char str[];
    int n;
    int width;
    {
    /* mark the end of the string */
    str[width] = '\0';

    /* if empty number, return the empty field */
    if (n == EMPTY_NUM)
        memset(str, '_', width);

    /* otherwise, convert the integer */
    else
        fmtint(str, n, width, ' ');

    debug("fmt_num:  n=%d  str=%s\n", n, str);
    }



cvt_num(str)
    char str[];
    {
    char text[81];
    cvt_text(str, text);
    if (*text == '\0')
        return EMPTY_NUM;
    else
        return cvtint(text);
    }



fmt_flt(str, x, width, dec)
/*****************************************************************************
fmt_flt converts a floating pt number to a string   "999999.9999"
*****************************************************************************/
    char *str;
    double x;
    int width;
    int dec;
    {
    int negative;
    int integer, fract;
    double absolute;

    static double pow10[] =
    {1., 10., 100., 1000., 10000., 100000., 1000000., 10000000., 100000000.};

    /* mark the end of string */
    str[width] = '\0';

    /* if empty value, make it be an empty field */
    if (x == EMPTY_FLT)
        {
        memset(str, '_', width);
        return;
        }
```

```
    absolute = (x < 0)? -x: x;

    /* separate into integer and fractional parts */
    integer = (int) absolute;
    fract   = (absolute - integer) * pow10[dec] + .5;

    /* let the integer portion contain the sign */
    if (x < 0) integer = -integer;

    /* Format integer and fraction separately */
    fmtint(str, integer, width-dec-1, ' ');
    str[width-dec-1] = '.';
    fmtint(str+width-dec, fract, dec, '0');
    }



double cvt_flt(str)
    char str[];
    {
    char text[81];
    char *t;
    double value;
    int div;
    int fract;
    int negative;
    int i;

    /* normalize the text */
    cvt_text(str, text);
    if (*text == '\0')
        return EMPTY_FLT;

    negative = NO;
    fract = NO;
    value = 0;
    div = 1.0;

    negative = (text[0] == '-');
    if (negative)    t = text+1;
    else             t = text;

    for (;  *t !=  '\0' ; t++)
        {

        if (*t == '.')
                        if (fract)  return INVALID_FLT;
            else        fract = YES;

        else if (isdigit(*t))
            {
            value = value*10 + (int)*t - (int)'0';
            if (fract) div *= 10;
            }

        else
            return INVALID_FLT;
        }

    if (fract)
        value /= div;

    if (negative)
        value = -value;

    return value;
    }
```

```
fmt_text(s, text, width)
    char *s, *text;
    int width;
    {

    /* if an empty string, then all underscores */
    if (*text == '\0')
        for (; width > 0; width--)
            *s++  = '_';

    /* otherwise, blank fill it */
    else
        {

        /* copy the text into the new buffer */
        for ( ; *text != '\0';  width--)
            *s++ = *text++;

        /* fill in the rest with blanks */
        for (;  width > 0;  width--)
            *s++ = ' ';
        }

    /* and finally, terminate the string */
    *s = '\0';
    }



cvt_text(s, text)
    char *s;
    char *text;
    {
    char *lastnb;

    /* skip leading blanks and underscores */
    for (; *s == ' ' || *s == '_'; s++)
        ;

    /* copy the characters, keeping track of last blank or underscore */
    lastnb = text-1;
    for (; *s != '\0'; *text++ = *s++)
        if (*s != ' ' && *s != '_')
            lastnb = text;

    /* truncate the text string to last nonblank character */
    *(lastnb+1) = '\0';
    }



fmtint(field, value, size, fill)
/*********************************************************************
fmtint formats an integer value into a character field to make the integer
right-justified within the character field, padded with leading fill
characters (e.g. leading blanks if a blank is passed in for the fill argument
*********************************************************************************/
    int value;
    char *field;
    int size;
    char fill;
    {
    int negative;
    int dividend;
    int remainder;
    char *p;
```

```
    /* create characters from right to left */
    p = field + size - 1;

    /* make note if this is a negative number */
    negative = value < 0;
    if (negative)
        value = -value;

    /* Case: Null field.  Can't do anything */
    if (p < field)
        ;

    /* Case: value is zero.  Print a leading '0' */
    else if (value == 0)
        *p-- = '0';

    /* Otherwise, convert each digit in turn */
    else do
        {

        dividend = value / 10;
        remainder = value - dividend * 10;
        value = dividend;

        *p-- = (char) (  (int)'0' +  remainder  );

        } while (p >= field && value > 0);

    /* insert a minus sign if appropriate */
    if (negative && p >= field)
        *p-- = '-';

    /* fill in leading characters */
    while (p >= field)
        *p-- = fill;
    }


int cvtint(str)
/*****************************************************************
getint extracts an integer value from the given character field
(ex: turns the string "123" into the integer 123)
*****************************************************************/
    char *str;
    {
    int value;
    char c;
    int negative;
    debug("cvtint: str=%s\n", str);

    negative = (*str == '-');
    if (negative)  str++;

    /* convert the integer */
    for (value = 0; isdigit(*str); str++)
        value = value*10 + (int)(*str) - (int)'0';

    /* if any non-digit characters, error */
    if (*str != '\0')
                    return INVALID_NUM;

    /* make negative if there was a minus sign */
    if (negative)
        value = -value;

    debug("cvtint: value=%d\n", value);
    return value;
    }
```

```
fmt_phone(str, phone)
    char str[20];
    char *phone;

    {
    /* copy phone number and insert dashes  999999-999-999-9999 */
    str[0] = phone[0];  str[1] = phone[1]; str[2] = phone[2];
    str[3] = phone[3];  str[4] = phone[4]; str[5] = phone[5];
    str[6] = '-';
    str[7] = phone[6];  str[8] = phone[7]; str[9] = phone[8];
    str[10] = '-';
    str[11] = phone[9]; str[12] = phone[10]; str[13] = phone[11];
    str[14] = '-';
    str[15] = phone[12]; str[16] = phone[13]; str[17] = phone[14];
    str[18] = phone[15];
    str[19] = '\0';
    }


fmt_zip(str,zip)
    char str[20];
    char *zip;
    {
    /* copy zip code and insert dashes 99999-9999 */
    str[0] = zip[0];  str[1] = zip[1];  str[2] = zip[2];
    str[3] = zip[3];  str[4] = zip[4];
    str[5] = '-';
    str[6] = zip[5]; str[7] = zip[6]; str[8] = zip[7];  str[9] = zip[8];
    str[10] = '\0';
    }
```

lib/iobuf.h

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

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

/*********************************************************************
History
941220 LAN Added definition and initialization of the line_col[] array.
       This was needed for modifications made of client program to do
       block I/O using a WYSE terminal.
*********************************************************************/

/* structure for screen emulation */
typedef struct
    {
    int row;
    int col;
    char buf[25][81];
    } screen_t;


typedef struct {
    char *beg;
    char *end;   /* for output buffers */
    char *max;
    char *cur;   /* for input buffers */
    } iobuf;

/* Macro do define an I/O buffer of x characters, initialized to empty */
#define define_iobuf(name, size)                    \
char name##_data[size];                             \
iobuf name[1] = {{name##_data, name##_data,       \
```

```
            name##_data+size, name##_data}}

#define reset(buf)  if (1) { \
    (buf)->cur = (buf)->end = (buf)->beg;               \
    *(buf)->beg = '\0';                                 \
    } else (void)0

#define flush()  if(1) { \
        display(out_buf);    \
        reset(out_buf);      \
     } else (void)0


/* Standard I/O to and from in_buf and out_buf */
#ifdef DECLARE_IO_BUFFERS
define_iobuf(output_stuff, 4*1024);
define_iobuf(input_stuff, 1024);
iobuf *in_buf = input_stuff;
iobuf *out_buf = output_stuff;
#else
iobuf *in_buf;
iobuf *out_buf;
#endif

#define pushc(c) if (1) { \
    if (out_buf->end >= out_buf->max)   \
        error("out_buf overflow:  beg=0x%x end=%d  max=%d\n",  \
         out_buf->beg, out_buf->end-out_buf->beg,out_buf->max-out_buf->beg); \
    *(out_buf->end++) = (c);  \
    *(out_buf->end) = '\0';  /* debug */ \
    } else (void)0

#define popc()  \
    (*in_buf->cur++)


/* Standard characters used for screen control */
#define ENTER '\015'
#define TAB   '\t'
#define BACKTAB '\02' /*  ^B  */
#define CNTRLC '\03'
#define BACKSPACE '\010'
#define BELL '\07'
#define BLANK ' '
#define UNDERLINE '_'
#define ESCAPE '\033'
/*#define EOF    ((char)-1) */
#define TRIGGER '\021' /* dc1 */
```

lib/iobuf.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

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


string(str)
    char str[];
    {
    for (; *str != '\0'; str++)
        pushc(*str);
    }


push(str, len)
    char *str;
    int len;
    {
    for (; len > 0; len --)
        pushc(*str++);
    }


display(scr)
    iobuf *scr;
    {
/* Note: if problems doing output, let the input routine detect it */
    char *p;
    int len;
    for (p = scr->beg; p < scr->end; p+=len)
        {
        len = write(1, p, scr->end - p);
        if (len <= 0) break;
        }
    }


input(scr)
    iobuf *scr;
    {
    int len;

    /* read in as many characters as are available */
    len = read(0, scr->end, scr->max - scr->end);

    /* if end of input, then pretend we read an END character */
    if (len == 0 || (len == -1 && errno == ECONNRESET))
        {
        *scr->end = EOF;
        len = 1;
        }

    /* Check for errors */
    else if (len == -1)
        syserror("input(scr): unable to read stdin\n");

    /* update the pointers to reflect the new data */
    scr->end += len;
    *scr->end='\0';  /* for debugging */
    }


getkey()
    {
    if (in_buf->cur == in_buf->end)
        {
        flush();
        reset(in_buf);
        input(in_buf);
        }

    return popc();
    }
```

lib/random.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 97/12/15 14:01:59 $

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

double drand48();

char lastNames[1000][16];
char customerData1[10][301];
char customerData2[10][201];
char stockData1[10][27];
char stockData2[10][25];
char historyData1[10][13];
char historyData2[10][13];
char citystreetData1[10][11];
char citystreetData2[10][11];
char firstNameData1[10][9];
char firstNameData2[10][9];
char StockDistrict[10][25];
char phoneData[10][17];

static long RandySeedIter = 7;

void GenerateLastNames()
{
   int      i;
   char *name;
   static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
                       "ESE", "ANTI", "CALLY", "ATION", "EING"};

   for(i = 0; i < 1000; i++) {
      name = lastNames[i];
      strcpy(name, n[(i/100)%10]);
      strcat(name, n[(i/10) %10]);
      strcat(name, n[(i/1)  %10]);
   }
}

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

   length = RandomNumber(min, max);

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

   return length;
   }

ID RandomWarehouse(local, scale, percent)
   ID local;
   ID scale;
   int percent;   /* percent of remote transactions */
   {
   ID w_id;

   /* For the given percent of the time, pick the local warehouse */
   if (RandomNumber(1, 100) > percent || scale == 1)
      w_id = local;

   /* Otherwise, pick a non-local warehouse */
   else
      {
      w_id = RandomNumber(2, scale);
      if (w_id == local)
         w_id = 1;
      }
   return w_id;
   }


/* Initialize a table of Random strings for the stock-district
   field in the stock table.  We can use a table of 10 elements
   and select randomly from this table via rule 4.3.2.2 in
   the TPC-C spec */
void InitRandomStrings()
{
   int i;

   for (i=0; i < 10; i++) {
                        MakeAlphaString(24,24,&StockDistrict[i]);

                        MakeAlphaString(300,300,&customerData1[i]);
                        MakeAlphaString(0,200,&customerData2[i]);

                        MakeAlphaString(26,26,&stockData1[i]);
                        MakeAlphaString(0,24,&stockData2[i]);

                        MakeAlphaString(12,12,&historyData1[i]);
                        MakeAlphaString(0,12, &historyData2[i]);

                        MakeAlphaString(10,10,&citystreetData1[i]);
                        MakeAlphaString(0 ,10,&citystreetData2[i]);

                        MakeAlphaString(8,8,&firstNameData1[i]);
                        MakeAlphaString(0,8,&firstNameData2[i]);

                        MakeNumberString(16,16,&phoneData[i]);
                     }
                     GenerateLastNames();
}

int MakeAlphaString(min, max, str)
   int min;
   int max;
   TEXT str[];
   {
   static char character[] = "abcdefghijklmnopqrstuvwxyz";
   int length;
   int i;

   length = RandomNumber(min, max);

   for (i=0;  i<length;  i++) {
      /* NOTE:  we use sizeof(character)-2 because of the following:
         subtract 1 because we are numbering from 0 instead of 1 and
         subtract 1 because the sizeof(character) is 1 greater than
         the data in character because of the inivisible C string
         terminator at the end. */
      str[i] = character[RandomNumber(0, sizeof(character)-2)];
   }
   str[length] = '\0';

   return length;
```

```
    }

void RandomPermutation(perm, n)
    int perm[];
    int n;
    {
    int i, r, t;

    /* generate the identity permutation to start with */
    for (i=1; i<=n; i++)
        perm[i] = i;

    /* randomly shuffle the permutation */
    for (i=1; i<=n; i++)
        {
        r = RandomNumber(i, n);
        t = perm[i]; perm[i] = perm[r]; perm[r] = t;
        }
    }


void RandomDelay(mean, adjust)
/*************************************************************************
random_sleep sleeps according to the TPC specification
*************************************************************************/
    double mean;
    double adjust;
    {
    double secs;
    double exponential();

    secs = exponential(mean);

    delay(secs+adjust);
    }

double exponential(mean)
/*************************************************************************
exponential generates a reverse exponential distribution
*************************************************************************/
    double mean;
    {
    double x;
    double log();

#ifdef USE_DRAND48
    x = -log(1.0-drand48()) * mean;
#else
    x = -log(1.0-randy()) * mean;
#endif

    return x;
    }

void SetRandomSeed(val)
long val;
    {
#ifdef USE_DRAND48
    srand48(val);
#else
    RandySeedIter = val;
    randy();
#endif
    }

void Randomize()
    {
    SetRandomSeed(time(0)+getpid());
    }
```

```
/* Random number generator from Proceeding of the ACM */
#define RANDY_A_VAL 16807
/* 2^31 - 1 */
#define RANDY_M_VAL 2147483647
/* m / a */
#define RANDY_Q_VAL 127773
/* m % a */
#define RANDY_R_VAL 2836

double randy()
{
  long    hi, lo, test;

  hi = RandySeedIter / RANDY_Q_VAL;
  lo = RandySeedIter % RANDY_Q_VAL;

  test = (RANDY_A_VAL * lo) - (RANDY_R_VAL * hi);
  RandySeedIter = (test > 0) ? test : test + RANDY_M_VAL;

  return( (double)RandySeedIter / (double)RANDY_M_VAL );
}  /* end of fn randy  */
```

lib/random.h

```
/*************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

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

#ifdef USE_DRAND48
double drand48();
#else
double randy();
#endif

extern int    MakeNumberString();
extern ID     RandomWarehouse();
extern int    MakeAlphaString();
extern void   RandomPermutation();
extern void   RandomDelay();
extern double exponential();
extern void   Randomize();
extern void   SetRandomSeed();

extern char lastNames[1000][16];
extern char customerData1[10][301];
extern char customerData2[10][201];
extern char stockData1[10][27];
extern char stockData2[10][25];
extern char historyData1[10][13];
extern char historyData2[10][13];
extern char citystreetData1[10][11];
extern char citystreetData2[10][11];
extern char firstNameData1[10][9];
extern char firstNameData2[10][9];
extern char StockDistrict[10][25];
extern char phoneData[10][17];

/*************************************************************************
RandomNumber selects a uniform random number from min to max inclusive
*************************************************************************/
#ifdef USE_DRAND48
#define RandomNumber(min,max)  \
                ((int)(drand48() * ((int)(max) - (int)(min) + 1)) + (int)(min))
```

```
#else
#define RandomNumber(min,max)  \
                ((int)(randy() * ((int)(max) - (int)(min) + 1)) + (int)(min))
#endif

/*************************************************************************
NURandomNumber selects a non-uniform random number
*************************************************************************/
#define NURandomNumber(a, min, max, c) \
   ((RandomNumber(0, a) | RandomNumber(min, max)) + (c)) % \
                                ((max) - (min) + 1) + (min)


#define SelectCityStreetData(data) \
{ \
   strcpy(data,citystreetData1[RandomNumber(0,9)]); \
   strcat(data,citystreetData2[RandomNumber(0,9)]); \
}

#define SelectFirstName(data) \
{ \
   strcpy(data,firstNameData1[RandomNumber(0,9)]); \
   strcat(data,firstNameData2[RandomNumber(0,9)]); \
}

#define SelectHistoryData(data) \
{ \
   strcpy(data,historyData1[RandomNumber(0,9)]); \
   strcat(data,historyData2[RandomNumber(0,9)]); \
}

#define SelectStockData(data) \
{ \
   strcpy(data,stockData1[RandomNumber(0,9)]); \
   strcat(data,stockData2[RandomNumber(0,9)]); \
}

#define SelectClientData(data) \
{ \
   strcpy(data,customerData1[RandomNumber(0,9)]); \
   strcat(data,customerData2[RandomNumber(0,9)]); \
}

#define SelectPhoneData(data) strcpy(data,phoneData[RandomNumber(0,9)])
#define SelectStockDistrict(data) strcpy(data,StockDistrict[RandomNumber(0,9)])

#define MakeZip(zip) \
{ \
  MakeNumberString(4, 4, zip); \
  zip[4] = '1'; \
  zip[5] = '1'; \
  zip[6] = '1'; \
  zip[7] = '1'; \
  zip[8] = '1'; \
  zip[9] = '\0'; \
}

#define MakeAddress(str1, str2, city, state, zip) \
{ \
   SelectCityStreetData(str1); \
   SelectCityStreetData(str2); \
   SelectCityStreetData(city); \
   MakeAlphaString(2,2,state); \
   MakeZip(zip); \
}

#define LastName(num, name) strcpy(name, lastNames[(num)])

#define Original(str) \
{ \
   int len = strlen(str); \
   if (len >= 8) { \
      int 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'; \
   } \
}


#endif
```

lib/results_file.c

```
/*************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $

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



static FILE *rfile;

results_open(id)
   int id;
   {
   char fullname[128];
   char *basename;

   /* get the base file name for the deferred results */
   /*
    * Make it a directory under /tmp so at least we can set it to a
    * symbolic link in case /tmp doesn't have enough room.
    */
   basename = getenv("TPCC_RESULTS_FILE");
   if (basename == NULL)
      basename = "/tmp/TPCC_RESULTS_FILE";

   /* create the full file name */
   sprintf(fullname, "%s.%d", basename, id);

   /* open the file */
   unlink(fullname);
   rfile = fopen(fullname, "wb");
   if (rfile == NULL)
      syserror("Delivery server %d can't open file %s\n", id, fullname);

   /* allocate a larger buffer */
   }



results(t)
   delivery_trans *t;
   {
```

```
    if (fwrite(t, sizeof(*t), 1, rfile) != 1)
        syserror("Delivery server: Can't post results\n");
    }


results_close()
    {
    if (fclose(rfile) < 0)
        syserror("Delivery server can't close file\n");
    }
```

lib/Makefile

```
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/24 17:21:52 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

include ../buildenv.mk

CFLAGS= $(BUILDFLAGS) -Wl,-a,archive_shared

utils=iobuf.o delay.o errlog.o fmt.o random.o tas.o null_key.o null_select.o results_file.o  date.o prepare_socket.o shm.o spinlock.o

all: tpc_lib.a  server_default.o

tpc_lib.a: ${utils}
                rm -f tpc_lib.a
                ar -r tpc_lib.a ${utils}

clean:
                rm -f *.o
                rm -f *.a

clobber: clean

.s.o:
                cc -c $*.s
```

A.3 Transaction Source

client/sybase/transaction.c

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/27 18:49:34 $

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

#include <sybfront.h>
#include <sybdb.h>
#include "tpcc.h"

#define MaxTries 10

int user;
LOGINREC        *login;
DBPROCESS *dbproc;

#define from_sybase_date(sybdate, date) { \
    date.day = sybdate.dtdays; \
    date.sec = sybdate.dttime / 300; \
}

transaction_begin(u)
    int u;
    {
    char *packet;
    int message_handler(), error_handler();

    user = u;

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

    /* install a message handler */
    (void)dbmsghandle(message_handler);
    (void)dberrhandle(error_handler);

    /* set up Sybase structures */
    login = dblogin();
    DBSETLUSER(login, "sa");
    DBSETLPACKET(login,4096);

    /* Open the connection to the server. */
    if ((dbproc = dbopen(login, (char *)NULL )) == NULL)
        error("Could not open connection\n");

    /* Use the TPCC database */
    dbuse(dbproc, "tpcc");
    }


transaction_done()
    {
    /* put detach from database here */
    dbexit();
    }



#define INT2(p) (BYTE *)(((short *)(p)+1))
#define INT1(p) (BYTE *)(((char *)(p)+3))


void neworder_transaction(t)
    neworder_trans *t;
    {
    int try;

    /* repeat until we give up trying */
    for (try=0; try<MaxTries; try++)
        {

        /* if the transactation succeeds, then done */
        if (neworder_body(t))
            break;

        /* clean up and try again */
        dbcancel(dbproc);
        sleep_before_retry();

        /* don't retry if caused by operator error */
        if (t->status == E_INVALID_INPUT) break;
        }

    /* if we finally gave up, then display a message */
    if (try >= MaxTries)
        t->status = E_DB_ERROR;
```

```
    }



int neworder_body (t)
    neworder_trans *t;
    {
    int i;
    DBDATETIME o_entry_d;
    DATE       o_entry_d_DATE;
    REAL tax_n_discount;

    debug("Neworder: w_id=%d d_id=%d c_id=%d\n", t->W_ID, t->D_ID, t->C_ID);

    /* assume everthing fine unless otherwise */
    t->status = OK;

    /* see if our items are all local */
    for (i=0;  i<t->O_OL_CNT; i++)
        if (t->item[i].OL_SUPPLY_W_ID != t->W_ID) break;
    t->all_local = (i >= t->O_OL_CNT);

    /* prepare the parameters for the "neworder" transaction. */
    if (t->all_local)    dbrpcinit(dbproc, "neworder_local", 0);
    else                 dbrpcinit(dbproc, "neworder_remote", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, INT2(&t->W_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, INT1(&t->D_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, (BYTE *)(&t->C_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, (BYTE *)(&t->O_OL_CNT));

    /* Send the orderlines (up to 15) */
    for(i = 0; i < t->O_OL_CNT; i++)
        {
        debug("    i=%d  i_id=%d  w_id=%d  qty=%d\n", i, t->item[i].OL_I_ID,
            t->item[i].OL_SUPPLY_W_ID, t->item[i].OL_QUANTITY);
        dbrpcparam(dbproc,NULL,0,SYBINT4, -1, -1,(BYTE *)(&t->item[i].OL_I_ID));
        if (!t->all_local)
            dbrpcparam(dbproc,NULL,0,SYBINT2, -1, -1,
                INT2(&t->item[i].OL_SUPPLY_W_ID));
        dbrpcparam(dbproc,NULL,0,SYBINT1,-1,-1,INT1(&t->item[i].OL_QUANTITY));
        }

    /* execute the neworder transaction */
    if (dbrpcsend(dbproc) != SUCCEED) return NO;
    if (dbsqlok(dbproc)   != SUCCEED) return NO;

    /* get results from order lines */
    for (i = 0; i < t->O_OL_CNT; i++)
        if (!order_line_result(dbproc, &t->item[i], &t->status))
            break;

    /* get the results of the overall neworder transaction */
    if (dbresults(dbproc) != SUCCEED) return NO;
    dbbind(dbproc, 1, FLT8BIND,        0, (BYTE *)(&t->W_TAX));
    dbbind(dbproc, 2, FLT8BIND,        0, (BYTE *)(&t->D_TAX));
    dbbind(dbproc, 3, INTBIND,         0, (BYTE *)(&t->O_ID));
    dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(t->C_LAST), (BYTE *)t->C_LAST);
    dbbind(dbproc, 5, FLT8BIND,        0,          (BYTE *)(&t->C_DISCOUNT));
    dbbind(dbproc, 6, NTBSTRINGBIND, sizeof(t->C_CREDIT), (BYTE *)t->C_CREDIT);
    dbbind(dbproc, 7, DATETIMEBIND,  0,    (BYTE *)(&o_entry_d));
    if (dbnextrow(dbproc) != REG_ROW) return NO;
    if (dbcanquery(dbproc) != SUCCEED) return NO;

    /* convert the date */
    from_sybase_date(o_entry_d, o_entry_d_DATE);
    fmt_date(&t->O_ENTRY_D,&o_entry_d_DATE);

    /* Check for invalid input (what is -6 anyway?) */
    if (dbretstatus(dbproc) == -6)      {
        t->status = E_INVALID_INPUT;
        return NO;
    }
    /* done */
    return YES;
    }


int order_line_result(dbproc, item, status)
    DBPROCESS *dbproc;
    neworder_item *item;
    int *status;
    {

    /* Each order line is a separate query. Fetch the data */
    if (dbresults(dbproc) != SUCCEED) return NO;
    dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(item->I_NAME),(BYTE *)item->I_NAME);
    dbbind(dbproc, 2, FLT8BIND, 0, (BYTE *)(&item->I_PRICE));
    dbbind(dbproc, 3, INTBIND,  0, (BYTE *)(&item->S_QUANTITY));
    dbbind(dbproc, 4, CHARBIND, 1, (BYTE *)(&item->brand_generic));
    if (dbnextrow(dbproc) != REG_ROW)  return NO;
    if (dbcanquery(dbproc) != SUCCEED) return NO;
    if (dbhasretstat(dbproc) && dbretstatus(dbproc) != 0)  return NO;

    /* Note: items that weren't found will have empty I_NAME */
    if (item->I_NAME[0] == '\0')
        *status = E_INVALID_ITEM;
    return YES;
    }



void payment_transaction(t)
    payment_trans *t;
    {
    int try;

    /* repeat until we give up trying */
    for (try=0; try<MaxTries; try++)
        {

        /* if the transactation succeeds, then done */
        if (payment_body(t))
            break;

        /* clean up and try again */
        dbcancel(dbproc);
        sleep_before_retry();

        /* don't retry if caused by operator error */
        if (t->status == E_INVALID_INPUT) break;
        }

    /* if we finally gave up, then display a message */
    if (try >= MaxTries)
        t->status = E_DB_ERROR;
    }


int payment_body(t)
    payment_trans *t;
    {
    DBDATETIME H_DATE;
    DATE  H_DATE_DATE;
    DBDATETIME C_SINCE;
    DATE C_SINCE_DATE;

    if (t->byname)
        {
```

```
    dbrpcinit(dbproc, "payment_byname", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->C_W_ID));
    dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, (BYTE *)(&t->H_AMOUNT));
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->C_D_ID));
    dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(t->C_LAST), (BYTE *)t->C_LAST);
    }
else
    {
    dbrpcinit(dbproc, "payment_byid", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->C_W_ID));
    dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, (BYTE *)(&t->H_AMOUNT));
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->C_D_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, (BYTE *)(&t->C_ID));
    }


if (dbrpcsend(dbproc) != SUCCEED) return NO;
if (dbsqlok(dbproc) != SUCCEED) return NO;
if (dbresults(dbproc) != SUCCEED) return NO;

dbbind(dbproc,  1, INTBIND,     0, (BYTE *)(&t->C_ID));
dbbind(dbproc,  2, NTBSTRINGBIND, sizeof(t->C_LAST), (BYTE *)t->C_LAST);
dbbind(dbproc,  3, DATETIMEBIND,  0, (BYTE *)(&H_DATE));
dbbind(dbproc,  4, NTBSTRINGBIND, sizeof(t->W_STREET_1), (BYTE *)t->W_STREET_1);
dbbind(dbproc,  5, NTBSTRINGBIND, sizeof(t->W_STREET_2), (BYTE *)t->W_STREET_2);
dbbind(dbproc,  6, NTBSTRINGBIND, sizeof(t->W_CITY), (BYTE *)t->W_CITY);
dbbind(dbproc,  7, NTBSTRINGBIND, sizeof(t->W_STATE), (BYTE *)t->W_STATE);
dbbind(dbproc,  8, NTBSTRINGBIND, sizeof(t->W_ZIP), (BYTE *)t->W_ZIP);

dbbind(dbproc,  9, NTBSTRINGBIND, sizeof(t->D_STREET_1), (BYTE *)t->D_STREET_1);
dbbind(dbproc, 10, NTBSTRINGBIND, sizeof(t->D_STREET_2), (BYTE *)t->D_STREET_2);
dbbind(dbproc, 11, NTBSTRINGBIND, sizeof(t->D_CITY), (BYTE *)t->D_CITY);
dbbind(dbproc, 12, NTBSTRINGBIND, sizeof(t->D_STATE), (BYTE *)t->D_STATE);
dbbind(dbproc, 13, NTBSTRINGBIND, sizeof(t->D_ZIP), (BYTE *)t->D_ZIP);

dbbind(dbproc, 14, NTBSTRINGBIND, sizeof(t->C_FIRST), (BYTE *)t->C_FIRST);
dbbind(dbproc, 15, NTBSTRINGBIND, sizeof(t->C_MIDDLE), (BYTE *)t->C_MIDDLE);
dbbind(dbproc, 16, NTBSTRINGBIND, sizeof(t->C_STREET_1), (BYTE *)t->C_STREET_1);
dbbind(dbproc, 17, NTBSTRINGBIND, sizeof(t->C_STREET_2), (BYTE *)t->C_STREET_2);
dbbind(dbproc, 18, NTBSTRINGBIND, sizeof(t->C_CITY), (BYTE *)t->C_CITY);
dbbind(dbproc, 19, NTBSTRINGBIND, sizeof(t->C_STATE), (BYTE *)t->C_STATE);
dbbind(dbproc, 20, NTBSTRINGBIND, sizeof(t->C_ZIP), (BYTE *)t->C_ZIP);
dbbind(dbproc, 21, NTBSTRINGBIND, sizeof(t->C_PHONE), (BYTE *)t->C_PHONE);
dbbind(dbproc, 22, DATETIMEBIND,  0, (BYTE *)(&C_SINCE));
dbbind(dbproc, 23, NTBSTRINGBIND, sizeof(t->C_CREDIT), (BYTE *)t->C_CREDIT);
dbbind(dbproc, 24, FLT8BIND,      0, (BYTE *)(&t->C_CREDIT_LIM));
dbbind(dbproc, 25, FLT8BIND,      0, (BYTE *)(&t->C_DISCOUNT));
dbbind(dbproc, 26, FLT8BIND,      0, (BYTE *)(&t->C_BALANCE));
dbbind(dbproc, 27, NTBSTRINGBIND, sizeof(t->C_DATA), (BYTE *)t->C_DATA);

if (dbnextrow(dbproc) != REG_ROW) return NO;
if (dbcanquery(dbproc) != SUCCEED) return NO;

t->status = OK;
/* need to be in string format */

from_sybase_date(C_SINCE, C_SINCE_DATE);
from_sybase_date(H_DATE, H_DATE_DATE);
fmt_date(t->H_DATE, &H_DATE_DATE);
fmt_date(t->C_SINCE, &C_SINCE_DATE);

/* Check for invalid input */
if (dbretstatus(dbproc) == -6)      {
    t->status = E_INVALID_INPUT;
    return NO;
}
    return YES;
    }


void ordstat_transaction(t)
    ordstat_trans *t;
    {
    int try;

    /* repeat until we give up trying */
    for (try=0; try<MaxTries; try++)
        {

        /* if the transactation succeeds, then done */
        if (ordstat_body(t))
            break;

        /* clean up and try again */
        dbcancel(dbproc);
        sleep_before_retry();

        /* don't retry if caused by operator error */
        if (t->status == E_INVALID_INPUT) break;
        }

    /* if we finally gave up, then display a message */
    if (try >= MaxTries)
        t->status = E_DB_ERROR;
    }



int ordstat_body(t)
    ordstat_trans *t;
    {
    ID ol_supply_w_id;
    ID ol_i_id;
    COUNT ol_quantity;
    MONEY ol_amount;
    DBDATETIME ol_delivery_d;
    DBDATETIME o_entry_d;
    DATE       ol_delivery_d_DATE;
    DATE       o_entry_d_DATE;

    int i, code;


    /* if this is by name, then invoke the byname procedure */
    if (t->byname)
        {
        dbrpcinit(dbproc, "order_status_byname", 0);
        dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
        dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(t->C_LAST), (BYTE *)t->C_LAST);
        }

    /* otherwise, invoke the by id procedure */
    else
        {
        dbrpcinit(dbproc, "order_status_byid", 0);
        dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
        dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, (BYTE *)(&t->C_ID));
        }

    /* make the rpc call and check for errors */
    if (dbrpcsend(dbproc) != SUCCEED) return NO;
    if (dbsqlok(dbproc) != SUCCEED) return NO;
```

```
    if (dbresults(dbproc) != SUCCEED) return NO;

    /* Code for TPC-C rev. 3.3 error checking.  */
    if (dbrows(dbproc) != SUCCEED )     {
        t->status = E_INVALID_INPUT;
    }
    /* prepare to fetch the results */
    dbbind(dbproc, 1, INTBIND,       0,     (BYTE *)(&ol_supply_w_id));
    dbbind(dbproc, 2, INTBIND,       0,     (BYTE *)(&ol_i_id));
    dbbind(dbproc, 3, INTBIND,       0,     (BYTE *)(&ol_quantity));
    dbbind(dbproc, 4, FLT8BIND,      0,     (BYTE *)(&ol_amount));
    dbbind(dbproc, 5, DATETIMEBIND,  0,     (BYTE *)(&ol_delivery_d));


    /* do for each row */
    for (i=0; (code = dbnextrow(dbproc)) == REG_ROW &&  i<15; i++) {

        /* move the information into the structure  */
        t->item[i].OL_SUPPLY_W_ID = ol_supply_w_id;
        t->item[i].OL_I_ID = ol_i_id;
        t->item[i].OL_QUANTITY = ol_quantity;
        t->item[i].OL_AMOUNT = ol_amount;
        from_sybase_date(ol_delivery_d, ol_delivery_d_DATE);
        if (IsEmptyDate(&ol_delivery_d_DATE)) {
            t->item[i].OL_DELIVERY_DATE[0] = '\0';
        } else {
            fmt_date(t->item[i].OL_DELIVERY_DATE, &ol_delivery_d_DATE);
        }
    }

    if (code != NO_MORE_ROWS) return NO;

    /* remember how many rows we found */
    t->ol_cnt = i;

    if (dbresults(dbproc) != SUCCEED) return NO;

    /* Code for TPC-C rev. 3.3 error checking.  */
    if (dbrows(dbproc) != SUCCEED )     {
        t->status = E_INVALID_INPUT;
    }

    /* fetch the remaining information */
    dbbind(dbproc, 1, INTBIND,     0,     (BYTE *)(&t->C_ID));
    dbbind(dbproc, 2, NTBSTRINGBIND, sizeof(t->C_LAST), (BYTE *)t->C_LAST);
    dbbind(dbproc, 3, NTBSTRINGBIND, sizeof(t->C_FIRST), (BYTE *)t->C_FIRST);
    dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(t->C_MIDDLE), (BYTE *)t->C_MIDDLE);
    dbbind(dbproc, 5, FLT8BIND,      0,     (BYTE *)(&t->C_BALANCE));
    dbbind(dbproc, 6, INTBIND,       0,     (BYTE *)(&t->O_ID));
    dbbind(dbproc, 7, DATETIMEBIND,  0, (BYTE *)(&o_entry_d));
    dbbind(dbproc, 8, INTBIND,     0,     (BYTE *)(&t->O_CARRIER_ID));
    if (dbnextrow(dbproc) != REG_ROW) return NO;
    if (dbcanquery(dbproc) != SUCCEED) return NO;

    /* convert the date */
    from_sybase_date(o_entry_d, o_entry_d_DATE);
    fmt_date(t->O_ENTRY_DATE, &o_entry_d_DATE);

    t->status = OK;
    return YES;
    }



delivery_transaction(t)
    delivery_trans *t;
    {
    ID d;

    int try;

    d = 1;

    /* repeat until we give up trying */
    for (try=0; try<MaxTries; try++)
        {

        /* if the transactation succeeds, then done */
        d = delivery_body(t, d);
        if (d > 10) break;

        /* clean up and try again */
        dbcancel(dbproc);
        sleep_before_retry();

        /* don't retry if caused by operator error */
        if (t->status == E_INVALID_INPUT) break;
        }

    /* any uncompleted districts have an error */
    for (  ; d <= 10;  d++)
        t->order[d-1].status = E_DB_ERROR;
    }


int delivery_body(t, d)
    delivery_trans *t;
    ID d;
    {

    dbrpcinit(dbproc, "delivery", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->O_CARRIER_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&d));
    if (dbrpcsend(dbproc) != SUCCEED) return d;
    if (dbsqlok(dbproc) != SUCCEED) return d;

    for ( ; d <= 10; d++)
        {

        /* Each order line is a separate query. Fetch the data */
        if (dbresults(dbproc) != SUCCEED) break;
        dbbind(dbproc, 1, INTBIND,  0, (BYTE *)(&t->order[d-1].O_ID));
        if (dbnextrow(dbproc) != REG_ROW)  break;
        if (dbcanquery(dbproc) != SUCCEED) break;
        if (dbhasretstat(dbproc) && dbretstatus(dbproc) != 0)  break;

        if (t->order[d-1].O_ID == 0) t->order[d-1].status=E_NOT_ENOUGH_ORDERS;
        else                         t->order[d-1].status = OK;
        }

    return d;
    }



stocklev_transaction(t)
    stocklev_trans *t;
    {
    int try;

    /* repeat until we give up trying */
    for (try=0; try<MaxTries; try++)
        {

        /* if the transactation succeeds, then done */
        if (stocklev_body(t))
            break;
```

```
        /* clean up and try again */
        dbcancel(dbproc);
        sleep_before_retry();

        /* don't retry if caused by operator error */
        if (t->status == E_INVALID_INPUT) break;
        }

    /* if we finally gave up, then display a message */
    if (try >= MaxTries)
        t->status = E_DB_ERROR;

    }


int stocklev_body (t)
    stocklev_trans *t;
    {
    int iid, uniq[500];
    int i, j, count;
    int duplicate_found;

    dbrpcinit(dbproc, "stock_level", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->W_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, INT1(&t->D_ID));
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, INT2(&t->threshold));
    if (dbrpcsend(dbproc) != SUCCEED) return NO;
    if (dbsqlok(dbproc) != SUCCEED) return NO;

    if (dbresults(dbproc) != SUCCEED) return NO;
    dbbind(dbproc, 1, INTBIND,  0, (BYTE *)(&iid));
    count = 0;
    while (dbnextrow(dbproc) == REG_ROW) {
        duplicate_found = 0;
        for (j = 0; j < count; j++) {
            if (iid == uniq[j]) {
              duplicate_found = 1;
              break;
            }
        }
        /* if this was a dupicate of something already found, then
           don't count it and continue */
        if (duplicate_found) continue;

        if (count < 500) {
            uniq[count++] = iid;
        } else {
            return NO;
        }
    }
    if (dbcanquery(dbproc) != SUCCEED) return NO;

    t->status    = OK;
    t->low_stock = count;
    return YES;
    }


int sleep_before_retry()
    {
    delay(.1);
    }


to_sybase_date(date, sybdate)
    DATE *date;
    DBDATETIME *sybdate;
    {
    sybdate->dtdays = date->day;
    sybdate->dttime = date->sec*300;
    }

int deadlock = NO;

/*****************************************************************************
 * error_handler deals with error messages
 *****************************************************************************/
int
error_handler(DBPROCESS *dbproc,
                int severity,
        int errno,
                int oserr)
{
    if (deadlock)
        message("Error: Deadlock detected: Error #%d, severity = %d, oserr = %d\n", errno, severity, oserr);
    else
        message("Error #%d, severity = %d, oserr = %d\n",
                errno,    severity, oserr);
    deadlock = NO;
    return INT_CANCEL;
}



/*****************************************************************************
 * message_handler deals with informational messges
 *****************************************************************************/
message_handler(DBPROCESS *dbproc,
                int msgno,
                        int msgstate,
                        int severity,
                        char *msgtext,
                        char *srvname,
            char *procname,
                int  line)
{
    /* Ignore messages that will be passed to error handler anyway */
    if (msgno == SYBESMSG) {
        message("Message: Error detected from %s line %d, msgstate = %d, severity = %d, msgtext = %s\n", procname, line, msgstate,
severity, msgtext);
        return(SUCCEED);
    }

    /* Force an error for Deadlocks */
    else if (msgno == 1205)
        {
        deadlock = YES;
        message("Message: Deadlock detected from %s line %d\n", procname, line);
        return(FAIL);
        }

    else if (msgno != 5701 && msgno != 5703 && msgno != 5704)
        {
        message("Message #%d from %s line %d\n%s\n",
                msgno,    procname, line, msgtext);
        return(FAIL);
        }
}
```

client/service.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 97/12/15 10:53:26 $
```

```
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/

#include <unistd.h>
#include <sys/types.h>
#include "tpcc.h"
#include "atmi.h"

extern int userid;
char *cmd          = NULL;

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

   /*
    * search for the options
    *    "-n" server number
    *    "-S" server program
    * purpose: to get svr_id & progname for DVRY_LOG files
    *
    */
   while ((c = getopt(argc, argv, "n:S:h:")) != EOF) {
      switch(c) {
      case 'n':
         userid = atoi(optarg);
         break;
      case 'S':
         cmd  = optarg;
         break;
     }
   }

   ret = transaction_begin(userid);
   results_open(userid);

   return 0;
}


void NEWO_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
   neworder_transaction((neworder_trans *)svcinfo->data);
   tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}


void PMT_SVC(svcinfo)
TPSVCINFO *svcinfo;
{

   payment_transaction((payment_trans *)svcinfo->data);
   tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void ORDS_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
   ordstat_transaction((ordstat_trans *)svcinfo->data);
   tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void STKL_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
   stocklev_transaction((stocklev_trans *)svcinfo->data);
   tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

void DVRY_SVC(svcinfo)
TPSVCINFO *svcinfo;
{
   delivery_trans *t = (delivery_trans *)svcinfo->data;
   gettimeofday(t->deque, NULL);
   delivery_transaction(t);
   gettimeofday(t->complete, NULL);
   results(t);

   /* Why do we return things ? */
   tpreturn(TPSUCCESS, 0, svcinfo->data, svcinfo->len, 0);
}

/****************************************************************
tpsrvdone cleans up after the TPC transaction service
*****************************************************************/
void tpsvrdone()
{
   transaction_done();
   results_close();

   /* Log a message saying we are done */
   message("TUXEDO service %s has shutdown\n", cmd);
}
```

A.4 TPC-C Stored Procedures

tpcc_proc.sh

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

# Stored procedure for TPC-C 3.2 on SQL Server 11.1 and later
# Copyright Sybase 1997
```

```
#
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       int,

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

declare
        @w_tax          real,           @d_tax          real,
        @c_last         char(16),       @c_credit       char(2),
        @c_discount     real,           @commit_flag    int,
        @c_ins_id       int,            @local_d_id     int,

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

        @s_quantity     smallint,       @ten_smallint   smallint,
        @s_ytd          int,            @s_order_cnt    int,
        @s_dist         char(24),       @s_data         char(50),
        @one_smallint   smallint,       @zero_smallint  smallint,
        @ninenine_smallint      smallint,

        @ol_number      int,            @o_id           int,
        @o_entry_d      datetime,       @b_g            char(1),
        @ol_amount      float



begin

        begin transaction NO


        -- @#@# UPDATE district FROM district, warehouse, customer
        --
        UPDATE district
        SET             d_next_o_id  = d_next_o_id + 1
                      , @o_id        = d_next_o_id
                      , @d_tax       = d_tax
                      , @commit_flag = 1
                      , @ol_number   = 0
                      , @local_d_id  = @d_id
                      , @ten_smallint = 10
                      , @zero_smallint = 0
                      , @ninenine_smallint = 99
                      , @one_smallint = 1
                      , @o_entry_d  = getdate()
        WHERE           d_w_id      = @w_id
                AND     d_id        = @d_id

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

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

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

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

                if (@@rowcount = 0)
                  begin
                    select @commit_flag = 0
                    select NULL, NULL, NULL, NULL
                    continue
                  end
          /*Otherwise if the item is found */
                  update stock
                    set s_ytd = s_ytd + @ol_qty,
                                  @ol_amount = @ol_qty * @i_price,
                              @s_quantity = s_quantity - @ol_qty +
                          case when (s_quantity - @ol_qty < @ten_smallint)
                          then @ninenine_smallint else @zero_smallint end,
                        s_quantity = s_quantity - @ol_qty +
                          case when (s_quantity - @ol_qty < @ten_smallint)
```

```
                then @ninenine_smallint  else @zero_smallint end,
                    s_order_cnt = s_order_cnt + @one_smallint,
            @s_data = s_data,
            @s_dist = case @local_d_id
                when 1 then s_dist_01
                when 2 then s_dist_02
                when 3 then s_dist_03
                when 4 then s_dist_04
                when 5 then s_dist_05
                when 6 then s_dist_06
                when 7 then s_dist_07
                when 8 then s_dist_08
                when 9 then s_dist_09
                when 10 then s_dist_10
                end
                    where s_w_id = @w_id and
                        s_i_id = @i_id
        if (@@rowcount = 0)
            begin
              select @commit_flag = 0
              select NULL, NULL, NULL, NULL
              continue
                    end
         /*Otherwise if the Stock is found */

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


/* send line-item data to client */
select
   @i_name,
   @i_price,
   @s_quantity,
   b_g= case when((patindex("%ORIGINAL%", @i_data) > 0) and
            (patindex("%ORIGINAL%", @s_data) > 0))
        then "B" else "G" end

      end /* while */

      SELECT      @c_last       = c_last,
                  @c_discount   = c_discount,
                  @c_credit     = c_credit,
                  @c_ins_id     = c_id
      FROM        customer (index c_clu prefetch 4 lru)  HOLDLOCK
      WHERE       c_w_id        = @w_id
        AND       c_d_id        = @d_id
        AND       c_id          = @c_id

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


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

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

      select            /* Return to client */
            @w_tax, @d_tax, @o_id, @c_last,
            @c_discount, @c_credit, @o_entry_d
end
go

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

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

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

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

      @s_quantity       smallint,   @ten_smallint smallint,
      @s_ytd            int,                    @s_order_cnt int,
      @s_dist           char(24),         @s_data           char(50),
      @one_smallint     smallint,         @zero_smallint    smallint,
      @ninenine_smallint      smallint,

      @ol_number        int,              @o_id             int,
      @o_entry_d        datetime,   @b_g                    char(1),
      @ol_amount        float


begin

      begin transaction NO

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

      UPDATE district
      SET         d_next_o_id  = d_next_o_id + 1
                  , @o_id                 = d_next_o_id
                  , @d_tax          = d_tax
```

```
                , @commit_flag          = 1
                , @ol_number = 0
                , @local_d_id  = @d_id
                , @ten_smallint         = 10
                , @zero_smallint = 0
                , @ninenine_smallint    = 99
                , @one_smallint         = 1
                , @o_entry_d  = getdate()
WHERE           d_w_id          = @w_id
                AND             d_id            = @d_id

while (@ol_number < @o_ol_cnt) begin
                SELECT @ol_number = @ol_number + 1
                ,@i_id = case @ol_number
 when 1 then @i_id2
        when 2 then @i_id3
        when 3 then @i_id4
        when 4 then @i_id5
        when 5 then @i_id6
        when 6 then @i_id7
        when 7 then @i_id8
        when 8 then @i_id9
        when 9 then @i_id10
        when 10 then @i_id11
        when 11 then @i_id12
        when 12 then @i_id13
        when 13 then @i_id14
        when 14 then @i_id15
        else @i_id
        end
                , @ol_qty = case @ol_number
        when 1 then @ol_qty2
        when 2 then @ol_qty3
        when 3 then @ol_qty4
        when 4 then @ol_qty5
        when 5 then @ol_qty6
        when 6 then @ol_qty7
        when 7 then @ol_qty8
        when 8 then @ol_qty9
        when 9 then @ol_qty10
        when 10 then @ol_qty11
        when 11 then @ol_qty12
        when 12 then @ol_qty13
        when 13 then @ol_qty14
        when 14 then @ol_qty15
        else @ol_qty
        end
                ,@s_w_id = case @ol_number
 when 1 then @s_w_id2
        when 2 then @s_w_id3
        when 3 then @s_w_id4
        when 4 then @s_w_id5
        when 5 then @s_w_id6
        when 6 then @s_w_id7
        when 7 then @s_w_id8
        when 8 then @s_w_id9
        when 9 then @s_w_id10
        when 10 then @s_w_id11
        when 11 then @s_w_id12
        when 12 then @s_w_id13
        when 13 then @s_w_id14
        when 14 then @s_w_id15
        else @s_w_id
        end

 /* convert c_no_is cursor to a simple select */
 /* get item data (no one update item) */
                select @i_price = i_price,
                        @i_name = i_name ,
                        @i_data = i_data
                        from item HOLDLOCK
                        where i_id = @i_id

                if (@@rowcount = 0)
                  begin
                        select @commit_flag = 0
                        select NULL, NULL, NULL, NULL
                                continue
                  end
        /* Otherwise if the item is found */
          update stock
            set s_ytd = s_ytd + @ol_qty,
                        @ol_amount = @ol_qty * @i_price,
                        @s_quantity = s_quantity - @ol_qty +
                       case when (s_quantity - @ol_qty < @ten_smallint)
                    then @ninenine_smallint else @zero_smallint end,
                s_quantity = s_quantity - @ol_qty +
                    case when (s_quantity - @ol_qty < @ten_smallint)
                    then @ninenine_smallint  else @zero_smallint end,
                            s_order_cnt = s_order_cnt + @one_smallint,
                @s_data = s_data,
                @s_dist = case @local_d_id
                        when 1 then s_dist_01
                        when 2 then s_dist_02
                        when 3 then s_dist_03
                        when 4 then s_dist_04
                        when 5 then s_dist_05
                        when 6 then s_dist_06
                        when 7 then s_dist_07
                        when 8 then s_dist_08
                        when 9 then s_dist_09
                        when 10 then s_dist_10
                        end,
                            s_remote_cnt = s_remote_cnt +
                            case when (@s_w_id = @w_id)
                then 0 else 1 end
where s_w_id = @s_w_id and
      s_i_id = @i_id

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

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

             /* send line-item to client */
             select
                        @i_name,
 @i_price,
 @s_quantity,
 b_g = case when ((patindex("%ORIGINAL%", @i_data) > 0) and
                 (patindex("%ORIGINAL%", @s_data) > 0))
        then "B" else "G" end
         end /* while */


        SELECT          @c_last         = c_last,
                        @c_discount  = c_discount,
                        @c_credit       = c_credit,
                        @c_ins_id       = c_id
```

```
        FROM        customer (index c_clu prefetch 4 lru)  HOLDLOCK
        WHERE       c_w_id          = @w_id
          AND       c_d_id          = @d_id
          AND       c_id            = @c_id

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

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

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

        select                  /* Return to client */
                @w_tax, @d_tax, @o_id, @c_last,
                @c_discount, @c_credit, @o_entry_d
end
go
if exists (select * from sysobjects where name = 'payment_byid')
        DROP PROC payment_byid
go
CREATE PROC payment_byid
        @w_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,                  @w_id_retrieved smallint

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,                  @commit_flg     int

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(12,0),  @c_balance      float,
        @c_discount     real,
        @data1                          char(250),      @data2                  char(250),
        @c_data_1       char(250),      @c_data_2       char(250)

declare @screen_data    char(200),      @today          datetime


BEGIN TRANSACTION PID

        UPDATE district
                SET d_ytd = d_ytd + @h_amount
                  ,@d_ytd = d_ytd
                  ,@d_street_1 = d_street_1
                  ,@d_street_2 = d_street_2
                  ,@d_city = d_city
                  ,@d_state = d_state
                  ,@d_zip = d_zip
                  ,@d_name = d_name
                  ,@commit_flg = 1
                WHERE d_w_id            = @w_id
                AND     d_id            = @d_id

UPDATE warehouse
        SET w_ytd = w_ytd + @h_amount
          ,@w_ytd = w_ytd
          ,@w_id_retrieved = w_id
          ,@w_street_1 = w_street_1
          ,@w_street_2 = w_street_2
          ,@w_city = w_city
          ,@w_state = w_state
          ,@w_zip = w_zip
          ,@w_name = w_name
        WHERE w_id = @w_id

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

/* Customer data */
UPDATE customer SET
          @c_first = c_first
        , @c_middle = c_middle
        , @c_last = c_last
        , @c_street_1 = c_street_1
        , @c_street_2 = c_street_2
        , @c_city = c_city
        , @c_state = c_state
        , @c_zip = c_zip
        , @c_phone = c_phone
        , @c_credit = c_credit
        , @c_credit_lim = c_credit_lim
        , @c_discount = c_discount
        , c_balance = c_balance - @h_amount
        , @c_balance = c_balance - @h_amount
        , c_ytd_payment = c_ytd_payment + @h_amount
        , c_payment_cnt = c_payment_cnt + 1
        , @c_since = c_since
        , @data1 = c_data1
        , @data2 = c_data2
        , @today = getdate()
where
        c_id = @c_id
        and c_w_id = @c_w_id
        and c_d_id = @c_d_id

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

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

        UPDATE customer SET
```

```
                                        c_data1 = @c_data_1
                                      , c_data2 = @c_data_2
                                      , @screen_data = substring(@c_data_1, 1, 200)
                        WHERE
                                        c_id = @c_id
                                        AND c_w_id = @c_w_id
                                        AND c_d_id = @c_d_id
        end /* if */

        /* Create the history record */
        INSERT INTO history (
                        h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                        h_date, h_amount, h_data)
        VALUES (
                        @c_id, @c_d_id, @c_w_id, @d_id, @w_id_retrieved,
                        @today, @h_amount, (@w_name + "    " + @d_name))

/* COMMIT TRANSACTION PID */

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

        select                  /* Return to client */
                        @c_id,
                        @c_last,
                        @today,
                        @w_street_1,
                        @w_street_2,
                        @w_city,
                        @w_state,
                        @w_zip,

                        @d_street_1,
                        @d_street_2,
                        @d_city,
                        @d_state,
                        @d_zip,

                        @c_first,
                        @c_middle,
                        @c_street_1,
                        @c_street_2,
                        @c_city,
                        @c_state,
                        @c_zip,
                        @c_phone,
                        @c_since,
                        @c_credit,
                        @c_credit_lim,
                        @c_discount,
                        @c_balance,
                        @screen_data
go
if exists (select * from sysobjects where name = 'payment_byname')
        DROP PROC payment_byname
go
CREATE PROC payment_byname
        @w_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,                  @w_id_retrieved smallint

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,                  @commit_flg     int

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(12,0),  @c_balance      float,
        @c_discount     real,
        @data1                          char(250),      @data2                  char(250),
        @c_data_1       char(250),      @c_data_2       char(250)

declare @screen_data    char(200),      @today          datetime


BEGIN TRANSACTION PNM
        SELECT @n = (count(*)+1)/2
        FROM customer (index c_non1 prefetch 4 lru) HOLDLOCK
        WHERE   c_w_id = @c_w_id and
                c_d_id = @c_d_id and
                c_last = @c_last

        set rowcount @n

        -- @#@ SELECT FROM customer HOLDLOCK
        SELECT @c_id = c_id
        FROM customer (index c_non1 prefetch 4 lru) HOLDLOCK
        WHERE   c_w_id = @c_w_id and
                c_d_id = @c_d_id and
                c_last = @c_last

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

        UPDATE district
                SET d_ytd = d_ytd + @h_amount
                  ,@d_ytd = d_ytd
                  ,@d_street_1 = d_street_1
                  ,@d_street_2 = d_street_2
                  ,@d_city = d_city
                  ,@d_state = d_state
                  ,@d_zip = d_zip
                  ,@d_name = d_name
                  ,@commit_flg = 1
                WHERE d_w_id            = @w_id
                AND     d_id            = @d_id

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

        UPDATE warehouse
                SET w_ytd = w_ytd + @h_amount
                  ,@w_ytd = w_ytd
                  ,@w_id_retrieved = w_id
                  ,@w_street_1 = w_street_1
                  ,@w_street_2 = w_street_2
                  ,@w_city = w_city
                  ,@w_state = w_state
                  ,@w_zip = w_zip
                  ,@w_name = w_name
                WHERE w_id = @w_id

        /* Customer data */
```

```
        UPDATE customer SET
                        @c_first = c_first
                      , @c_middle = c_middle
                      , @c_last = c_last
                      , @c_street_1 = c_street_1
                      , @c_street_2 = c_street_2
                      , @c_city = c_city
                      , @c_state = c_state
                      , @c_zip = c_zip
                      , @c_phone = c_phone
                      , @c_credit = c_credit
                      , @c_credit_lim = c_credit_lim
                      , @c_discount = c_discount
                      , c_balance = c_balance - @h_amount
                      , @c_balance = c_balance - @h_amount
                      , c_ytd_payment = c_ytd_payment + @h_amount
                      , c_payment_cnt = c_payment_cnt + 1
                      , @c_since = c_since
                      , @data1 = c_data1
                      , @data2 = c_data2
                      , @today = getdate()
        where
                      c_id = @c_id
                      and c_w_id = @c_w_id
                      and c_d_id = @c_d_id

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

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

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

        /* Create the history record */
        INSERT INTO history (
                      h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                      h_date, h_amount, h_data)
        VALUES (
                      @c_id, @c_d_id, @c_w_id, @d_id, @w_id_retrieved,
                      @today, @h_amount, (@w_name + "    " + @d_name))

/* COMMIT TRANSACTION PNM */

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

              select                  /* Return to client */
                            @c_id,
                            @c_last,
                            @today,
                            @w_street_1,
                            @w_street_2,
                            @w_city,
                            @w_state,
                            @w_zip,

                            @d_street_1,
                            @d_street_2,
                            @d_city,
                            @d_state,
                            @d_zip,

                            @c_first,
                            @c_middle,
                            @c_street_1,
                            @c_street_2,
                            @c_city,
                            @c_state,
                            @c_zip,
                            @c_phone,
                            @c_since,
                            @c_credit,
                            @c_credit_lim,
                            @c_discount,
                            @c_balance,
                            @screen_data
go
if exists (select * from sysobjects where name = 'order_status_byid')
              DROP PROC order_status_byid
go
CREATE PROC order_status_byid
              @w_id                   smallint,
              @d_id                   tinyint,
              @c_id                   int
as

DECLARE       @o_id                   int,
              @o_entry_d    datetime,
              @o_carrier_id smallint

BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
              set rowcount 1
              SELECT        @o_id = o_id, @o_carrier_id = o_carrier_id,
                            @o_entry_d = o_entry_d
              FROM          orders (index o_clu prefetch 16 mru) HOLDLOCK
              WHERE         o_w_id          = @w_id
              AND           o_d_id          = @d_id
              AND           o_c_id          = @c_id
              ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
              set rowcount 0

/* Select 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 4 lru) HOLDLOCK
                WHERE   c_id    = @c_id
                AND     c_d_id  = @d_id
                AND     c_w_id  = @w_id

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

DECLARE @o_id                   int,
        @o_entry_d      datetime,
        @o_carrier_id smallint

declare         @n              int,            @c_id           int

BEGIN TRANSACTION OSNM
        SELECT @n = (count(*)+1)/2
        FROM customer (index c_non1 prefetch 4 lru) HOLDLOCK
        WHERE   c_w_id = @w_id and
                c_d_id = @d_id and
                c_last = @c_last


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

        -- @#@# SELECT FROM customer HOLDLOCK
        SELECT @c_id = c_id
        FROM customer (index c_non1 prefetch 4 lru) HOLDLOCK
        WHERE   c_w_id = @w_id and
                c_d_id = @d_id and
                c_last = @c_last


/* Get the latest order made by the customer */
        set rowcount 1
        SELECT  @o_id = o_id, @o_carrier_id = o_carrier_id,
                @o_entry_d = o_entry_d
        FROM    orders (index o_clu prefetch 16 mru) HOLDLOCK
        WHERE   o_w_id          = @w_id
        AND     o_d_id          = @d_id
        AND     o_c_id          = @c_id
        ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
        set rowcount 0

/* Select 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 4 lru) HOLDLOCK
        WHERE   c_id    = @c_id
        AND     c_d_id  = @d_id
        AND     c_w_id  = @w_id

COMMIT TRANSACTION OSNM
go

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

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

declare @no_o_id    int,        @o_c_id     smallint,
    @ol_total   float,      @ol_amount  float,
    @junk_id    smallint,   @ten_tinyint    tinyint,
            @one_tinyint    tinyint,        @one_smallint           smallint,
    @today      datetime

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

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

    OPEN c_del_no

    FETCH c_del_no INTO @no_o_id

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

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

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

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

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

        COMMIT TRANSACTION DEL

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


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

CREATE PROC stock_level
            @w_id           smallint,
            @d_id           tinyint,
            @threshold smallint
as
            select  s_i_id /* Return to client */
            FROM            district,
                            order_line (index ol_clu prefetch 4 lru),
                            stock (index s_clu prefetch 4 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

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

tux_transaction.c

```
/*************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/29 16:24:31 $

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

#include <varargs.h>
#include <errno.h>


#include "tpcc.h"
#include "atmi.h"
#include "Uunix.h"
int user;


neworder_trans *neworder_ptr;
payment_trans  *payment_ptr;
ordstat_trans *ordstat_ptr;
stocklev_trans *stocklev_ptr;
delivery_trans *delivery_ptr;

long result;

static tux_error(format, va_alist)
   char *format;
   va_dcl
   {
   va_list argptr;

   va_start(argptr);
   vmessage(format, argptr);

   message("Tuxedo error %d\n", tperrno);

   errno = Uunixerr;
   if (tperrno == TPEOS)
      syserror("Tuxedo encountered O/S error\n");

   if (tperrno != TPESVCERR)
                error("EXITING !!!\n");
   else
                message("Retrying transation\n");

   }


transaction_begin(u)
   int u;
   {

   /* keep track of which user we are (for error messages only) */
   user = u;

   /* attach to Tuxedo */
   if (tpinit( (TPINIT *)NULL) == -1)
      tux_error("Failed to attach to Tuxedo\n");

   /* allocate structures for each transaction */
   neworder_ptr = (neworder_trans *)tpalloc("CARRAY", NULL, sizeof(neworder_trans));
   payment_ptr = (payment_trans *)tpalloc("CARRAY", NULL, sizeof(payment_trans));
   ordstat_ptr = (ordstat_trans *)tpalloc("CARRAY", NULL, sizeof(ordstat_trans));
   stocklev_ptr = (stocklev_trans *)tpalloc("CARRAY", NULL, sizeof(stocklev_trans));
   delivery_ptr = (delivery_trans *)tpalloc("CARRAY", NULL, sizeof(delivery_trans));
   if (neworder_ptr == NULL || payment_ptr == NULL || ordstat_ptr == NULL
      || stocklev_ptr == NULL || delivery_ptr == NULL)
      tux_error("Unable to allocate Tuxedo memory\n");
   }



transaction_done()
   {
   if (tpterm() == -1)
      tux_error("Unable to detach from Tuxedo\n");
   }
```

```
void neworder_transaction(t)
    neworder_trans *t;
    {
    *neworder_ptr = *t;
    while (tpcall("NEWO_SVC", (char *)neworder_ptr, sizeof(neworder_trans),
            (char **)(&neworder_ptr), &result, TPSIGRSTRT|TPNOTIME) == -1) {
        tux_error("Tuxedo failed for neworder transaction\n");
                *neworder_ptr = *t;
                }
    *t = *neworder_ptr;
    }


void payment_transaction(t)
    payment_trans *t;
    {
    *payment_ptr = *t;
    while (tpcall("PMT_SVC", (char *)payment_ptr, sizeof(payment_trans),
                  (char **)(&payment_ptr), &result, TPSIGRSTRT|TPNOTIME) == -1) {
        tux_error("Tuxedo failed for payment transaction\n");
                *payment_ptr = *t;
                }
    *t = *payment_ptr;
    }


void ordstat_transaction(t)
    ordstat_trans *t;
    {
    *ordstat_ptr = *t;
    while (tpcall("ORDS_SVC", (char *)ordstat_ptr, sizeof(ordstat_trans),
                   (char **)(&ordstat_ptr), &result, TPSIGRSTRT|TPNOTIME) == -1) {
                tux_error("Tuxedo failed for ordstat transaction\n");
                *ordstat_ptr = *t;
                }
    *t = *ordstat_ptr;
    }



stocklev_transaction(t)
    stocklev_trans *t;
    {
    *stocklev_ptr = *t;
    while (tpcall("STKL_SVC", (char *)stocklev_ptr, sizeof(stocklev_trans),
                   (char **)(&stocklev_ptr), &result, TPSIGRSTRT|TPNOTIME) == -1) {
        tux_error("Tuxedo failed for stocklev transaction\n");
                *stocklev_ptr = *t;
                }
    *t = *stocklev_ptr;
    }


delivery_init(u)
    int u;
    {
    }


delivery_enque(t)
    delivery_trans *t;
    {
    gettimeofday(&t->enque[0], NULL);
    t->status = OK;

    *delivery_ptr = *t;
    while (tpacall("DVRY_SVC", (char *)delivery_ptr, sizeof(delivery_trans),
                TPNOREPLY) == -1) {
        tux_error("Tuxedo failed enqueing delivery transaction\n");
                *delivery_ptr = *t;
                }
    }


delivery_done()
    {
    }
```

random.c

```
/*********************************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/27 13:18:41 $

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

double drand48();

char lastNames[1000][16];
char customerData1[10][301];
char customerData2[10][201];
char stockData1[10][27];
char stockData2[10][25];
char historyData1[10][13];
char historyData2[10][13];
char citystreetData1[10][11];
char citystreetData2[10][11];
char firstNameData1[10][9];
char firstNameData2[10][9];
char StockDistrict[10][25];
char phoneData[10][17];

static long RandySeedIter = 7;

void GenerateLastNames()
{
    int      i;
    char *name;
    static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
                        "ESE", "ANTI", "CALLY", "ATION", "EING"};

    for(i = 0; i < 1000; i++) {
        name = lastNames[i];
        strcpy(name, n[(i/100)%10]);
        strcat(name, n[(i/10) %10]);
        strcat(name, n[(i/1)  %10]);
    }
}

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

```
    length = RandomNumber(min, max);

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

    return length;
    }

ID RandomWarehouse(local, scale, percent)
    ID local;
    ID scale;
    int percent;   /* percent of remote transactions */
    {
    ID w_id;

    /* For the given percent of the time, pick the local warehouse */
    if (RandomNumber(1, 100) > percent || scale == 1)
        w_id = local;

    /* Otherwise, pick a non-local warehouse */
    else
        {
        w_id = RandomNumber(2, scale);
        if (w_id == local)
            w_id = 1;
        }
    return w_id;
    }


/* Initialize a table of Random strings for the stock-district
   field in the stock table.  We can use a table of 10 elements
   and select randomly from this table via rule 4.3.2.2 in
   the TPC-C spec */
void InitRandomStrings()
{
    int i;

    for (i=0; i < 10; i++) {
                        MakeAlphaString(24,24,&StockDistrict[i]);

                        MakeAlphaString(300,300,&customerData1[i]);
                        MakeAlphaString(0,200,&customerData2[i]);

                        MakeAlphaString(26,26,&stockData1[i]);
                        MakeAlphaString(0,24,&stockData2[i]);

                        MakeAlphaString(12,12,&historyData1[i]);
                        MakeAlphaString(0,12, &historyData2[i]);

                        MakeAlphaString(10,10,&citystreetData1[i]);
                        MakeAlphaString(0 ,10,&citystreetData2[i]);

                        MakeAlphaString(8,8,&firstNameData1[i]);
                        MakeAlphaString(0,8,&firstNameData2[i]);

                        MakeNumberString(16,16,&phoneData[i]);
                    }
                    GenerateLastNames();
}

int MakeAlphaString(min, max, str)
    int min;
    int max;
    TEXT str[];
    {
    static char character[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    int length;
    int i;

    length = RandomNumber(min, max);

    for (i=0;  i<length;  i++) {
        /* NOTE:  we use sizeof(character)-2 because of the following:
           subtract 1 because we are numbering from 0 instead of 1 and
           subtract 1 because the sizeof(character) is 1 greater than
           the data in character because of the inivisible C string
           terminator at the end. */
        str[i] = character[RandomNumber(0, sizeof(character)-2)];
    }
    str[length] = '\0';

    return length;
    }

void RandomPermutation(perm, n)
    int perm[];
    int n;
    {
    int i, r, t;

    /* generate the identity permutation to start with */
    for (i=1; i<=n; i++)
        perm[i] = i;

    /* randomly shuffle the permutation */
    for (i=1; i<=n; i++)
        {
        r = RandomNumber(i, n);
        t = perm[i]; perm[i] = perm[r]; perm[r] = t;
        }
    }


void RandomDelay(mean, adjust)
/*************************************************************************
random_sleep sleeps according to the TPC specification
*************************************************************************/
    double mean;
    double adjust;
    {
    double secs;
    double exponential();

    secs = exponential(mean);

    delay(secs+adjust);
    }

double exponential(mean)
/*************************************************************************
exponential generates a reverse exponential distribution
*************************************************************************/
    double mean;
    {
    double x;
    double log();

#ifdef USE_DRAND48
    x = -log(1.0-drand48()) * mean;
#else
    x = -log(1.0-randy()) * mean;
#endif

    return x;
    }

void SetRandomSeed(val)
```

```
long val;
   {
#ifdef USE_DRAND48
   srand48(val);
#else
   RandySeedIter = val;
   randy();
#endif
   }

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

/* Random number generator from Proceeding of the ACM */
#define RANDY_A_VAL 16807
/* 2^31 - 1 */
#define RANDY_M_VAL 2147483647
/* m / a */
#define RANDY_Q_VAL 127773
/* m % a */
#define RANDY_R_VAL 2836

double randy()
{
  long    hi, lo, test;

  hi = RandySeedIter / RANDY_Q_VAL;
  lo = RandySeedIter % RANDY_Q_VAL;

  test = (RANDY_A_VAL * lo) - (RANDY_R_VAL * hi);
  RandySeedIter = (test > 0) ? test : test + RANDY_M_VAL;

  return( (double)RandySeedIter / (double)RANDY_M_VAL );
}  /* end of fn randy  */
```

ncpu.c

```
#include <sys/mpctl.h>

main()
{
        printf("%d\n", mpctl(MPC_GETNUMSPUS, 0, 0));
}
```

Get_master_size.sh

```
#!/usr/bin/sh
#*****************************************************************************
#@(#) Version: A.10.10 $Date: 97/12/15 13:16:43 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************

bm=y

in_device=n
in_db=n

logical_name=
physical_name=
device_size=

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

                                in_device=y
                                in_db=n
                                ;;

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

                                # Start the new database
                                db_size=0

                                in_db=y
                                ;;


                log)            # This disk is the log disk for the current db
                                ;;

                vstart=*)
                                ;;

                mirror=*)
                                ;;

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

                segment=*)
                                ;;

                DEVICE_END)             # Complete the device

                                if [ "$in_device" = "y" -a "$logical_name" = "master" ]
                                then
                                        echo $db_size
                                fi
                                exit 0
                                ;;

                *)              # could be one of several
                                if [ "$in_device" = "y" ]
                                then
                                        if [ "$logical_name" = "" ]
                                        then
                                                logical_name=$token
                                        elif [ "$physical_name" = "" ]
                                        then
                                                physical_name=$token
                                        elif [ "$device_size" = "" ]
                                        then
                                                device_size=$token
                                        else
                                                echo
```

```
                                        fi
                                else
                                        echo
                                fi

                                ;;
        esac
done
```

Set_processors.load

```
#! /usr/bin/sh

nprocessors=‘ncpu‘
loopcount=0
num=0

# get list of Engin pids
list=‘ps -u tpcc | grep dataserver | awk ’     {print $1;}‘
echo "List of dataserver pids =" $list

# Count engines and assign pids to processors
num=0
for pp in $list
do
        set_affinity $num $pp
        num=‘expr $num + 1‘
done
echo Number of processors = $nprocessors
echo Number of Engines = $num
```

set_processors

```
#! /usr/bin/sh

nprocessors=‘ncpu‘
loopcount=0
num=0

while [ 1 ]
do
        loopcount=‘expr $loopcount + 1‘

        # get list of Engin pids
        list=‘ps -u tpcc | grep dataserver | awk ’     {print $1;}‘
        echo "List of dataserver pids =" $list

        # Count engines and assign pids to processors
        num=0
        for pp in $list
        do
                set_affinity $num $pp
                num=‘expr $num + 1‘
        done
        echo Number of processors = $nprocessors
        echo Number of Engines = $num

        # if we have one engine per processor, be happy
        if [ $nprocessors = $num ]
        then
                echo "Number of engines == number of processors. Looks good"
                break;
        fi

        if [ $loopcount = 5 ]
        then
                echo " ################################################"
                echo " # "
                echo " # WARNING!! Number of Processors != Number of Engines!"
                echo " # "
                echo " # >>>> I have GIVEN UP after 5 trys!!!!"
                echo " # "
                echo " ################################################"
                break;
        fi

        echo " ################################################"
        echo " # WARNING!! Number of Processors != Number of Engines!"
        echo " # I will sleep and try again"
        echo " ################################################"
        sleep 15
done
```

Init_server

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

isql -Usa -P <<- EOF
use master
go

sp_dboption tpcc,"trunc log on chkpt",false
go

use tpcc
go

checkpoint
go

dbcc tune(log_prealloc,4,"on")
go

dbcc traceon(1131)
go

dbcc tune(des_greedyalloc, 4,"delivery", "on")
go
dbcc tune(des_greedyalloc, 4,"stock", "on")
go
dbcc tune(des_greedyalloc, 4,"customer", "on")
go
dbcc tune(des_greedyalloc, 4,"new_order", "on")
go
dbcc tune(des_greedyalloc, 4,"orders", "on")
go
dbcc tune(des_greedyalloc, 4,"order_line", "on")
go

dbcc tune(des_bind, 4, "warehouse")
go
dbcc tune(des_bind, 4, "district")
go
dbcc tune(des_bind, 4, "item")
go
dbcc tune(des_bind, 4, "stock")
go
dbcc tune(des_bind, 4, "order_line")
go
dbcc tune(des_bind, 4, "orders")
go
```

```
dbcc tune(des_bind, 4, "new_order")
go
dbcc tune(des_bind, 4, "customer")
go
dbcc tune(des_bind, 4, "history")
go

dbcc tune(des_bind, 4, "neworder_local")
go
dbcc tune(des_bind, 4, "neworder_remote")
go
dbcc tune(des_bind, 4, "payment_byid")
go
dbcc tune(des_bind, 4, "payment_byname")
go
dbcc tune(des_bind, 4, "order_status_byid")
go
dbcc tune(des_bind, 4, "order_status_byname")
go
dbcc tune(des_bind, 4, "delivery")
go
dbcc tune(des_bind, 4, "stock_level")
go

EOF

isql -Usa -P <<- EOF
dbcc tune(maxwritedes, 5)
go
dbcc tune(deviochar, -1, "8")
go
dbcc tune("doneinproc", 0)
go
dbcc tune("cleanup", 0)
go
dump tran tpcc with truncate_only
go
EOF

isql -Usa -P <<- EOF
use tpcc
go
sp_logiosize "4"
go
EOF
```

Cache_bind.sh

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

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

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

use tpcc
go
checkpoint
go

/*
** Cache c_log
*/

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

sp_bindcache "c_wid", "tpcc", "sysindexes"
go
sp_bindcache "c_wid", "tpcc", "sysindexes", "sysindexes"
go

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

use tpcc
go
checkpoint
go

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

/*
** Cache New Order
*/

sp_bindcache "c_no", "tpcc", "new_order"
go
sp_bindcache "c_no_order_index", "tpcc", "new_order", "no_clu"
go


/*
** Cache Order Line
*/
sp_bindcache "c_ol", "tpcc", "order_line"
go
sp_bindcache "c_ol_index", "tpcc", "order_line", "ol_clu"
go

/*
** Cache orders
*/

sp_bindcache "c_orders", "tpcc", "orders"
go
sp_bindcache "c_no_order_index", "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

/*
** Cache c_customer_index
*/

sp_bindcache "c_cust_index", "tpcc", "customer", "c_clu"
go

/*
** Cache c_customer_non_index
*/

sp_bindcache "c_cust_non_index", "tpcc", "customer", "c_non1"
go

/*
** Default cache
*/

sp_bindcache "default data cache", "tpcc", "history"
go
EOF
```

Set_affinity.c

```
#include <sys/types.h>
#include <sys/syscall.h>
#include <sys/mp.h>
main(argc, argv)
int             argc;
char            *argv[];
{
   spu_t        spuid;
   pid_t        pid;

   if (argc < 2) {
                printf("%s spunum pid\n", argv[0]);
                exit(1);
   }
   spuid = atoi(argv[1]);
   while(argc-- > 2) {
                pid = atoi(argv[argc]);
                if (spuid=mpctl(MPC_SETPROCESS_FORCE, spuid, pid) == -1)        {
                   perror("mpctl");
                   exit(1);
                }
                spuid=mpctl(MPC_SETPROCESS, MPC_SPUNOCHANGE, pid);
                printf ("pid = %8d spuid = %8d\n", pid, spuid);
   }
}
```

Appendix B Database Design

The source code for the process to define, create and populate the Sybase Adaptive Server Enterprise v12.5 TPC-C database is included in this appendix.

B.1 Main Shell Scripts

build

```
#!/usr/bin/csh
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:10:21 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

source ~tpcc/TESTENV
build.real
```

build.real

```
#!/usr/bin/ksh
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:08:12 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

remsh $SERVER -n create_devices
load_database
```

create_devices

```
#!/usr/bin/sh -f
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:07:58 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

#
# Clean up some stuff first
#
rm -rf ~tpcc/logs/dev_create.OUT
rm -rf ~tpcc/logs/*.log

#
# Create the log file
#
exec > ~tpcc/logs/dev_create.OUT 2>&1

shutdown_server.sh
rm -f ~tpcc/dev/errorlog

echo ‘date‘ "Started bld_system"

#
# Build the master device.
#
if [ ${DATABASE_REV} = "12.5" ]
then
master_size=‘get_master_size.sh < /project/tpcc/load/devices‘
(cd /project/tpcc/dev; dataserver -f -dmaster -b${master_size}M -z${TPCC_PAGESIZE}k)
else
bmbinary=$SQL_RELEASE/bin/buildmaster
(cd /project/tpcc/dev; ‘devcreate.sh buildmaster $bmbinary < /project/tpcc/load/devices‘)
fi

#
# Boot server, run installmaster, shutdown server
#
run_server.load - -T1608
isql -Usa -P < $SQL_RELEASE/$SYBASE_ASE/scripts/installmaster > /project/tpcc/logs/$$_im.log

#
# Reboot, build devices, database, and segments
#
echo ‘date‘ "Creating devices, databases and segments"
devcreate.sh sql System10 < /project/tpcc/load/devices | isql -e -Usa -P
echo ‘date‘ " Finished building database"

#
# Create tables, some indexes, and administrative procs.
#
tpcc_tables.sh

#
# If parallel loader, then partition the tables
#
if [ ${PARALLEL_LOADER} = "1" ]
then
echo "Partitioning tables"
tpcc_partition.sh
else
#
# Create all indexes up front
#
echo "Creating indexes up front"
index_wdi
index_customer
index_stock
index_neworder
index_orders
index_orderline
fi

# Truncate log, checkpoint, and shutdown
dumptran_server.sh master
dumptran_server.sh tpcc
shutdown_server.sh
```

load_database

```
#!/usr/bin/ksh
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:08:26 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

rm -rf ~tpcc/logs/build.OUT
```

```
exec > ~tpcc/logs/build.OUT 2>&1

#
# Load the data; shutdown again.
#
echo ‘date‘ " Started loading data"

#
# startup server without logging to prepare for build
#

if [ ${PARALLEL_LOADER} = "1" ]
then
run_server.load
else
run_server.load - -T699
fi

#
# Setup good loading parameters
#
isql -Usa -P$PASSWORD << EOF
use master
go

sp_dboption tpcc,"select into/bulkcopy",true
go

sp_dboption tpcc,"trunc log on chkpt",true
go

use tpcc
go

checkpoint
go
EOF


#
# Load everything.  Some go in parallel, some in serial.
#
load_wdi
load_history &
load_stock  &
load_customer &
load_orders &
load_neworder

wait

if [ ${PARALLEL_LOADER} = "1" ]
then
#
# Unpartition if doing parallel load
#
tpcc_unpartition.sh
else
#
# Need to restart server for non-parallel
#
stop_server
run_server.load
index_wdi
index_customer  &
index_cnon      &
index_stock     &
fi

wait

#
# Create stored procedures
#
tpcc_proc.sh

tpcc_partition_history.sh

#
# Get table sizes
#
echo ‘date‘ -- Done building, get the table sizes
table_size.sh

stop_server

echo ‘date‘ "Finished bld_system"
```

load_customer

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/build.customer.OUT
exec > ~tpcc/logs/build.customer.OUT 2>&1

echo ‘date‘ " Started loading Customer"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "customer", ${LOAD_IO_SIZE})
go
EOF

if [ $PARALLEL_LOADER = "1" ]
then
load -t customer -p $LOAD_PARTITIONS -P "" 1 $DB_SIZE
index_customer
index_cnon
else
load -t customer 1 $DB_SIZE
fi

echo ‘date‘ " Finished loading Customer"
```

load_history

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/build.history.OUT
exec > ~tpcc/logs/build.history.OUT 2>&1

echo ‘date‘ " Started loading History"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "history", ${LOAD_IO_SIZE})
go
EOF
```

```
if [ $PARALLEL_LOADER = "1" ]
then
load -t history -p $LOAD_PARTITIONS -P "" 1 $DB_SIZE
else
load -t history 1 $DB_SIZE
fi

echo ‘date‘ " Finished loading History"
```

load_neworder

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/build.new_order.OUT
exec > ~tpcc/logs/build.new_order.OUT 2>&1

echo ‘date‘ " Started loading New Order"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "new_order", ${LOAD_IO_SIZE})
go
EOF

if [ $PARALLEL_LOADER = "1" ]
then
load -t new_order -p $LOAD_PARTITIONS -P "" 1 $DB_SIZE
index_neworder
else
load -t new_order 1 $DB_SIZE
fi

echo ‘date‘ " Finished loading New Order"
```

load_orders

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/build.orders.OUT
exec > ~tpcc/logs/build.orders.OUT 2>&1

echo ‘date‘ " Started loading Order/Orderline"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "orders", ${LOAD_IO_SIZE})
dbcc iosize("tpcc", "order_line", ${LOAD_IO_SIZE})
go
EOF

if [ ${PARALLEL_LOADER} = "1" ]
then
#
# Loads both orders and orderline
#
load -t orders -p $LOAD_PARTITIONS -P "" 1 $DB_SIZE
index_orders &
index_orderline
else
#
# For 12.0, both of these are loaded separately
#
load -t orders 1 $DB_SIZE     &
load -t orderline 1 $DB_SIZE
fi

echo ‘date‘ " Finished loading Order/Orderline"
```

load_stock

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/build.stock.OUT
exec > ~tpcc/logs/build.stock.OUT 2>&1

echo ‘date‘ " Started loading Stock"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "stock", ${LOAD_IO_SIZE})
go
EOF

if [ ${PARALLEL_LOADER} = "1" ]
then
load -t stock -p $LOAD_PARTITIONS -P "" 1 $DB_SIZE
index_stock
else
load -t stock 1 $DB_SIZE
fi

echo ‘date‘ " Finished loading Stock"
```

load_wdi

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/build.wdi.OUT
exec > ~tpcc/logs/build.wdi.OUT 2>&1

echo ‘date‘ " Started loading wdi"

load -t item
load -t warehouse 1 $DB_SIZE
load -t district  1 $DB_SIZE

if [ ${PARALLEL_LOADER} = "1" ]
then
index_wdi
fi

echo ‘date‘ " Finished loading wdi"
```

devcreate.sh

```
#!/usr/bin/sh
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:10:15 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

# @(#) devcreate.sh 1.1 6/7/95
#
#               scripts/devcreate.sh
#
#               Read a device file from stdin in the format given in format/devices
#               and ouput on stdout the SQL statements to create the devices,
#               databases and segments defined by the input device file.
#
#               The SQL is output in the following order
#
#               1) Disk inits and disk mirrors
#               2) Create databases
#               3) sp_addsegments and sp_extendsegments
#

if [ "$1" = "buildmaster" ]
then
                bm=y
                bmbinary=$2
elif [ "$1" = "sql" ]
then
                bm=n
                release=$2
else
                echo "Usage : $0 [builmaster buildmaster_binary |sql] < device_file" >&2
                echo "buildmaster - generate buildmaster command" >&2
                echo "sql [release] - generate SQL commands" >&2
                exit 1
fi

in_device=n
in_db=n

logical_name=
physical_name=
device_size=

vdevno=0

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

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

                                                in_device=y
                                                in_db=n
                                                ;;

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

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

                                                in_db=y
                                                ;;


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

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

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

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

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

                DEVICE_END)             # Complete the device

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

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

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

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

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


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

                ;;

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

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

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

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

current_db=
in_db=n
logdbinfo=
export in_db current_db logdbinfo
cat $db_s_file | while read dbname device size log
do
        if [ "$dbname" = "$current_db" ]
        then
                if [ -z "$log" ]
                then
                        if [ -z "$dbinfo" ]
                        then
                                dbinfo="on $device = $size"
                        else
                                dbinfo="$dbinfo, $device = $size"
                        fi
                else
                        if [ -z "$logdbinfo" ]
                        then
                                logdbinfo="$log $device = $size"
                        else
                                logdbinfo="$logdbinfo, $device = $size"
                        fi
                fi
        elif  [ "$in_db" = "y" ]
        then
                echo "create database $current_db"
                echo $dbinfo
                if [ -n "$logdbinfo" ]
                then
                        echo $logdbinfo
                fi
                echo go
                logdbinfo=

                current_db=$dbname
                dbinfo="on $device = $size"
                in_db=y
        else
                current_db=$dbname
                if [ -z "$log" ]
                then
                        dbinfo="on $device = $size"
                else
                        logdbinfo="$log $device = $size"
                fi
                in_db=y
        fi
done
#rm $db_s_file

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

current_db=
current_seg=
seg_db=
export current_seg current_db seg_db
```

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

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

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

in_device=no
export in_device
sort +0 -1 +2 -3 $seg_file | while read dbname segment device garbage
do
        if [ "$device" = "$current_dev" ]
        then
                false
        else
                if [ "$in_device" = "yes" ]
                then
                        if [ "$drop_segs" = "yes" ]
                        then
                                echo "sp_dropsegment 'default', $seg_db $current_dev"
                                echo go
                                echo "sp_dropsegment 'system', $seg_db $current_dev"
                                echo go
                        fi
                fi

                in_device=yes
                drop_segs=yes
        fi

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

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

        current_dev=$device
        current_db=$dbname
done

rm $seg_file

echo    "use master"
echo    go
echo    "checkpoint"
echo    go
```

tpcc_tables.sh

```
#!/usr/bin/sh -f
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:09:10 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

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 = 'history' )
        alter table history unpartition
go
if exists ( select name from sysobjects where name = 'history' )
        drop table history
go

if exists ( select name from sysobjects where name = 'orders' )
        drop table orders
go
if exists ( select name from sysobjects where name = 'new_order' )
        drop table new_order
go
if exists ( select name from sysobjects where name = 'item' )
        drop table item
go
if exists ( select name from sysobjects where name = 'district' )
        drop table district
go
if exists ( select name from sysobjects where name = 'warehouse' )
        drop table warehouse
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 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 */
) with max_rows_per_page = 1 on Swarehouse
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 */
) with max_rows_per_page = 10 on Sdistrict
go

create table item (
        i_id                    int,
        i_im_id                 int,
        i_name                  char(24),
        i_price                 float,
        i_data                  char(50)
) on Sitem
go

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),  /*- Updated by Phil P. */
        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 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

create table new_order (
        no_o_id                 int,
        no_d_id                 tinyint,
        no_w_id                 smallint,
) on Snew_order
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

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

checkpoint
go
EOF
```

tpcc_partition.sh

```
#!/usr/bin/sh -f
#*****************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:09:21 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************

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

alter table history    partition $LOAD_PARTITIONS
alter table customer   partition $LOAD_PARTITIONS
alter table new_order  partition $LOAD_PARTITIONS
alter table order_line partition $LOAD_PARTITIONS
alter table orders     partition $LOAD_PARTITIONS
alter table stock      partition $LOAD_PARTITIONS

checkpoint
go
EOF
```

tpcc_partition_history.sh

```
#!/usr/bin/sh -f
#*****************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:06:42 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************

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

alter table history    partition $HISTORY_PARTITIONS
go
EOF
```

tpcc_unpartition.sh

```
#!/usr/bin/sh -f
#*****************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:08:18 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*****************************************************************************

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

alter table history    unpartition
alter table customer   unpartition
alter table new_order  unpartition
alter table order_line unpartition
alter table orders     unpartition
alter table stock      unpartition

checkpoint
go
EOF
```

index_customer

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/index.customer.OUT
exec > ~tpcc/logs/index.customer.OUT 2>&1

echo ‘date‘ " Started building customer index"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "customer", ${LOAD_IO_SIZE})
go
alter table customer unpartition
go

drop index customer.c_clu
go
create unique clustered index c_clu
        on customer(c_w_id, c_id, c_d_id)
        with sorted_data
        on Scustomer
go

EOF

echo ‘date‘ " Finished building customer index"
```

index_neworder

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/index.neworder.OUT
exec > ~tpcc/logs/index.neworder.OUT 2>&1

echo ‘date‘ " Started building new_order index"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "new_order", ${LOAD_IO_SIZE})
```

```
go
alter table new_order  unpartition
go

drop index new_order.no_clu
go
create unique clustered index no_clu
        on new_order(no_w_id, no_d_id, no_o_id)
        with sorted_data
        on Snew_order
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips , 100, new_order)
go
EOF

echo ‘date‘ " Finished building new_order index"
```

index_orderline

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/index.order_line.OUT
exec > ~tpcc/logs/index.order_line.OUT 2>&1


echo ‘date‘ " Started building order_line index"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "order_line", ${LOAD_IO_SIZE})
go
alter table order_line unpartition
go

drop index order_line.ol_clu
go
create unique clustered index ol_clu
        on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
        with sorted_data
        on Sorder_line
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go
EOF

echo ‘date‘ " Finished building order_line index"
```

index_orders

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/index.orders.OUT
exec > ~tpcc/logs/index.orders.OUT 2>&1


echo ‘date‘ " Started building orders index"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "orders", ${LOAD_IO_SIZE})
go
alter table orders     unpartition
go

drop index orders.o_clu
go
create unique clustered index o_clu
        on orders(o_w_id, o_d_id, o_id)
        with sorted_data
        on Sorders
go
dbcc tune(ascinserts, 1, orders)
go
dbcc tune(oamtrips , 100, orders)
go
EOF

echo ‘date‘ " Finished building orders index"
```

index_stock

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/index.stock.OUT
exec > ~tpcc/logs/index.stock.OUT 2>&1

echo ‘date‘ " Started building stock index"

LOAD_IO_SIZE=‘expr ${TPCC_PAGESIZE} \* 8‘

isql -Usa -P$PASSWORD << EOF
use tpcc
go
dbcc iosize("tpcc", "stock", ${LOAD_IO_SIZE})
go
alter table stock      unpartition
go

drop index stock.s_clu
go
create unique clustered index s_clu
        on stock(s_i_id, s_w_id)
        with sorted_data
        on Sstock
go
dbcc tune(indextrips, 10, stock)
go
EOF

echo ‘date‘ " Finished building stock index"
```

index_wdi

```
#!/usr/bin/ksh

rm -rf ~tpcc/logs/index.wdi.OUT
exec > ~tpcc/logs/index.wdi.OUT 2>&1
```

```
echo ‘date‘ " Started building wdi index"

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

drop index warehouse.w_clu
go
create unique clustered index w_clu
       on warehouse(w_id)
       with sorted_data
       on Swarehouse
go
dbcc tune(indextrips, 100, warehouse)
go

drop index district.d_clu
go
create unique clustered index d_clu
       on district(d_w_id, d_id)
       with sorted_data
       on Sdistrict
go
dbcc tune(indextrips, 100, district)
go

drop index item.i_clu
go
create unique clustered index i_clu
       on item(i_id)
       with sorted_data
       on Sitem
go
dbcc tune(indextrips, 10, item)
go

EOF

echo ‘date‘ " Finished building wdi index"
```

dumptran_server.sh

```
#!/usr/bin/sh -f
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:07:36 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

isql -Usa -P$PASSWORD << EOF
dbcc tune(maxwritedes, 50)
go
dump tran $1 with truncate_only
go
use $1
go
dbcc tune(maxwritedes, 5)
go
checkpoint
go
EOF
```

run_server

```
#!/usr/bin/ksh
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:06:31 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

server_options="-c${CONFIG_FILE}"

# Do we override this ?
if [ $# != 0 ]
then
       # Pick up the remaining arguments.
       shift
       server_options=$server_options" $*"
fi

if [ ${DATABASE_REV} = "12.5" ]
then
#
# GA version 12.5
#
(cd ~tpcc/dev; dataserver -d$MASTER_DEVICE -T1231 -T3444 -T4050 -T4057 $server_options ) &
elif [ ${DATABASE_REV} = "12.0" ]
then
#
# GA version 12.0
#
(cd ~tpcc/dev; dataserver -d$MASTER_DEVICE -T7822 -T1231 -T3444 -T4050 -T4057 $server_options )&
else
#
# 11.9.3 run sequence
(cd ~tpcc/dev; dataserver -d$MASTER_DEVICE -T1130 -T1131 $server_options )&
fi

touch ~tpcc/dev/errorlog; tail -f -c1 ~tpcc/dev/errorlog | grep -q 'Recovery complete’

#
# Configure the server for TPC-C
#
tpcc_proc.sh
cache_bind.sh
init_server.sh
set.processors
```

run_server_load

```
#!/usr/bin/sh
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:08:22 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

server_options="-c${CONFIG_FILE}"

# Do we override this ?
if [ $# != 0 ]
then
       # Pick up the remaining arguments.
       shift
       server_options=$server_options" $*"
fi
```

```
(cd ~tpcc/dev; dataserver -d$MASTER_DEVICE -T1231 -T3444 -T4050 -T4057 $server_options ) &

touch ~tpcc/dev/errorlog; tail -f -c1 ~tpcc/dev/errorlog | grep -q 'Recovery complete'

(sleep 10; set.processors.load) &
```

shutdown_server.sh

```
#!/usr/bin/sh -f
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:10:18 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

# Now shut the server down
isql -Usa -P$PASSWORD <<EOF
shutdown
go
EOF

# The server may not be completely shutdown yet.
# Wait a bit longer just to be sure.
# (We really would like some guarantee)
sleep 20
```

stop_server

```
#!/usr/bin/sh -f
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:10:04 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

isql -Usa -P$PASSWORD -e << EOF
use tpcc
go
dbcc tune(maxwritedes, 50)
go
checkpoint
go
dbcc tune(maxwritedes, 5)
go
shutdown
go
EOF
```

table_size.sh

```
#!/usr/bin/sh
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/29 16:10:12 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

isql -Usa -P <<- EOF

use tpcc
go

sp_spaceused history,1
go

sp_spaceused new_order,1
go

sp_spaceused orders,1
go

sp_spaceused order_line,1
go

sp_spaceused warehouse, 1
go

sp_spaceused district, 1
go

sp_spaceused customer, 1
go

sp_spaceused item, 1
go

sp_spaceused stock,1
go

EOF
```

devices

```
#
# Config for 11960 warehouse setup
#
DEVICE master master 1000 db=tpcc size=1000 segment=default segment=system DEVICE_END

DEVICE w_d_i w_d_i 200 db=tpcc size=200 segment=Swarehouse segment=Sdistrict segment=Sitem DEVICE_END

DEVICE new_order1 new_order1 4000 db=tpcc size=4000 segment=Snew_order DEVICE_END

DEVICE history1 history1 24000 db=tpcc size=24000 segment=Shistory DEVICE_END

DEVICE log1 log1 32000 db=tpcc size=32000 log DEVICE_END
DEVICE log2 log2 32000 db=tpcc size=32000 log DEVICE_END
DEVICE log3 log3 32000 db=tpcc size=32000 log DEVICE_END

DEVICE orders1 orders1 16000 db=tpcc size=16000 segment=Sorders DEVICE_END

DEVICE order_line1 order_line1 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line2 order_line2 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line3 order_line3 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line4 order_line4 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line5 order_line5 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line6 order_line6 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line7 order_line7 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line8 order_line8 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line9 order_line9 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line10 order_line10 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line11 order_line11 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line12 order_line12 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line13 order_line13 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line14 order_line14 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line15 order_line15 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
```

```
DEVICE order_line16 order_line16 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line17 order_line17 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line18 order_line18 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line19 order_line19 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line20 order_line20 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line21 order_line21 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line22 order_line22 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END
DEVICE order_line23 order_line23 12060 db=tpcc size=12060 segment=Sorder_line DEVICE_END

DEVICE customer1 customer1 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer2 customer2 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer3 customer3 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer4 customer4 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer5 customer5 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer6 customer6 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer7 customer7 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer8 customer8 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer9 customer9 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer10 customer10 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer11 customer11 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer12 customer12 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer13 customer13 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer14 customer14 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer15 customer15 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer16 customer16 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer17 customer17 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer18 customer18 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer19 customer19 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer20 customer20 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer21 customer21 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer22 customer22 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END
DEVICE customer23 customer23 10800 db=tpcc size=10800 segment=Scustomer DEVICE_END

DEVICE c_index1 c_index1 24000 db=tpcc size=24000 segment=Sc_index DEVICE_END

DEVICE stock1 stock1 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock2 stock2 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock3 stock3 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock4 stock4 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock5 stock5 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock6 stock6 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock7 stock7 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock8 stock8 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock9 stock9 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock10 stock10 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock11 stock11 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock12 stock12 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock13 stock13 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock14 stock14 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock15 stock15 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock16 stock16 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock17 stock17 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock18 stock18 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock19 stock19 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock20 stock20 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock21 stock21 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock22 stock22 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
DEVICE stock23 stock23 17430 db=tpcc size=17430 segment=Sstock DEVICE_END
```

tpcc_warm.sh

B.2 Code to Populate

load.c

```
/**********************************************************************
 @(#) Version: A.10.10 $Date: 2001/08/27 13:15:33 $

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

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

To Do:
  o Need to add CLAST and CID constants and way to set them

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

#include <unistd.h>
#include <time.h>
#include <stdio.h>
#include <cspublic.h>
#include <bkpublic.h>
#include "tpcc.h"
#include "random.h"

/* configurable parameters */
#define NURAND_C 123

typedef unsigned long   BitVector;
#define WSZ             (sizeof(BitVector)*8)
#define nthbit(map,n)   map[(n)/WSZ] &  (((BitVector)0x1)<< ((n)%WSZ))
#define setbit(map,n)   map[(n)/WSZ] |= (((BitVector)0x1)<< ((n)%WSZ))

int load_item;
int load_warehouse;
int load_district;
int load_history;
int load_orders;
int load_orderline;
int load_neworder;
int load_customer;
int load_stock;
int sliceNum;
int numOfSlices = -1;

ID w1, w2;
ID i1, i2;

#define PASSWORD_LENGTH 64
char password[PASSWORD_LENGTH];

void LoadWarehouse();
void LoadDistrict();
void LoadItems();
void LoadOrders();
void Orders();
void configure();

#define print_prologue(name)                                    \
do {                                                            \
   printf("Loading %s table from warehouse %d to %d into partition %d\n",  \
                                        (name),  w1, w2, sliceNum);                             \
} while (0)

#define print_prologue_item(name)                               \
```

```
do {                                                    \
   printf("Loading %s table from warehouse %d to %d and from item %d to %d into partition %d\n", (name),  w1, w2, i1, i2,
sliceNum);                  \
} while (0)

int main(argn, argv)
   int argn;
   char **argv;
   {
   ID starting_wid, ending_wid, winc, last_wid;
   ID starting_iid, ending_iid, iinc;
   int current_slicenum;
   pid_t pid = 0, parallel_parent = 0;
   ID  w_id;

   configure(argn, argv);

   /* Need to create the Random Strings the same for each load */
   SetRandomSeed(100);
   InitRandomStrings();

   /* These are so small and load fast, always do sequentially */
   sliceNum = 1;
   if (load_item)        { begin_load(); LoadItems(); return 0;}
   if (load_warehouse)   { begin_load(); LoadWarehouse(w1, w2); return 0; }
   if (load_district)    { begin_load(); LoadDistrict(w1, w2); return 0; }

   /*
    ** Except for item and stock all others are parallely
    ** loaded based on the warehouse id. Item and Stock
    ** are loaded parallely based on the item.
    **
    ** We make sure that at least 2 waehouses are given
    ** to each thread. For Item and Stock we don't need
    ** to do this check, because we will be sharing the
    ** load of MAXITEMS equally among all threads.
    */
   if (load_stock || load_customer || load_orders || load_neworder ||
                load_history) {
   if ( numOfSlices != -1  &&
      ((load_item || load_stock) || (w2 >(numOfSlices*2))))
   {
                winc = w2 / numOfSlices;
                iinc = MAXITEMS / numOfSlices;

                last_wid = w2;
                for( starting_wid = w1, ending_wid = winc, current_slicenum =1,
                    starting_iid = 1, ending_iid = iinc;
                    current_slicenum <= numOfSlices;
                    starting_wid += winc,
                    ending_wid += winc,
                    starting_iid += iinc,
                    ending_iid += iinc,
                    current_slicenum++)
                {
                    if ( current_slicenum == numOfSlices ) {
                                /* Last thread. Give the remaining */
                                ending_wid = last_wid;
                                ending_iid = MAXITEMS;
                    }
                    if ( load_item || load_stock) {
                                i1 = starting_iid;
                                i2 = ending_iid;
                    } else {
                                w1 = starting_wid;
                                w2 = ending_wid;
                    }
                    sliceNum = current_slicenum;
                    if ( (pid = fork()) < 0 ) {
                       printf("Fork failed \n");
                                exit(1);
                    } else if ( pid > 0 ) {
                                parallel_parent = 1;
                                /* Parent */
                                continue;
                    } else {
                                parallel_parent = 0;
                                /* child */
                                break;
                    }
                }
   }

   if ( parallel_parent ) {
                while( wait(0) != -1 );
                exit(0);
   } else {
                /* From here on out, things can be random for each thread */
                Randomize();
      if (load_stock)       { LoadStock(w1, w2); }
      if (load_customer)    { LoadCustomer(w1, w2); }
                if (load_orders)      { LoadOrders(w1, w2); }
                if (load_neworder)    { LoadNeworder(w1, w2); }
      if (load_history)     { LoadHist(w1, w2); }
                return 0;
   }
   }
}


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

Warehouse

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

void
LoadWarehouse(first, last)
   ID first, last;
   {
   warehouse_row r[1];
   ID w_id;

   begin_warehouse_serial_load();

   printf("loading warehouses %d to %d\n",first,last);
   r->W_YTD = 30000000;
   for (w_id = first;  w_id <= last; w_id++) {
      printf("loading warehouse %d\n",w_id);

      r->W_ID = w_id;
      MakeAlphaString(6, 10, r->W_NAME);
      MakeAddress(r->W_STREET_1, r->W_STREET_2, r->W_CITY, r->W_STATE,
                r->W_ZIP);
      r->W_TAX = RandomNumber(0, 2000) / 10000.0;

      warehouse_serial_load(r);
   }

   end_warehouse_serial_load();
}



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

District
```

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

void
LoadDistrict(first, last)
   ID first, last;
   {
   ID w_id, d_id;
   district_row r[1];

   begin_district_serial_load();
   r->D_YTD = 3000000;
   r->D_NEXT_O_ID = 3001;

   for (w_id = first;  w_id <= last; w_id++) {
      printf("loading districts for warehouse %d\n",w_id);
      r->D_W_ID = w_id;

      for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++) {
         r->D_ID = d_id;
         MakeAlphaString(6, 10, r->D_NAME);
         MakeAddress(r->D_STREET_1, r->D_STREET_2, r->D_CITY, r->D_STATE,
                  r->D_ZIP);
         r->D_TAX = RandomNumber(0, 2000) / 10000.0;

         district_serial_load(r);
      }
   }

   end_district_serial_load();
}


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

Item

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

void
LoadItems()
{
   item_row r[1];
   int perm[MAXITEMS+1];
   ID i_id;


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

   /* do for each item */
                  printf("loading item table\n");
   for (i_id = 1;  i_id <= MAXITEMS; i_id++) {
      /* Generate Item Data */
      r->I_ID = i_id;
      MakeAlphaString(14, 24, r->I_NAME);
      r->I_PRICE = RandomNumber(100,10000);
      MakeAlphaString(26, 50, r->I_DATA);
      if (perm[r->I_ID] <= (MAXITEMS+9)/10)
         Original(r->I_DATA);
      r->I_IM_ID  = RandomNumber(1, 10000);

      item_serial_load(r);
   }

   end_item_serial_load();
}



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

History

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


LoadHist(first, last)
   ID first, last;
{
   ID w_id, d_id, c_id;
   static history_row r;
   CS_BLKDESC *blkdesc;
   int bulk_h;

   print_prologue("History");
   bulk_h = bulk_open("tpcc", "history", password, &blkdesc, &sliceNum);

   for (w_id = first;  w_id <= last; w_id++) {
      printf("Loading history for warehouse %d\n",w_id);
      for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++) {
         for (c_id = 1; c_id <= CUST_PER_DIST; c_id++) {
                        r.H_C_D_ID = r.H_D_ID = d_id;
                        r.H_C_W_ID = r.H_W_ID = w_id;
                        r.H_C_ID = c_id;
                        CurrentDate(&r.H_DATE);
                        r.H_AMOUNT = 1000;
                        SelectHistoryData(r.H_DATA);
                        history_load(bulk_h, blkdesc, r);
                     }
                  }
   }
   end_history_load(bulk_h, blkdesc);
}


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

Customer

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

void Customer(ID w_id, int bulk_c, CS_BLKDESC *blkdesc);

LoadCustomer(first, last)
   ID first, last;
{
   ID w_id;
   int bulk_c;
   CS_BLKDESC *blkdesc;

   print_prologue("Customer");
   bulk_c = bulk_open("tpcc", "customer", password, &blkdesc, &sliceNum);

   for (w_id = first;  w_id <= last; w_id++) {
                  printf("Loading customer for warehouse %d\n",w_id);
      Customer(w_id, bulk_c, blkdesc);
   }
   end_customer_load(bulk_c, blkdesc);
}
```

```
void
Customer(w_id, bulk_c, blkdesc)
/*********************************************************************
Load customers for the given warehouse and district
*********************************************************************/
    ID w_id;
    int bulk_c;
    CS_BLKDESC *blkdesc;
    {
    int i;
    ID   id[CUST_PER_DIST+1];
    ID c_id;
    ID d_id;
    customer_row r[1];
    static int bad_credit_perm[DIST_PER_WARE+1][CUST_PER_DIST+1];

    /* 10% of customers will have bad credit */
    for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
        RandomPermutation(bad_credit_perm[d_id], CUST_PER_DIST);


    /* Order by customer id, then district */
    r->C_W_ID         = w_id;
    r->C_CREDIT_LIM   = 5000000;
    r->C_BALANCE      = -1000;
    r->C_YTD_PAYMENT  = 1000;
    r->C_PAYMENT_CNT  = 1;
    r->C_DELIVERY_CNT = 0;
    for (c_id=1; c_id <= CUST_PER_DIST; c_id++)
        {
        r->C_ID   = c_id;
        for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
            {
            r->C_D_ID = d_id;

            if (c_id <= 1000)
                LastName(c_id - 1, r->C_LAST);
            else
                LastName(NURandomNumber(255, 0, 999, NURAND_C), r->C_LAST);

            strcpy(r->C_MIDDLE, "OE");
            SelectFirstName(r->C_FIRST);
            MakeAddress(r->C_STREET_1, r->C_STREET_2, r->C_CITY, r->C_STATE,
                    r->C_ZIP);
            SelectPhoneData(r->C_PHONE);
            CurrentDate(&r->C_SINCE);
            if (bad_credit_perm[d_id][r->C_ID] <= ((CUST_PER_DIST + 9)/10))
                strcpy(r->C_CREDIT, "BC");
            else
                strcpy(r->C_CREDIT, "GC");
            r->C_DISCOUNT     = RandomNumber(0, 5000) / 10000.0;
            SelectClientData(r->C_DATA);

            customer_load(bulk_c, blkdesc, r);
            }
        }
}

void
LoadOrders(first, last)
    ID first, last;
{
    ID w_id, d_id;
    int ol_bulk, o_bulk;
    CS_BLKDESC *ol_blkdesc;
    CS_BLKDESC *o_blkdesc;

    ol_bulk = bulk_open("tpcc", "order_line", password, &ol_blkdesc, &sliceNum);
    o_bulk  = bulk_open("tpcc", "orders", password, &o_blkdesc, &sliceNum);
    print_prologue("Orders");

    for (w_id = first;  w_id <= last; w_id++) {
        printf("Loading Orders/OrderLine for warehouse %d\n",w_id);
        for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++) {
            Orders(w_id, d_id, o_bulk, ol_bulk, o_blkdesc, ol_blkdesc);
                }
    }
    end_orderline_load(ol_bulk, ol_blkdesc);
    end_order_load(o_bulk, o_blkdesc);
}

void
Orders(w_id, d_id, o_bulk, ol_bulk, o_blkdesc, ol_blkdesc)
    ID w_id, d_id;
    int o_bulk, ol_bulk;
    CS_BLKDESC *o_blkdesc, *ol_blkdesc;
{
    int cust[ORD_PER_DIST+1];
    ID o_id, ol_number;
    ID ol;
    order_row       r[1];
    orderline_row olr[1];
    int sum;

    r->O_W_ID = w_id;
    r->O_D_ID = d_id;

    RandomPermutation(cust, ORD_PER_DIST);

    r->O_ALL_LOCAL    = 1;
    olr->OL_QUANTITY  = 5;
    for (o_id = 1;  o_id <= ORD_PER_DIST;  o_id++) {
        r->O_ID   = o_id;
        r->O_C_ID = cust[o_id];
        CurrentDate(&r->O_ENTRY_D);

        if (r->O_ID <= 2100) r->O_CARRIER_ID = RandomNumber(1,10);
        else                    r->O_CARRIER_ID = EMPTY_NUM;

        /* map the range 1..n onto 5..15 for orderline count */
        r->O_OL_CNT = RandomNumber(5,15);

        /* load the order */
        order_load(o_bulk, o_blkdesc, r);

        /* generate the order lines */
                olr->OL_O_ID        = o_id;
                olr->OL_D_ID        = d_id;
        olr->OL_W_ID        = w_id;
                olr->OL_SUPPLY_W_ID = w_id;
        for (ol_number = 1;  ol_number <= r->O_OL_CNT; ol_number++) {
                    olr->OL_NUMBER      = ol_number;
                    olr->OL_I_ID        = RandomNumber(1, MAXITEMS);

            /* Store null CurrentDate in the DB as "01/01/1800 12:00:00AM" */
            if (o_id <= 2100) {
                        olr->OL_DELIVERY_D = *(&r->O_ENTRY_D);
                olr->OL_AMOUNT     = 0;
            } else {
                        EmptyDate(&olr->OL_DELIVERY_D);
                olr->OL_AMOUNT = RandomNumber(1, 999999);
                    }

            SelectStockDistrict(olr->OL_DIST_INFO);
            orderline_load(ol_bulk, ol_blkdesc, olr);
                }
    }
}

LoadNeworder(first, last)
```

```
    ID  first, last;
    {
    ID w_id, d_id;
    int no_bulk;
    CS_BLKDESC *blkdesc;

    print_prologue("NewOrder");
    no_bulk = bulk_open("tpcc", "new_order", password, &blkdesc, &sliceNum);

    for (w_id = first;  w_id <= last; w_id++) {
                printf("Loading NewOrder for warehouse %d\n",w_id);
        for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++) {
            neworder_row r[1];

            r->NO_D_ID = d_id;
            r->NO_W_ID = w_id;
            for (r->NO_O_ID=2101;  r->NO_O_ID <= ORD_PER_DIST; r->NO_O_ID++) {
                                neworder_load(no_bulk, blkdesc, r);
                    }
                }
    }

    end_neworder_load(no_bulk, blkdesc);
}

#define ITEM_BITVEC_SIZE ((MAXITEMS/(8*sizeof(BitVector)))+1)*sizeof(BitVector)

/*
** 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(first, last)
    ID first, last;
{
    int bulk_s;
    BitVector **perm;
    stock_row r[1];
    ID w_id;
    ID i_id;
    unsigned long count = 0;
    unsigned long checkPointTime, checkPointChunk, totalRowsToLoad;
    int i;
    long j;
    CS_BLKDESC *blkdesc;

    print_prologue_item("Stock");

    perm = (BitVector **) malloc((last-first)*sizeof(BitVector *));
    if (perm == NULL) {
                syserror("LoadStock: can't allocate memory for permutations\n");
    }
    /* select exactly 10% of items to be labeled "original" */
    for (w_id = first; w_id <= last; w_id++)
    {
                int index = w_id - first;
        perm[index] = (BitVector *)malloc(ITEM_BITVEC_SIZE);
        if (perm[index] == NULL) {
                                syserror("LoadStock: can't allocate memory\n");
                }
                (void) memset(perm[index],0,ITEM_BITVEC_SIZE);
                /* Mark exactly 10% of items as "original" */
                for (i = 0; i < (MAXITEMS+9)/10; i++) {
                                do {
                                                j = RandomNumber(0,MAXITEMS-1);
                                } while (nthbit(perm[index],j));
                                setbit(perm[index],j);
                }
    }

    /* do for each item and warehouse */
    bulk_s = bulk_open("tpcc", "stock", password, &blkdesc, &sliceNum);

    r->S_YTD        = 0;
    r->S_ORDER_CNT  = 0;
    r->S_REMOTE_CNT = 0;
    totalRowsToLoad = (i2 - i1 + 1) * (last - first + 1);
    /* Every 5% loaded, print message 8 */
    checkPointTime = checkPointChunk = (totalRowsToLoad)/20;
    for (i_id = i1;  i_id <= i2;  i_id++) {
        r->S_I_ID = i_id;
                if (checkPointTime < count) {
                                printf("Loaded %4.1lf%% of stock rows for partition
%d\n",100.0*((double)count/(double)totalRowsToLoad), sliceNum);
                                checkPointTime += checkPointChunk;
                }
        for (w_id = first; w_id <= last; w_id++) {
            /* Generate Stock Data */
            r->S_W_ID = w_id;
            r->S_QUANTITY = RandomNumber(10,100);
            SelectStockDistrict(r->S_DIST_01);
            SelectStockDistrict(r->S_DIST_02);
            SelectStockDistrict(r->S_DIST_03);
            SelectStockDistrict(r->S_DIST_04);
            SelectStockDistrict(r->S_DIST_05);
            SelectStockDistrict(r->S_DIST_06);
            SelectStockDistrict(r->S_DIST_07);
            SelectStockDistrict(r->S_DIST_08);
            SelectStockDistrict(r->S_DIST_09);
            SelectStockDistrict(r->S_DIST_10);
            SelectStockData(r->S_DATA);
                        if (nthbit(perm[w_id - first], r->S_I_ID -1)) {
                            Original(r->S_DATA);
                        }
                        stock_load(bulk_s,blkdesc,r);
                        count++;
        }
    }
    printf("finished loading stock items for warehouses %d to %d, partition %d\n",
                        first,last, sliceNum);

    end_stock_load(bulk_s, blkdesc);

    for (w_id = first; w_id <= last; w_id++) {
        free(perm[w_id-first]);
    }

    free(perm);
}

void
configure(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;
    int any_except_item,  any_at_all;

    /* use unbuffered I/O (for output to files) */
```

```
setvbuf(stdout, 0, _IONBF, 0);
setvbuf(stderr, 0, _IONBF, 0);

w1 = 1;
w2 = 1;
i1 = 1;
i2 = MAXITEMS;

/* define the defaults */
load_item = load_warehouse = load_district = load_history =
load_orders = load_neworder = load_customer = load_stock = NO;

/* do for each option */
while ((ch = getopt (argc, argv, "P:p:t:")) != EOF) {
   /* process according to options */
   switch ( ch ) {
      case 't':
         if     (strcmp(optarg, "warehouse") == 0)load_warehouse = YES;
         else if (strcmp(optarg, "district") == 0)  load_district = YES;
         else if (strcmp(optarg, "stock") == 0)  load_stock = YES;
         else if (strcmp(optarg, "item") == 0)  load_item = YES;
         else if (strcmp(optarg, "history") == 0)  load_history = YES;
         else if (strcmp(optarg, "orders") == 0)  load_orders = YES;
         else if (strcmp(optarg, "new_order") == 0)  load_neworder = YES;
         else if (strcmp(optarg, "customer") == 0)  load_customer = YES;
         else
            error("%s is not a valid table name\n", optarg);
                        break;

                case 'p':
                        numOfSlices = atoi(optarg);
                        printf("Number of partitions is %d\n",numOfSlices);
                        break;

      case 'P':
                        strncpy(password,optarg,PASSWORD_LENGTH-1);
                        break;

      default:
                        error("Bad runstring argument.\n");
         break;
      }
}


/* some common flags depending on tables asked for */
any_except_item = load_warehouse || load_district || load_stock ||
            load_history  || load_orders || load_orderline ||
                                    load_neworder || load_customer;
any_at_all = any_except_item || load_item;


/* if only asked for item, don't allow warehouse to be specified */
if (!any_except_item && load_item) {
   if (optind != argc)
      error("Don't specify warehouse when loading items");
} else {
   /* otherwise get the warehouse number */
   if (optind >= argc)
      error("Must specify warehouses to load\n");
   w1 = atoi(argv[optind++]);

   if (optind >= argc)
      w2 = w1;
   else
      w2 = atoi(argv[optind++]);

   if (w1 > w2)
      error("First warehouse is greater than last warehouse\n");
   if (w1 <= 0)
      error("Warehouse must be positive non-zero\n");
   }

   /* if no tables mentioned explicitly, then load them all */
   if (!any_at_all) {
      load_warehouse = load_district = load_history = load_orders =
      load_neworder = load_customer = load_stock = YES;
      load_item = (w1 == 1);
   }
}
```

bulk_sybase.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 2001/09/10 11:36:44 $

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

/*********************************************************************
**********************************************************************
Sybase Specific Routines
**********************************************************************
**********************************************************************/

#include "tpcc.h"
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <ctpublic.h>
#include <bkpublic.h>

#define BATCH 1024

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

#define COUNT_T 0
#define ID_T 1
#define MONEY_T 2
#define FLOAT_T 3
#define TEXT_T 4
#define DATE_T 5
#define LOGICAL_T 6

bind_parm parm[7] =
{
                /* COUNT */  {NULL, 0, CS_INT_TYPE, SYBINT4},
                /* ID            */                {NULL, 0, CS_INT_TYPE, SYBINT4},
                /* MONEY */ {NULL, 0, CS_FLOAT_TYPE, SYBFLT8},
                /* FLOAT */   {NULL, 0, CS_FLOAT_TYPE, SYBFLT8},
        /* TEXT */      {"",   1, CS_CHAR_TYPE, SYBCHAR},
                /* DATE */     {NULL, 0, CS_DATETIME_TYPE, SYBDATETIME},
                /* LOGICAL */                 {NULL, 0, CS_INT_TYPE, SYBINT4}
};

#define bulk_bind_adjustlen(column, name, address) \
        datalen[column] = strlen(address)
```

```
#define MAXOPENS 256

#define RETURN_IF(a,b)                                                  \
        if (a != CS_SUCCEED)                                            \
        {                                                               \
                fprintf(stderr, "Error in: %s\n", b);                   \
                return a;                                               \
        \
        }


#define EXIT_IF(a) if (a != CS_SUCCEED)\
        { fprintf(stderr, "FATAL ERROR! Line %d\n", __LINE__); exit(-1);}

CS_CONNECTION *dbconn[MAXOPENS];
int             count[MAXOPENS];

CS_RETCODE    init_db();
CS_RETCODE    connect_db();
CS_RETCODE    send_sql();
void          handle_returns();

extern  CS_INT   server_msg_handler();
extern  CS_INT   cl_err_handler();
extern  CS_INT   cs_err_handler();

void to_sybase_date(DATE *date, DBDATETIME *sybdate);

CS_CONTEXT    *cntx_ptr;
int bulk_open(database, table, password, blk_desc, sliceNum)
        CS_CHAR    *database;
        CS_CHAR    *table;
        CS_CHAR    *password;
        CS_BLKDESC **blk_desc;
        CS_INT          *sliceNum;
{
        int db;

        CS_RETCODE     retcode=0;
        CS_CONNECTION *conn_ptr;
        CS_COMMAND    *cmd_ptr;

        char    tblname[25];

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

        retcode = CS_SUCCEED;
        retcode = init_db(&cntx_ptr);
        EXIT_IF(retcode);

        strcpy(tblname, table);
        retcode = connect_db(cntx_ptr, &conn_ptr, "sa", "", table);
        EXIT_IF(retcode);

        retcode = ct_cmd_alloc(conn_ptr, &cmd_ptr);
        EXIT_IF(retcode);

        retcode = send_sql(cmd_ptr, "use tpcc");
        EXIT_IF(retcode);

        handle_returns(cmd_ptr);
        /* prepare to do a bulk copy */
        retcode = blk_alloc(conn_ptr, CS_CURRENT_VERSION, blk_desc);
        EXIT_IF(retcode);

        if ( *sliceNum != -1 )
        {
                retcode =  retcode = blk_props(*blk_desc, CS_SET, BLK_SLICENUM,
                sliceNum, CS_UNUSED, NULL) ;
                        EXIT_IF(retcode);
        }

        retcode = blk_init(*blk_desc, CS_BLK_IN,
                        tblname, strlen(tblname));
        EXIT_IF(retcode);

        return db;
}

CS_RETCODE
init_db(cntx_ptr)
CS_CONTEXT **cntx_ptr;
{
        CS_RETCODE    retcode=CS_SUCCEED;

        retcode = cs_ctx_alloc(CS_CURRENT_VERSION, cntx_ptr);
        RETURN_IF(retcode, "init_db: cs_ctx_alloc");

        retcode = ct_init(*cntx_ptr, CS_CURRENT_VERSION);
        RETURN_IF(retcode, "init_db: ct_init");

        retcode = ct_callback(*cntx_ptr, NULL, CS_SET,
                        CS_SERVERMSG_CB, (CS_VOID *)server_msg_handler);
        RETURN_IF(retcode, "init_db: ct_callback-server_msg");

        retcode = cs_config(*cntx_ptr, CS_SET, CS_MESSAGE_CB,
                        (CS_VOID *)cs_err_handler, CS_UNUSED, NULL);
        RETURN_IF(retcode, "init_db: cs_config-cs_err");

        retcode = ct_callback(*cntx_ptr, NULL, CS_SET, CS_CLIENTMSG_CB,
                        (CS_VOID *)cl_err_handler);
        RETURN_IF(retcode, "init_db: ct_callback- cl_err");
        return retcode;
} /* end of init_db() */

CS_RETCODE
connect_db(cntx_ptr, conn_ptr, user_name, password, table)
CS_CONTEXT     *cntx_ptr;
CS_CONNECTION  **conn_ptr;
CS_CHAR        *user_name;
CS_CHAR        *password;
CS_CHAR        *table;
{
        CS_RETCODE    retcode=0;
        CS_BOOL          bool;
        CS_INT       packetsize=4096;

        retcode = ct_con_alloc(cntx_ptr, conn_ptr);
        RETURN_IF(retcode, "connect_db: ct_con_alloc");

        retcode = ct_con_props(*conn_ptr, CS_SET, CS_USERNAME,
                        user_name, CS_NULLTERM, NULL);
        RETURN_IF(retcode, "connect_db: ct_con_props");

        retcode = ct_con_props(*conn_ptr, CS_SET, CS_PASSWORD,
                        password, CS_NULLTERM, NULL);
        RETURN_IF(retcode, "connect_db: ct_con_props");

        retcode = ct_con_props(*conn_ptr, CS_SET, CS_APPNAME,
                        table, CS_NULLTERM, NULL);
        RETURN_IF(retcode, "connect_db: ct_con_props");

        retcode = ct_con_props(*conn_ptr, CS_SET, CS_PACKETSIZE,
                        &packetsize, CS_UNUSED, NULL);
        RETURN_IF(retcode, "connect_db: ct_con_props");
```

```
		bool = CS_TRUE;
		retcode = ct_con_props(*conn_ptr, CS_SET, CS_BULK_LOGIN,
					&bool, CS_UNUSED, NULL);
		RETURN_IF(retcode, "connect_db: ct_con_props");

		/*establish a connection with the server specified by DSQUERY */
		retcode = ct_connect(*conn_ptr, NULL, 0);
		RETURN_IF(retcode, "connect_db: ct_connect");
		return retcode;
} /* connect_db() */

CS_RETCODE
send_sql(cmd_ptr, sqltext)
CS_COMMAND	*cmd_ptr;
CS_CHAR		*sqltext;
{
		CS_RETCODE	retcode=0;

		retcode = ct_command(cmd_ptr, CS_LANG_CMD, sqltext,
					CS_NULLTERM, CS_UNUSED);
		RETURN_IF(retcode, "send_sql:ct_command");

		retcode = ct_send(cmd_ptr);
		RETURN_IF(retcode, "send_sql:ct_send");
		return retcode;
} /* end of send_sql()  */

void
handle_returns(cmd_ptr)
CS_COMMAND	*cmd_ptr;
{
		CS_INT		result_type;

		while (ct_results(cmd_ptr, &result_type) == CS_SUCCEED)
		{
			;
		}
		return;
} /* end of handle_returns()  */


CS_DATAFMT	datafmt[255];
CS_INT		datalen[255];

CS_RETCODE
bulk_bind(int column, char *name, void *address, int type, CS_BLKDESC *blk_desc)
{
		CS_RETCODE	retcode=0;


		datafmt[column].locale = 0;
		datafmt[column].count  = 1;
		datafmt[column].datatype = parm[type].type;

		if (parm[type].cstype == SYBCHAR)
		{
			datafmt[column].maxlength = 255;
	datalen[column] = strlen(address);
		}
		else
		{
			datafmt[column].maxlength = sizeof(CS_INT);
			datalen[column] = CS_UNUSED;
		}

		retcode = blk_bind(blk_desc, column, &datafmt[column],
					address, &datalen[column], NULL);
		RETURN_IF(retcode, "bulk_bind: blk_bind");
		return retcode;
}
```

```
/*void */
CS_RETCODE
bulk_load(db, blk_desc)
		int db;
		CS_BLKDESC		*blk_desc;
{
		CS_RETCODE retcode=CS_SUCCEED;
		CS_INT	numofrows=0;

		count[db]++;
		retcode = blk_rowxfer(blk_desc);
		RETURN_IF(retcode, "bulk_load: blk_rowxfer");

		if (count[db]%1024 == 0  )
		{
			retcode = blk_done(blk_desc, CS_BLK_BATCH, &numofrows);
			if (retcode == CS_FAIL)
			{
				printf("bulk_load: Can't post rows\n");
				RETURN_IF(retcode, "bulk_load:blk_done");
			}
		}

		RETURN_IF(retcode, "bulk_load:blk_done");
		return retcode;
}


void
bulk_close(db, blk_desc)
int db;
CS_BLKDESC *blk_desc;
{
		CS_INT numofrows=0;

		if (blk_done(blk_desc, CS_BLK_ALL, &numofrows) == CS_FAIL)
			printf("Problems completing the bulk copy.\n");

		if (blk_drop(blk_desc) == CS_FAIL)
		{
			printf("blk_drop failed\n");
		}

		dbconn[db] = NULL;
		return;
}


/*****************************CHRIS*****************************************/




/*************************************************************************
We rely on the static location row structures in the 'load' program so
   we only have to bind to the variables once.
*************************************************************************/

void
warehouse_load(int bulk_w, CS_BLKDESC *blkdesc, warehouse_row *r)
{
   static warehouse_row *old = NULL;

   if (old != r) {
      old = r;
		bulk_bind(1, "w_id",   &r->W_ID, ID_T,blkdesc);
		bulk_bind(2, "w_name",     r->W_NAME, TEXT_T, blkdesc);
		bulk_bind(3, "w_street_1", r->W_STREET_1, TEXT_T, blkdesc);
```

```
                bulk_bind(4, "w_street_2", r->W_STREET_2, TEXT_T, blkdesc);
                bulk_bind(5, "w_city",     r->W_CITY, TEXT_T, blkdesc);
                bulk_bind(6, "w_state",    r->W_STATE, TEXT_T, blkdesc);
                bulk_bind(7, "w_zip",      r->W_ZIP, TEXT_T, blkdesc);
                bulk_bind(8, "w_tax",      &r->W_TAX, FLOAT_T, blkdesc);
                bulk_bind(9, "w_ytd",      &r->W_YTD, MONEY_T, blkdesc);
    }
    bulk_bind_adjustlen(2,"w_name",r->W_NAME);
    bulk_bind_adjustlen(3,"w_street_1",r->W_STREET_1);
    bulk_bind_adjustlen(4,"w_street_2",r->W_STREET_2);
    bulk_bind_adjustlen(5,"w_city",r->W_CITY);
    bulk_bind_adjustlen(6,"w_state",r->W_STATE);
    bulk_bind_adjustlen(7,"w_zip",r->W_ZIP);
    bulk_load(bulk_w, blkdesc);
}

void
end_warehouse_load(int b, CS_BLKDESC *d)
{
    bulk_close(b,d);
}


void
district_load(int bulk_d, CS_BLKDESC *blkdesc, district_row *r)
{
    static district_row *old = NULL;

    if (r != old) {
        old = r;

                bulk_bind(1, "d_id", &r->D_ID, ID_T,blkdesc);
                bulk_bind(2, "d_w_id", &r->D_W_ID, ID_T,blkdesc);
                bulk_bind(3, "d_name", r->D_NAME, TEXT_T,blkdesc);
                bulk_bind(4, "d_street_1", r->D_STREET_1, TEXT_T,blkdesc);
                bulk_bind(5, "d_street_2", r->D_STREET_2, TEXT_T,blkdesc);
                bulk_bind(6, "d_city", r->D_CITY, TEXT_T,blkdesc);
                bulk_bind(7, "d_state", r->D_STATE, TEXT_T,blkdesc);
                bulk_bind(8, "d_zip", r->D_ZIP, TEXT_T,blkdesc);
                bulk_bind(9, "d_tax", &r->D_TAX, FLOAT_T,blkdesc);
                bulk_bind(10, "d_ytd", &r->D_YTD, MONEY_T,blkdesc);
                bulk_bind(11, "d_next_o_id", &r->D_NEXT_O_ID, ID_T,blkdesc);
    }
    bulk_bind_adjustlen(3,"d_name", r->D_NAME);
    bulk_bind_adjustlen(4,"d_street_1", r->D_STREET_1);
    bulk_bind_adjustlen(5,"d_street_2", r->D_STREET_2);
    bulk_bind_adjustlen(6,"d_city", r->D_CITY);
    bulk_bind_adjustlen(7,"d_state", r->D_STATE);
    bulk_bind_adjustlen(8,"d_zip", r->D_ZIP);

    bulk_load(bulk_d, blkdesc);
}


void
end_district_load(int b, CS_BLKDESC *d)
{
    bulk_close(b,d);
}

void
item_load(int bulk_i, CS_BLKDESC *blkdesc, item_row *r)
{
    static item_row *old = NULL;

    if (r != old) {
        old = r;
                bulk_bind(1, "i_id", &r->I_ID, ID_T,blkdesc);
                bulk_bind(2, "i_im_id", &r->I_IM_ID, ID_T,blkdesc);
                bulk_bind(3, "i_name", r->I_NAME, TEXT_T,blkdesc);
                bulk_bind(4, "i_price", &r->I_PRICE, MONEY_T,blkdesc);
                bulk_bind(5, "i_data", r->I_DATA, TEXT_T,blkdesc);
    }
    bulk_bind_adjustlen(3, "i_name", r->I_NAME);
    bulk_bind_adjustlen(5, "i_data", r->I_DATA);
    bulk_load(bulk_i, blkdesc);
}


void
end_item_load(int b, CS_BLKDESC *d)
{
    bulk_close(b,d);
}

static DBDATETIME h_date;

void
history_load(int bulk_h, CS_BLKDESC *blkdesc, history_row *r)
{
    static history_row *old = NULL;

    if (r != old) {
        old = r;

                bulk_bind(1, "h_c_id",   &r->H_C_ID,   ID_T,blkdesc);
                bulk_bind(2, "h_c_d_id", &r->H_C_D_ID, ID_T,blkdesc);
                bulk_bind(3, "h_c_w_id", &r->H_C_W_ID, ID_T,blkdesc);
                bulk_bind(4, "h_d_id",   &r->H_D_ID,   ID_T,blkdesc);
                bulk_bind(5, "h_w_id",   &r->H_W_ID,   ID_T,blkdesc);
                bulk_bind(6, "h_date",   &h_date,   DATE_T,blkdesc);
                bulk_bind(7, "h_amount", &r->H_AMOUNT, FLOAT_T,blkdesc);
                bulk_bind(8, "h_data",   r->H_DATA,    TEXT_T,blkdesc);
    }
    bulk_bind_adjustlen(8, "h_data",   r->H_DATA);
    to_sybase_date(&r->H_DATE, &h_date);
    bulk_load(bulk_h, blkdesc);
}


void
end_history_load(int b, CS_BLKDESC *d)
{
    bulk_close(b,d);
}


static char c_data_1[250+1];
static char c_data_2[250+1];

void
customer_load(bulk_c, blkdesc, r)
    int bulk_c;
    CS_BLKDESC *blkdesc;
    customer_row *r;
{
    int len;
    char *p;
    static customer_row *old = NULL;

    /* break up the data string into two pieces */
    len = strlen(r->C_DATA);
    if (len > 250) {
        memcpy(c_data_1, r->C_DATA, 250);
        c_data_1[250] = '\0';
        memcpy(c_data_2, r->C_DATA+250, len-250+1);
    } else {
        memcpy(c_data_1, r->C_DATA, 250+1);
        strcpy(c_data_2, " ");
    }
```

```
    if (r != old) {
        old = r;

                bulk_bind(1, "c_id",                &r->C_ID,   ID_T,blkdesc);
                bulk_bind(2, "c_d_id",   &r->C_D_ID, ID_T,blkdesc);
                bulk_bind(3, "c_w_id",   &r->C_W_ID, ID_T,blkdesc);
                bulk_bind(4, "c_first",   r->C_FIRST, TEXT_T,blkdesc);
                bulk_bind(5, "c_middle",  r->C_MIDDLE, TEXT_T,blkdesc);
                bulk_bind(6, "c_last",    r->C_LAST,   TEXT_T,blkdesc);
                bulk_bind(7, "c_street_1",  r->C_STREET_1 , TEXT_T,blkdesc);
                bulk_bind(8, "c_street_2",  r->C_STREET_2,  TEXT_T,blkdesc);
                bulk_bind(9, "c_city",    r->C_CITY,              TEXT_T,blkdesc);
                bulk_bind(10, "c_state",   r->C_STATE,        TEXT_T,blkdesc);
                bulk_bind(11, "c_zip",              r->C_ZIP,     TEXT_T,blkdesc);
                bulk_bind(12, "c_phone",   r->C_PHONE,     TEXT_T,blkdesc);
                bulk_bind(13, "c_since",   &r->C_SINCE,     DATE_T,blkdesc);
                bulk_bind(14, "c_credit",  r->C_CREDIT,    TEXT_T,blkdesc);
                bulk_bind(15, "c_credit_lim", &r->C_CREDIT_LIM, MONEY_T,blkdesc);
                bulk_bind(16, "c_discount", &r->C_DISCOUNT, FLOAT_T,blkdesc);
                bulk_bind(17, "c_delivery_cnt", &r->C_DELIVERY_CNT, COUNT_T,blkdesc);
                bulk_bind(18, "c_payment_cnt",&r->C_PAYMENT_CNT, COUNT_T,blkdesc);
                bulk_bind(19, "c_balance", &r->C_BALANCE, FLOAT_T,blkdesc);
                bulk_bind(20, "c_ytd_payment", &r->C_YTD_PAYMENT, FLOAT_T,blkdesc);
                bulk_bind(21, "c_data1",  c_data_1, TEXT_T, blkdesc);
                bulk_bind(22, "c_data_2", c_data_2, TEXT_T, blkdesc);
    }
    bulk_bind_adjustlen(4, "c_first",   r->C_FIRST);
    bulk_bind_adjustlen(5, "c_middle",  r->C_MIDDLE);
    bulk_bind_adjustlen(6, "c_last",    r->C_LAST);
    bulk_bind_adjustlen(7, "c_street_1",  r->C_STREET_1);
    bulk_bind_adjustlen(8, "c_street_2",  r->C_STREET_2);
    bulk_bind_adjustlen(9, "c_city",    r->C_CITY);
    bulk_bind_adjustlen(10, "c_state",   r->C_STATE);
    bulk_bind_adjustlen(11, "c_zip",                r->C_ZIP);
    bulk_bind_adjustlen(12, "c_phone",   r->C_PHONE);
    bulk_bind_adjustlen(14, "c_credit",  r->C_CREDIT);
    bulk_bind_adjustlen(21, "c_data1",  c_data_1);
    bulk_bind_adjustlen(22, "c_data_2", c_data_2);

    bulk_load(bulk_c, blkdesc);
}

void
end_customer_load(int bulk_c, CS_BLKDESC *blkdesc)
{
    bulk_close(bulk_c, blkdesc);
}

DBDATETIME o_entry_d;
COUNT o_carrier_id;

void
order_load(int o_bulk, CS_BLKDESC *blkdesc, order_row *r)
{
    static order_row *old = NULL;

    if (r != old) {
        old = r;
                bulk_bind(1, "o_id", &r->O_ID, ID_T,blkdesc);
                bulk_bind(2, "o_c_id", &r->O_C_ID, ID_T,blkdesc);
                bulk_bind(3, "o_d_id", &r->O_D_ID, ID_T,blkdesc);
                bulk_bind(4, "o_w_id", &r->O_W_ID, ID_T,blkdesc);
                bulk_bind(5, "o_entry_d", &o_entry_d, DATE_T,blkdesc);
                bulk_bind(6, "o_carrier_id", &o_carrier_id, ID_T,blkdesc);
                bulk_bind(7, "o_ol_cnt", &r->O_OL_CNT, COUNT_T,blkdesc);
                bulk_bind(8, "o_all_local", &r->O_ALL_LOCAL, LOGICAL_T,blkdesc);
    }
    to_sybase_date(&r->O_ENTRY_D, &o_entry_d);
    if (r->O_CARRIER_ID == EMPTY_NUM)  o_carrier_id = -1;
    else                        o_carrier_id = r->O_CARRIER_ID;

    bulk_load(o_bulk, blkdesc);
}

void
end_order_load(int o_bulk, CS_BLKDESC *blkdesc)
{
    bulk_close(o_bulk, blkdesc);
}

DBDATETIME ol_delivery_d;
void
orderline_load(int ol_bulk, CS_BLKDESC *ol_blkdesc, orderline_row *r)
{
    static orderline_row *old;

    if (r != old) {
        old = r;
        bulk_bind(1, "ol_o_id", &r->OL_O_ID, ID_T, ol_blkdesc);
                bulk_bind(2, "ol_d_id", &r->OL_D_ID, ID_T, ol_blkdesc);
                bulk_bind(3, "ol_w_id", &r->OL_W_ID, ID_T, ol_blkdesc);
                bulk_bind(4, "ol_number", &r->OL_NUMBER, ID_T, ol_blkdesc);
                bulk_bind(5, "ol_i_id", &r->OL_I_ID, ID_T, ol_blkdesc);
                bulk_bind(6, "ol_supply_w_id", &r->OL_SUPPLY_W_ID,ID_T, ol_blkdesc);
                bulk_bind(7, "ol_delivery_d", &ol_delivery_d, DATE_T, ol_blkdesc);
                bulk_bind(8, "ol_quantity", &r->OL_QUANTITY, COUNT_T, ol_blkdesc);
                bulk_bind(9, "ol_amount", &r->OL_AMOUNT, MONEY_T, ol_blkdesc);
                bulk_bind(10, "ol_dist_info", r->OL_DIST_INFO, TEXT_T, ol_blkdesc);
    }
    bulk_bind_adjustlen(10, "ol_dist_info", r->OL_DIST_INFO);
    to_sybase_date(&r->OL_DELIVERY_D, &ol_delivery_d);
    bulk_load(ol_bulk, ol_blkdesc);
}


void
end_orderline_load(int ol_bulk, CS_BLKDESC *ol_blkdesc)
{
    bulk_close(ol_bulk, ol_blkdesc);
}


void
neworder_load(int no_bulk, CS_BLKDESC *blkdesc, neworder_row *r)
{
    static neworder_row *old;

    if (r != old) {
        old = r;
        bulk_bind(1, "no_o_id", &r->NO_O_ID, ID_T, blkdesc);
        bulk_bind(2, "no_d_id", &r->NO_D_ID, ID_T, blkdesc);
        bulk_bind(3, "no_w_id", &r->NO_W_ID, ID_T, blkdesc);
    }
    bulk_load(no_bulk, blkdesc);
}

void
end_neworder_load(int bulk_no, CS_BLKDESC *blkdesc)
{
    bulk_close(bulk_no, blkdesc);
}

void
stock_load(int bulk_s, CS_BLKDESC *blkdesc, stock_row *r)
{
    static stock_row *old;

    if (r != old) {
        old = r;
```

```
        bulk_bind(1, "s_i_id", &r->S_I_ID, ID_T,blkdesc);
        bulk_bind(2, "s_w_id", &r->S_W_ID, ID_T,blkdesc);
        bulk_bind(3, "s_quantity", &r->S_QUANTITY, COUNT_T,blkdesc);
        bulk_bind(4, "s_ytd",      &r->S_YTD, COUNT_T,blkdesc);
        bulk_bind(5, "s_order_cnt", &r->S_ORDER_CNT,COUNT_T,blkdesc);
        bulk_bind(6, "s_remote_cnt", &r->S_REMOTE_CNT,COUNT_T,blkdesc);
        bulk_bind(7, "s_dist_01", r->S_DIST_01, TEXT_T,blkdesc);
        bulk_bind(8, "s_dist_02", r->S_DIST_02, TEXT_T,blkdesc);
        bulk_bind(9, "s_dist_03", r->S_DIST_03, TEXT_T,blkdesc);
        bulk_bind(10, "s_dist_04", r->S_DIST_04, TEXT_T,blkdesc);
        bulk_bind(11, "s_dist_05", r->S_DIST_05, TEXT_T,blkdesc);
        bulk_bind(12, "s_dist_06", r->S_DIST_06, TEXT_T,blkdesc);
        bulk_bind(13, "s_dist_07", r->S_DIST_07, TEXT_T,blkdesc);
        bulk_bind(14, "s_dist_08", r->S_DIST_08, TEXT_T,blkdesc);
        bulk_bind(15, "s_dist_09", r->S_DIST_09, TEXT_T,blkdesc);
        bulk_bind(16, "s_dist_10", r->S_DIST_10, TEXT_T,blkdesc);
        bulk_bind(17, "s_data",    r->S_DATA,   TEXT_T,blkdesc);
    }
    bulk_bind_adjustlen(7, "s_dist_01", r->S_DIST_01);
    bulk_bind_adjustlen(8, "s_dist_02", r->S_DIST_02);
    bulk_bind_adjustlen(9, "s_dist_03", r->S_DIST_03);
    bulk_bind_adjustlen(10, "s_dist_04", r->S_DIST_04);
    bulk_bind_adjustlen(11, "s_dist_05", r->S_DIST_05);
    bulk_bind_adjustlen(12, "s_dist_06", r->S_DIST_06);
    bulk_bind_adjustlen(13, "s_dist_07", r->S_DIST_07);
    bulk_bind_adjustlen(14, "s_dist_08", r->S_DIST_08);
    bulk_bind_adjustlen(15, "s_dist_09", r->S_DIST_09);
    bulk_bind_adjustlen(16, "s_dist_10", r->S_DIST_10);
    bulk_bind_adjustlen(17, "s_data",    r->S_DATA);
    bulk_load(bulk_s, blkdesc);
    return;
}

void
end_stock_load(int bulk_s, CS_BLKDESC *blkdesc)
{
    bulk_close(bulk_s, blkdesc);
}

void
to_sybase_date(DATE *date, DBDATETIME *sybdate)
{
    sybdate->dtdays = date->day;
    sybdate->dttime = date->sec*300;
}

CS_INT  cl_err_handler(context, connection, errmsg)
CS_CONTEXT      *context;
CS_CONNECTION   *connection;
CS_CLIENTMSG    *errmsg;

{
                fprintf(stderr, "Open Client Error (%d, %d, %d, %d)\n",
                                        CS_LAYER(errmsg->msgnumber),
                                        CS_ORIGIN(errmsg->msgnumber),
                                        CS_SEVERITY(errmsg->msgnumber),
                                        CS_NUMBER(errmsg->msgnumber));

                fprintf(stderr, "%s\n", errmsg->msgstring);

                if (errmsg->osstringlen > 0)
                {
                        fprintf(stderr, "Operating System Error:\n%s\n",
                                        errmsg->osstring);
                }

                return (CS_SUCCEED);
}   /* end of cl_err_handler () */

CS_INT server_msg_handler(context, connection, srvmsg)
CS_CONTEXT      *context;
CS_CONNECTION   *connection;
CS_SERVERMSG    *srvmsg;

{
                if (srvmsg->msgnumber != 5701)
                {
                        fprintf(stderr, "Server message %ld, Severity %ld, State %ld\n",
                                        srvmsg->msgnumber, srvmsg->severity, srvmsg->state);

                        if (srvmsg->svrnlen > 0)
                        {
                                fprintf(stderr, "Server '%s'", srvmsg->svrname);
                        }

                        if (srvmsg->proclen > 0)
                        {
                                fprintf(stderr, "Procedure '%s'", srvmsg->proc);
                        }

                        if (srvmsg->line > 0)
                        {
                                fprintf(stderr, "Line '%d'", srvmsg->line);
                        }

                        fprintf(stderr, "\n %s\n", srvmsg->text);
                }

                return(CS_SUCCEED);
}  /* end of server_msg_handler()  */

CS_INT cs_err_handler(context, errmsg)
CS_CONTEXT      *context;
CS_CLIENTMSG    *errmsg;
{
                fprintf(stderr, "CS Lib Error %d:\n%s\n",
                                        errmsg->msgnumber, errmsg->msgstring);

                return (CS_SUCCEED);
}  /* end of cs_err_handler() */

/* various bind routines */
#define bind_ID bind_int
#define bind_COUNT bind_int
#define bind_LOGICAL bind_int
#define bind_MONEY bind_double
#define bind_REAL bind_double
#define bind_TEXT bind_str

#define load_row(table) \
{ \
    (table)->count++; \
    if (bcp_sendrow((table)->dbproc) != SUCCEED) \
        error("load_row: Can't load row in %s table\n", (table)->name); \
    if (table->count%BATCH == 0 && (bcp_batch(table->dbproc) == -1)) \
        error("load_row: Can't post rows in %s table\n", (table)->name); \
}

typedef struct
    {
    int count;
    char name[40];
    DBPROCESS *dbproc;
    } table_info;

table_info w_table[1];

begin_warehouse_serial_load()
    {
    table_open(w_table, "warehouse");
```

```
    }


warehouse_serial_load(r)
    warehouse_row *r;
    {
    static warehouse_row *old = NULL;

    if (old != r)
        {
        old = r;
        bind_ID  (w_table, 1, &r->W_ID);
        bind_TEXT(w_table, 2, r->W_NAME);
        bind_TEXT(w_table, 3, r->W_STREET_1);
        bind_TEXT(w_table, 4, r->W_STREET_2);
        bind_TEXT(w_table, 5, r->W_CITY);
        bind_TEXT(w_table, 6, r->W_STATE);
        bind_TEXT(w_table, 7, r->W_ZIP);
        bind_REAL(w_table, 8, &r->W_TAX);
        bind_MONEY(w_table, 9, &r->W_YTD);
        }

    load_row(w_table);
    }


end_warehouse_serial_load()
    {
    table_close(w_table);
    }



table_info d_table[1];

begin_district_serial_load()
    {
    table_open(d_table, "district");
    }

district_serial_load(r)
    district_row *r;
    {
    static district_row *old = NULL;

    if (r != old)
        {
        old = r;
        bind_ID(d_table, 1, &r->D_ID);
        bind_ID(d_table, 2, &r->D_W_ID);
        bind_TEXT(d_table, 3, r->D_NAME);
        bind_TEXT(d_table, 4, r->D_STREET_1);
        bind_TEXT(d_table, 5, r->D_STREET_2);
        bind_TEXT(d_table, 6, r->D_CITY);
        bind_TEXT(d_table, 7, r->D_STATE);
        bind_TEXT(d_table, 8, r->D_ZIP);
        bind_REAL(d_table, 9, &r->D_TAX);
        bind_MONEY(d_table, 10, &r->D_YTD);
        bind_ID(d_table, 11, &r->D_NEXT_O_ID);
        }

    load_row(d_table);
    }


end_district_serial_load()
    {
    table_close(d_table);
    }



table_info i_table[1];

begin_item_serial_load()
    {
    table_open(i_table, "item");
    }

item_serial_load(r)
    item_row *r;
    {
    static item_row *old = NULL;

    if (r != old)
        {
        old = r;
        bind_ID(i_table, 1, &r->I_ID);
        bind_ID(i_table, 2, &r->I_IM_ID);
        bind_TEXT(i_table, 3, r->I_NAME);
        bind_MONEY(i_table, 4, &r->I_PRICE);
        bind_TEXT(i_table, 5, r->I_DATA);
        }
    load_row(i_table);
    }


end_item_serial_load()
    {
    table_close(i_table);
    }



table_info h_table[1];

static DBDATETIME h_date;

begin_history_serial_load()
    {
    table_open(h_table, "history");
    }

history_serial_load(r)
    history_row *r;
    {
    static history_row *old = NULL;

    if (r != old)
        {
        old = r;
        bind_ID(h_table, 1, &r->H_C_ID);
        bind_ID(h_table, 2, &r->H_C_D_ID);
        bind_ID(h_table, 3, &r->H_C_W_ID);
        bind_ID(h_table, 4, &r->H_D_ID);
        bind_ID(h_table, 5, &r->H_W_ID);
        bind_SYBDATE(h_table, 6, &h_date);
        bind_MONEY(h_table, 7, &r->H_AMOUNT);
        bind_TEXT(h_table, 8, r->H_DATA);
        }

    to_sybase_date(&r->H_DATE, &h_date);
    load_row(h_table);
    }


end_history_serial_load()
    {
```

```
    table_close(h_table);
    }

LOGINREC *login;

table_open(table, table_name)
    table_info *table;
    char *table_name;
    {
    int message_handler(), error_handler();

    /* save the table name and clear the row counte */
    strncpy(table->name, table_name, sizeof(table->name));
    table->count = 0;

    /* make note we have established a connection */
    DBSETLUSER(login, "sa");
    DBSETLAPP(login, table_name);
    BCP_SETL(login, TRUE);
    DBSETLPACKET(login,4096);

    /* establish a connection with the server specified by DSQUERY env var */
    table->dbproc = dbopen(login, NULL);
    if (table->dbproc == NULL)
        error("Can't establish connection. Is the DSQUERY environment set?\n");

    /* select the database to use */
    if (dbuse(table->dbproc, "tpcc") != SUCCEED)
        error("Can't select database: TPCC\n");

    /* prepare to do a bulk copy */
    if (bcp_init(table->dbproc, table->name, NULL, NULL, DB_IN) != SUCCEED)
      error("Can't initialize the bulk copy to table %s\n", table);

    }

table_close(table)
    table_info *table;
    {
    if (bcp_done(table->dbproc) == -1)
        error("Problems completing the bulk copy in %s table.\n", table->name);
    dbclose(table->dbproc);
    }

bind_int(table, column, addr)
    table_info *table;
    int column;
    int *addr;
    {

    if (bcp_bind(table->dbproc, (BYTE *)addr, 0, -1, NULL, 0, SYBINT4, column)
            != SUCCEED)
        error("Can't bind INT to col %d in %s table\n", column, table->name);
    }

bind_double(table, column, addr)
    table_info *table;
    int column;
    double *addr;
    {
    if (bcp_bind(table->dbproc, (BYTE *)addr, 0, -1, NULL, 0, SYBFLT8, column)
          != SUCCEED)
        error("Can't bind DOUBLE to col %d in %s table\n", column,table->name);
    }

bind_str(table, column, addr)
    table_info *table;
    int column;
    char *addr;
    {
    if (bcp_bind(table->dbproc, (BYTE *)addr, 0, -1, (BYTE *)"", 1, SYBCHAR, column) != SUCCEED)
        error("Can't bind STRING to col %d in %s table/n", column,table->name);
    }

bind_SYBDATE(table, column, addr)
    table_info *table;
    int column;
    DBDATETIME *addr;
    {
    if (bcp_bind(table->dbproc, (BYTE *)addr, 0, -1, NULL, 0, SYBDATETIME, column)
            != SUCCEED)
        error("Can't bind SYBDATE to col %d in %s table\n",column,table->name);
    }


error_handler(dbproc, msgno, msgstate, severity, msgtext, srvname,
            procname, line)
/*****************************************************************************
error_handler deals with error messages
*****************************************************************************/
    DBPROCESS *dbproc;
    DBINT     msgno;
    int       msgstate;
    int       severity;
    char      *msgtext;
    char      *procname;
    DBUSMALLINT line;
    {
    message("Error %d at line %d: %s\n", msgno, line, msgtext);
    return INT_CANCEL;
    }

message_handler(dbproc, msgno, msgstate, severity, msgtext, srvname,
            procname, line)
/*****************************************************************************
message_handler deals with informational messges
*****************************************************************************/
    DBPROCESS *dbproc;
    DBINT     msgno;
    int       msgstate;
    int       severity;
    char      *msgtext;
    char      *procname;
    DBUSMALLINT line;
    {
    if (severity != 0)
        message("Message from DBLIB: %s\n", msgtext);
    }


begin_load()
    {

    /* start the row counter */
    /* use dblib version 10 for numeric datatypes */
    dbsetversion(DBVERSION_100);

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

    /* install a message handler */
    (void)dbmsghandle(message_handler);
    (void)dberrhandle(error_handler);

    /* allocate a login record and fill it in */
    login = dblogin();
    if (login == NULL)
        error("Can't allocate a login record.\n");
    }
```

Makefile

```
#*******************************************************************************
#@(#) Version: A.10.10 $Date: 2001/08/27 13:17:06 $
#
#(c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
#*******************************************************************************

#
# TPC-C load generation program
#

#
# compile options
#
OPTIMIZE= +O2 -z +Olibcalls +Ofastaccess +Oentrysched +Onolimit +Oprocelim -Wl,+pd,4M -Wl,+pi,1M

L=${WORK_DIR}/src/lib
CFLAGS=${OPTIMIZE} -I$L -I${SYBASE_INC}
LDFLAGS=${OPTIMIZE} -Wl,-E -Wl,-a,archive_shared
LIBS=$L/tpc_lib.a -L ${SYBASE_LIB} -lblk -lct -lcs -ltcl -lcomn -lintl -lsybdb -lpthread -lm -ldld

all: load

load: load.o  bulk_sybase.o
        cc ${LDFLAGS} load.o bulk_sybase.o ${LIBS} -o $@

install: load
        mv load ${WORK_DIR}/bin/sybase/

clean:
        rm -f *.o
        rm -f *.a

clobber: clean
        rm -f load
```

Appendix C Tunable Parameters

The HP-UX operating system tunable parameters employed to generate the kernel for the hp server rp8400 and the 5 hp workstation c3700 and 3 hp workstation j6000. All clients used the same configuration files as Client1's shown here. Included as well are the Sybase Adaptive Server Enterprise v12.5 and BEA TUXEDO 6.4 parameters.

C.1 HP-UX Configuration - Clients

Config/Client1/ostune.ver

```
****************************************
* $Source: /usr/local/kcs/sys.ROSE_800/filesets.info/CORE-KRN/RCS/generic,v $
* $Revision: 1.3.106.2 $        $Author: kcs $
* $State: Exp $                 $Locker: CRT $
* $Date: 97/07/12 21:51:58 $
*
****************************************
* Additional drivers required in every machine-type to create a complete
* system file during cold install.  This list is every driver that the
* master.d/ files do not force on the system or is not identifiable by
* ioscan.
* Other CPU-type specific files can exist for their special cases.
* see create_sysfile (1m).
****************************************
*
* Drivers/Subsystems
sba
lba
*btlan6
*btlan3
audio
side
sdisk
sctl
superio
asio0
SCentIf
hcd
hub
c720
graph3
cdfs
nfs_core
*nfs_client
*nfs_server
dlpi
inet
uipc
tun
telm
tels
netdiag1
nms
*fcgsc_lan
*fc_arp
*fcT1_fcp
*fcpmux
vxbase
lvm
lv
hpstreams
clone
strlog
sad
echo
sc
timod
tirdwr
pipedev
pipemod
ffs
ldterm
ptem
pts
ptm
pckt
diag0
diag2
dmem
dev_config
*cifs
sapic
diag1
usbd
hid
beep
maclan
token_arp
autofsc
*nfsm
rpcmod
stape
btlan

dump lvol

STRMSGSZ        65535
bufpages        8192
dnlc_hash_locks   512
create_fastlinks    1
dbc_min_pct         0
dbc_max_pct         0
default_disk_ir     1
fs_async            1
maxfiles        2048
maxfiles_lim    2048
maxdsiz         0x80000000
maxssiz         0X10000000
maxswapchunks   16384
maxuprc         16000
nproc           (300+MAXUPRC)

msgmni          (NPROC)
msgtql          (NPROC)
msgseg          (MSGMNI*2)
msgssz          512
msgmap          (MSGSEG)
msgmax          32768
msgmnb          (MSGMAX*2)

nfile           (NPROC*5)
ninode          (NPROC*5)
nflocks         4000
npty            128
nstrpty         200
```

```
semmni          32
semmns          NPROC
semmnu          (SEMMNS)
semume          4
semvmx          40960

shmmax          0X40000000
shmmni          16
shmseg          16

swapmem_on      0
unlockable_mem  1
```

C.2 HP-UX Configurtion – Server

Config/Server/ostune.ver

```
*****************************************
* Source: /ux/core/kern/filesets.info/CORE-KRN/generic
* @(#)B.11.11_LR
*
*****************************************
* Additional drivers required in every machine-type to create a complete
* system file during cold install.  This list is every driver that the
* master.d/ files do not force on the system or is not identifiable by
* ioscan.
* Other CPU-type specific files can exist for their special cases.
* see create_sysfile (1m).
*****************************************
*
* Drivers/Subsystems
sba
lba
asio0
c720
sdisk
sctl
cdfs
olar_psm
olar_psm_if
dev_olar
hd_fabric
diag0
diag1
diag2
dmem
dev_config
iomem
nfs_core
btlan
maclan
dlpi
token_arp
inet
uipc
tun
telm
tels
netdiag1
nms
hpstreams
clone
strlog
sad
echo
sc
timod
tirdwr
pipedev
pipemod
ffs
ldterm
ptem
pts
ptm
pckt
td
gelan
GSCtoPCI
lvm
lv
rpcmod
asyncdsk

dump lvol

STRMSGSZ    65535
nstrpty                 60
bufpages        8192
dbc_max_pct     0
dbc_min_pct     0
maxfiles        2048
maxfiles_lim    2048
maxswapchunks   9975
swchunk         8192
maxuprc         500
maxusers        16
maxvgs          20
nfile           (15*NPROC+1024)
nflocks         (NPROC)
ninode          (8*NPROC+2048)
nproc           600
shmmax          68719476736
shmmni          128
shmseg          16
swapmem_on      0
unlockable_mem  1
```

C.3 Sybase Adaptive Server Enterprise v12.5 Parameters

Config/Server/dbtune.ver

```
##########################################################################
#
#                       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 = 32767
        print recovery information = DEFAULT
        tape retention in days = DEFAULT

[Cache Manager]
        number of oam trips = DEFAULT
        number of index trips = DEFAULT
        memory alignment boundary = DEFAULT
        global async prefetch limit = 90
        global cache partition number = 16

[Named Cache:c_cust_index]
        cache size = 700M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 16

[4K I/O Buffer Pool]
        pool size = 700M
        wash size = 80K
        local async prefetch limit = 0

[Named Cache:c_cust_non_index]
        cache size = 3100M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT
        local cache partition number = 16

[4K I/O Buffer Pool]
        pool size = 3100M
        wash size = 512 K
        local async prefetch limit = 0

[Named Cache:c_customer]
        cache size = 16M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 16

[4K I/O Buffer Pool]
        pool size = 16M
        wash size =  1M
        local async prefetch limit = 0

[Named Cache:c_log]
        cache size = 20M
        cache status = log only
        cache replacement policy = DEFAULT
        local cache partition number = 1

[4K I/O Buffer Pool]
        pool size =  20M
        wash size =   8M
        local async prefetch limit = 0

[Named Cache:c_no]
        cache size = 1500M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 1500M
        wash size =   2M
        local async prefetch limit = 0

[Named Cache:c_no_order_index]
        cache size = 80M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 80M
        wash size = 80 K
        local async prefetch limit = 0

[Named Cache:c_ol]
        cache size = 2000M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 16

[4K I/O Buffer Pool]
        pool size = 2000M
        wash size =   2M
        local async prefetch limit = 0

[Named Cache:c_ol_index]
        cache size = 800M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 16

[4K I/O Buffer Pool]
        pool size = 800M
        wash size =   1M
        local async prefetch limit = 0

[Named Cache:c_orders]
        cache size = 2400M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 16

[4K I/O Buffer Pool]
        pool size = 1440M
        wash size =   8M
        local async prefetch limit = 0

[16K I/O Buffer Pool]
        pool size = 960M
        wash size =   4M
        local async prefetch limit = 0

[Named Cache:c_stock]
        cache size = 46000M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 32

[4K I/O Buffer Pool]
        pool size = 36000M
        wash size =   36M
        local async prefetch limit = 0

[Named Cache:c_stock_index]
        cache size   = 990M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 32

[4K I/O Buffer Pool]
        pool size = 990M
```

```
        wash size =  80K
        local async prefetch limit = 0

[Named Cache:c_wid]
        cache size = 160M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 32

[4K I/O Buffer Pool]
        pool size = 160M
        wash size = 256 K
        local async prefetch limit = 0

[Named Cache:default data cache]
        cache size = 40M
        cache status = default data cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 8

[4K I/O Buffer Pool]
        pool size = 40M
        wash size = 512 K
        local async prefetch limit = 0


[Meta-Data Caches]
        number of open databases = 5
        number of open objects = DEFAULT
        open object spinlock ratio = DEFAULT
        number of open indexes = DEFAULT
        open index hash spinlock ratio = DEFAULT
        open index spinlock ratio = DEFAULT
        partition groups = DEFAULT
        partition spinlock ratio = DEFAULT

[Disk I/O]
        disk i/o structures = 16384
        number of large i/o buffers = DEFAULT
        page utilization percent = DEFAULT
        number of devices = 128
        disable disk mirroring = 1
        allow sql server async i/o = DEFAULT

[Languages]
        disable character set conversions = DEFAULT

[Unicode]
        enable unicode normalization = DEFAULT
        enable surrogate processing = DEFAULT
        enable unicode conversions = DEFAULT
        size of unilib cache = DEFAULT

[Network Communication]
        default network packet size = DEFAULT
        max network packet size = 4096
        remote server pre-read packets = DEFAULT
        number of remote connections = 600
        number of remote logins = DEFAULT
        number of remote sites = DEFAULT
        max number network listeners = DEFAULT
        tcp no delay = DEFAULT
        allow sendmsg = DEFAULT
        syb_sendmsg port number = DEFAULT
        allow remote access = DEFAULT

[O/S Resources]
        max async i/os per engine = 4096
        max async i/os per server = 4096

[Parallel Query]
        number of worker processes = DEFAULT
        memory per worker process = DEFAULT
        max parallel degree = DEFAULT
        max scan parallel degree = DEFAULT

[Physical Resources]

[Physical Memory]
        max memory = 31457280
        additional network memory = 4915200
        shared memory starting address = DEFAULT
        allocate max shared memory = DEFAULT
        dynamic allocation on demand = DEFAULT
        lock shared memory = DEFAULT
        heap memory per user = DEFAULT

[Processors]
        max online engines = 16
        number of engines at startup = 16

[SQL Server Administration]
        procedure cache size = 128000
        default database size = DEFAULT
        identity burning set factor = DEFAULT
        allow nested triggers = DEFAULT
        allow updates to system tables = DEFAULT
        default fill factor percent = DEFAULT
        default exp_row_size percent = DEFAULT
        number of mailboxes = DEFAULT
        number of messages = DEFAULT
        number of alarms = DEFAULT
        number of pre-allocated extents = DEFAULT
        event buffers per engine = DEFAULT
        cpu accounting flush interval = 2147483647
        i/o accounting flush interval = 2147483647
        sql server clock tick length = DEFAULT
        runnable process search count = DEFAULT
        i/o polling process count = DEFAULT
        time slice = DEFAULT
        cpu grace time = DEFAULT
        number of sort buffers = DEFAULT
        size of auto identity column = DEFAULT
        identity grab size = DEFAULT
        housekeeper free write percent = 0
        enable housekeeper GC = 0
        allow resource limits = DEFAULT
        number of aux scan descriptors = DEFAULT
        SQL Perfmon Integration = DEFAULT
        allow backward scans = DEFAULT
        license information = DEFAULT
        enable sort-merge join and JTC = DEFAULT
        abstract plan load = DEFAULT
        abstract plan dump = DEFAULT
        abstract plan replace = DEFAULT
        abstract plan cache = DEFAULT
        text prefetch size = DEFAULT
        enable HA = DEFAULT

[User Environment]
        number of user connections = 600
        stack size = DEFAULT
        stack guard size = DEFAULT
        permission cache entries = DEFAULT
        user log cache size = 4096
        user log cache spinlock ratio = DEFAULT

[Lock Manager]
        number of locks = 20000
```

```
        deadlock checking period = 3000
        lock spinlock ratio = 10
        lock address spinlock ratio = 1
        lock table spinlock ratio = 1
        lock hashtable size = DEFAULT
        lock scheme = DEFAULT
        lock wait period = DEFAULT
        read committed with lock = DEFAULT
        print deadlock information = DEFAULT
        deadlock retries = DEFAULT
        page lock promotion HWM = DEFAULT
        page lock promotion LWM = DEFAULT
        page lock promotion PCT = DEFAULT
        row lock promotion HWM = DEFAULT
        row lock promotion LWM = DEFAULT
        row lock promotion PCT = DEFAULT

[Security Related]
        systemwide password expiration = DEFAULT
        audit queue size = DEFAULT
        curread change w/ open cursors = DEFAULT
        allow procedure grouping = DEFAULT
        select on syscomments.text = DEFAULT
        auditing = DEFAULT
        current audit table = DEFAULT
        suspend audit when device full = DEFAULT
        enable row level access = DEFAULT
        check password for digit = DEFAULT
        minimum password length = DEFAULT
        maximum failed logins = DEFAULT
        enable ssl = DEFAULT
        unified login required = DEFAULT
        use security services = DEFAULT
        msg confidentiality reqd = DEFAULT
        msg integrity reqd = DEFAULT
        secure default login = DEFAULT

[Extended Stored Procedure]
        esp unload dll = DEFAULT
        esp execution priority = DEFAULT
        esp execution stacksize = DEFAULT
        xp_cmdshell context = DEFAULT
        start mail session = DEFAULT

[Error Log]
        event logging = DEFAULT
        log audit logon success = DEFAULT
        log audit logon failure = DEFAULT
        event log computer name = DEFAULT

[Rep Agent Thread Administration]
        enable rep agent threads = 0

[Component Integration Services]
        enable cis = 0
        cis connect timeout = DEFAULT
        cis bulk insert batch size = DEFAULT
        max cis remote connections = DEFAULT
        cis packet size = DEFAULT
        cis cursor rows = DEFAULT
        enable file access = DEFAULT
        cis bulk insert array size = DEFAULT
        enable full-text search = DEFAULT
        cis rpc handling = DEFAULT

[Java Services]
        enable java = 0
        size of process object heap = DEFAULT
        size of shared class heap = DEFAULT
        size of global fixed heap = DEFAULT
        number of java sockets = DEFAULT
        enable enterprise java beans = DEFAULT

[DTM Administration]
        enable DTM = 0
        enable xact coordination = 0
        xact coordination interval = DEFAULT
        number of dtx participants = DEFAULT
        strict dtm enforcement = DEFAULT
        txn to pss ratio = DEFAULT
        dtm lock timeout period = DEFAULT
        dtm detach timeout period = DEFAULT

[Diagnostics]
        dump on conditions = DEFAULT
        maximum dump conditions = DEFAULT
        number of ccbs = DEFAULT
        caps per ccb = DEFAULT
        average cap size = DEFAULT

[Monitoring]
        enable monitoring = DEFAULT
        sql text pipe active = DEFAULT
        sql text pipe max messages = DEFAULT
        plan text pipe active = DEFAULT
        plan text pipe max messages = DEFAULT
        statement pipe active = DEFAULT
        statement pipe max messages = DEFAULT
        errorlog pipe active = DEFAULT
        errorlog pipe max messages = DEFAULT
        deadlock pipe active = DEFAULT
        deadlock pipe max messages = DEFAULT
        wait event timing = DEFAULT
        process wait events = DEFAULT
        object lockwait timing = DEFAULT
        SQL batch capture = DEFAULT
        statement statistics active = DEFAULT
        per object statistics active = DEFAULT
        max SQL text monitored = DEFAULT
```

C.4 Tuxedo UBBconfig

Config/Client1/tmcfg.ver

```
# This is a UBBconfig for a client1-server configuration.
#
# This UBBconfig requires settings for:
# SERVER_NAME CLIENT_NAME MASTER_NAME SERVER_ADDR CLIENT_ADDR NODE_NAMES
# TLISTEN_PORT TBRIDGE_PORT
# In addition, it requires setting the things all UBBconfig.gens need:
#               IPCKEY                  some decent IPCKEY, should be different for each config
#               ROOTDIR
#               TUXCONFIG
#               APPDIR
#               ULOGDIR
#

#------------------------------------------------------------------
*RESOURCES
#------------------------------------------------------------------
                IPCKEY          40001
                PERM            0666
                MASTER          client1

  MAXACCESSERS          15250           # 1024 or more
```

```
   MAXGTT    1024
   MAXSERVERS              65
   MAXSERVICES             310   # MAXSERVERS * #-of-services-each-server + 10( for BBL)
   MODEL     SHM
   LDBAL     Y

# During benchmark, don't want to scan too often.  In particular, while
# the client1s are stabilizing in virtual memory, we don't want to sanity
# scan; and if we do sanity scan, we want large timeouts, since the BRIDGE
# the BBL, the DBBL, and the client1s aren't getting much CPU time during that
# period.  Current settings:
#                 *  scan servers every 5 minutes (maximum allowed by TUXEDO);
#                 *  wait 1 minute for sanity  responses (maximum allowed by TUXEDO);
#                 *  scan all the BBLs from DBBL every 30 minutes (want one scan in the
#                    audited results);
#                 *  timeout a blocking call after 5 minutes (the maximum).
                  SCANUNIT    60
                  SANITYSCAN                5
                  DBBLWAIT    1
                  BBLQUERY    30
                  BLOCKTIME 5
#
#--------------------------------------------------------------------
*MACHINES
#--------------------------------------------------------------------
DEFAULT:
            TUXCONFIG="/project/iti/confs/TUXconfig.client1"
                              ROOTDIR="/project/iti"
                              APPDIR="/project/tpcc/bin"
                              ULOGPFX="/tmp/TUXEDO_LOG"

# for debugging, put both into the same log on the same machine
#                             ULOGPFX="/home/iti/confs/tpcc/ULOG"
# but for a big run, need some space, and want them local to the
# machine rather than across the net.

# Leave TUXCONFIG alone on the MASTER machine; over-ride for each
# other machine?
client1           LMID=client1
      TUXCONFIG="/project/iti/confs/TUXconfig.client1"
#--------------------------------------------------------------------
*GROUPS
#--------------------------------------------------------------------
group1            LMID=client1
                  GRPNO=1
group2            LMID=client1
                  GRPNO=2
group3            LMID=client1
                  GRPNO=3
group4            LMID=client1
                  GRPNO=4
group5            LMID=client1
                  GRPNO=5
group6            LMID=client1
                  GRPNO=6
group7            LMID=client1
                  GRPNO=7
group8            LMID=client1
                  GRPNO=8
group9            LMID=client1
                  GRPNO=9
group10             LMID=client1
                  GRPNO=10
group11             LMID=client1
                  GRPNO=11
group12             LMID=client1
                  GRPNO=12

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

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

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

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

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

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

service SRVGRP=group1
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n5"
      RQADDR=tpcc_5 SRVID=5

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n6"
      RQADDR=tpcc_6 SRVID=6

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n7"
      RQADDR=tpcc_7 SRVID=7

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n8"
      RQADDR=tpcc_8 SRVID=8

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n9"
      RQADDR=tpcc_9 SRVID=9

service SRVGRP=group2
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n10"
      RQADDR=tpcc_10 SRVID=10

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n11"
      RQADDR=tpcc_11 SRVID=11

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n12"
      RQADDR=tpcc_12 SRVID=12

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n13"
      RQADDR=tpcc_13 SRVID=13

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n14"
      RQADDR=tpcc_14 SRVID=14

service SRVGRP=group3
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n15"
      RQADDR=tpcc_15 SRVID=15

service SRVGRP=group4
                  CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n16"
```

```
        RQADDR=tpcc_16 SRVID=16

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n17"
        RQADDR=tpcc_17 SRVID=17

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n18"
        RQADDR=tpcc_18 SRVID=18

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n19"
        RQADDR=tpcc_19 SRVID=19

service SRVGRP=group4
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n20"
        RQADDR=tpcc_20 SRVID=20

service SRVGRP=group5
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n21"
        RQADDR=tpcc_21 SRVID=21

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

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

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

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

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

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

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

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

service SRVGRP=group6
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n30"
        RQADDR=tpcc_30 SRVID=30

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n31"
        RQADDR=tpcc_31 SRVID=31

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n32"
        RQADDR=tpcc_32 SRVID=32

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n33"
        RQADDR=tpcc_33 SRVID=33

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n34"
        RQADDR=tpcc_34 SRVID=34

service SRVGRP=group7
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n35"
        RQADDR=tpcc_35 SRVID=35

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n36"
        RQADDR=tpcc_36 SRVID=36

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n37"
        RQADDR=tpcc_37 SRVID=37

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n38"
        RQADDR=tpcc_38 SRVID=38

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n39"
        RQADDR=tpcc_39 SRVID=39

service SRVGRP=group8
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n40"
        RQADDR=tpcc_40 SRVID=40

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n41"
        RQADDR=tpcc_41 SRVID=41

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n42"
        RQADDR=tpcc_42 SRVID=42

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n43"
        RQADDR=tpcc_43 SRVID=43

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n44"
        RQADDR=tpcc_44 SRVID=44

service SRVGRP=group9
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n45"
        RQADDR=tpcc_45 SRVID=45

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n46"
        RQADDR=tpcc_46 SRVID=46

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n47"
        RQADDR=tpcc_47 SRVID=47

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n48"
        RQADDR=tpcc_48 SRVID=48

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n49"
        RQADDR=tpcc_49 SRVID=49

service SRVGRP=group10
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n50"
        RQADDR=tpcc_50 SRVID=50

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n51"
```

```
        RQADDR=tpcc_51 SRVID=51

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n52"
        RQADDR=tpcc_52 SRVID=52

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n53"
        RQADDR=tpcc_53 SRVID=53

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n54"
        RQADDR=tpcc_54 SRVID=54

service SRVGRP=group11
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n55"
        RQADDR=tpcc_55 SRVID=55

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n56"
        RQADDR=tpcc_56 SRVID=56

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n57"
        RQADDR=tpcc_57 SRVID=57

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n58"
        RQADDR=tpcc_58 SRVID=58

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n59"
        RQADDR=tpcc_59 SRVID=59

service SRVGRP=group12
                CLOPT="-s NEWO_SVC -s PMT_SVC -s ORDS_SVC -s STKL_SVC -s DVRY_SVC -- -n60"
        RQADDR=tpcc_60 SRVID=60
#------------------------------------------------------------------------
*SERVICES
#------------------------------------------------------------------------
*ROUTING
#------------------------------------------------------------------------
```

Appendix D RTE Configuration

This appendix lists RTE input parameters and code fragments used to generate each transaction input file, to demonstrate the RTE was configured to generate transaction input data as specified in *Clause 2* of the specification.

D.1 RTE Parameters

Run2/TESTENV

```
#############################################################################
# Environment variables for running TPC-C
#############################################################################
setenv COMMENT   "Keystone (16-way) 64bit 11.i (11.11)Kernel"
setenv DATABASE   "sybase"       # name of the database used to run the test
                  # can be either "oracle", "sybase", or
                  # "sqlserver"
setenv OPS          0      # Set to 1 if using OPS
setenv NT           0      # Set to 1 if using NT
setenv BATCH_TPCC       0      # Set to 1 for "batch_tpcc" with the
                  # RUNME interface, 0 for c/s TPC-C.
setenv DATABASE_REV     "12.5"   # Revision of the database
                  # For Sybase, valid versions are 11.9.3, 12.0,
                                                  # and 12.5


setenv TESTROOT     results

setenv RESULTS_NAME   test        # Directory name of RESULTS (put in root of
                  # ~tpcc).  So actual directory is
                  # ~tpcc/${TESTROOT}

#setenv TRANS_TIME        80                # Total time to run the test for (in minutes)
setenv TRANS_TIME       170                # Total time to run the test for (in minutes)
setenv CHKPNT_INTERVAL    900    # 15 min to wait before forcing a checkpoint
setenv CHKPNT_INTERVAL2  1800          # Seconds to wait before forcing the second
                                                  # checkpoint
                  # For Sybase and Sqlserver, this is the # of
                  # seconds after the first checkpoint til the
                  # 2nd checkpoint.

setenv DB_SIZE       11440     # Database size on SUT(<= size actually built)
                  # value in warehouses

setenv SERVER       "sut"       # The SUT (Database Server)
setenv NR_SERVER    "1"

setenv CLIENT       "client"     # NOTE:  the client name needs to have a
                  # suffix of 1,2,3,4,.. etc starting with
                  # 1 and going to the number of clients.  The
                  # actual client names will be client1,
                  # client2, client3, etc.  You need to put
                  # the base client name here.
setenv NR_CLIENT    "8"          # number of clients

#setenv SUBST_CLIENT_LST "1 2"    # client substitutes list
setenv SUBST_CLIENT_LST "6 7 8 "    # client substitutes list

setenv DRIVER       "driver"     # NOTE:  the driver name needs to have a
                  # suffix of 1,2,3,4,.. etc starting with
                  # 1 and going to the number of drivers.  The
                  # actual driver names will be driver1,
                  # driver2, driver3, etc.  You need to put
                  # the base driver name here.
setenv NR_DRIVER        "8"      # number of drivers
# setenv NR_HOSES       "1"      # For multiple lans between client and server
setenv NR_LAN           "1"      # For multiple lans between driver and client

#############################################################################
# Load options
#############################################################################
setenv PARALLEL_LOADER    1      # Use parallel loader
setenv TPCC_PAGESIZE      4      # size in KB of the basic database page

#############################################################################
# Sybase specific defines
#############################################################################
#setenv CONFIG_FILE  ~tpcc/config/load.cfg  # database config file for load
setenv CONFIG_FILE  ~tpcc/config/sybase.cfg  # database configuration file
setenv LOAD_PARTITIONS     23    # Number of partitions to use for load
setenv HISTORY_PARTITIONS 512    # Number of times to partition history table


#############################################################################
# statistics, should probably all be off during your audit runs (performance
# runs).
#############################################################################
setenv SERVER_STATS      0      # turn on statistics on the server(1)
setenv CLIENT_STATS      0      # turn on statistics on the client(1)
setenv SAR_STATS         0      # turn on SAR (1)
setenv PI_STATS          0      # turn on PI  (PA8600) (22)
setenv FULL_PI           0      # go for the full, 25-minute cuda counts
setenv KERNEL_STATS      0      # turn on kernel gprof (20)
setenv KERNEL_TIME       0      # collection time
setenv SPIN_STATS        0      # turn on spinwatcher (18)
setenv SPIN_TIME         0      # collection time in _SECONDS_
setenv CPI_STATS         0      # turn on cpi measurement (cyclemeter) (14)
setenv NET_STATS         0      # turn on netstat (1)
setenv SAMPLER_STATS     0      # turn on KI sampler (12)
setenv SAMPLER_TIME      0      # collection time
setenv DATABASE_STATS    0      # collect Sybase statistics (16)
setenv DATABASE_TIME     0      # collection time in _SECONDS_
setenv SMC_STATS         0      # collect smcpmon statistics (26)
setenv PMON_STATS        0      # collect pmon statistics V-CLASS ONLY
setenv PMON_SECONDS      0      # collection time >>in seconds<<.  Note that
                  # pmon makes 13 passes, each this long
setenv PEPSI_STATS       0      # turn on pepsi (T5xx)
setenv JOLT_STATS        0      # turn on JOLT counters (Jade) (12)
setenv JOLT_TIME         0      # collection time in seconds for _EACH_ sample
setenv TORNADO_STATS     0      # turn on tornade (PCX-T')
setenv LOGIC_ANALYZER    0      # turn on the logic analyzer
setenv LOG_ANAL_WAIT     0      # wait time in _MINUTES_
setenv LOG_ANAL_TIME     0      # collection time in _MINUTES_

#
# Audit related stuff + misc
#
setenv CONSISTENCY       0       # run consistency checks before/after run
                  # this should be 1 when doing your final
                  # performance runs.
setenv OUTPUT_LEVEL      3       # minimum level - 3
                  # maximum level - 1
                  # need to set to 1 for durability tests

setenv RANDOMIZE_OUTPUT  1       # Specifies the percentage of users that should
                  # output full terminal data (the works) only
                                                  # if the OUTPUT level is at maximum

setenv REMOVE_OUTPUT     0       # set to 1 to remove  "success" and
```

```
                    # "deliv_results" files after each run
setenv COMPRESS_OUTPUT   0      # set to 1 to compress "success" and
                    # "deliv_results" files after each run

setenv CLEAR_LOGS       0      # set to 1 to do a dumptrans after the run


###############################################################################
# The lines below should really not be modified much (if at all)
###############################################################################
# For ODBC
# setenv SHLIB_PATH /opt/odbc/drivers:/opt/odbc/lib
setenv TRANS_NUM   1300000000     # Total number of transactions to run

setenv DELIVERY_LOGS  logs      # Directory name for logfiles

setenv RPT_WINDOW_SIZE   30     # Reporting window size in number of
                                # RPT_GRANULARITY; for example,
                                # window size is 10 minutes if
                                # RPT_GRANULARITY=30 and RPT_WINDOW_SIZE=20

setenv TRANS_TYPE       0      # 0=all, 1=new-order, 2=payment,
                               # 3=order_status, 4=delivery, 5=stock_level


#
# For TPC-C rev 3.1 and later the difference between the LOAD value of
# CLAST_CONST_C and the run value needs to be within 65-119 inclusive
# but can't be 96 or 112
#
setenv CLAST_CONST_C     208    # a run-time constant chosen within [0..255]
setenv CID_CONST_C       498    # a run-time constant chosen within [0..1023]
setenv IID_CONST_C      3415    # a run-time constant chosen within [0..8191]

setenv COPY_ENV                 1          # 1 = Copy TESTENV to other Drivers.
                                # 0 = DO NOT copy. It is the tester's
                                #     responsibility to make TESTENVs on all
                                #     the other drivers.

#
# The following emulex communication values are measured, do not change these
#
#setenv COMM_ADJUST_NEWO  0.83  # new-order comm delay
#setenv COMM_ADJUST_PMT          0.35  # payment comm delay
#setenv COMM_ADJUST_ORDS  0.47  # order-status comm delay
#setenv COMM_ADJUST_DVRY  0.29      # delivery comm delay
#setenv COMM_ADJUST_STKL  0.27      # stock-level comm delay

#
# The following COMM delays should be used when using the HUB solution
#
setenv COMM_ADJUST_NEWO         0.00   # new-order comm delay, Convert TELNET to DTCs
setenv COMM_ADJUST_PMT          0.00   # payment comm delay
setenv COMM_ADJUST_ORDS         0.00   # order-status comm delay
setenv COMM_ADJUST_DVRY         0.00           # delivery comm delay
setenv COMM_ADJUST_STKL         0.00           # stock-level comm delay

#
# The following menu value are measured for Emulex, do not change these
#
#setenv NEWO_MENU        0.56   # new order menu RTE delay
#setenv PMT_MENU         0.41   # payment menu RTE delay
#setenv OS_MENU          0.32   # order status menu RTE delay
#setenv DVRY_MENU        0.45   # delivery menu RTE delay
#setenv STKL_MENU        0.46   # stock menu RTE delay

#
# Use the following menu times if using HUBs instead of Emulex
#
setenv NEWO_MENU        0.00   # new order menu RTE delay
setenv PMT_MENU         0.00   # payment menu RTE delay
setenv OS_MENU          0.00   # order status menu RTE delay
setenv DVRY_MENU        0.00   # delivery menu RTE delay
setenv STKL_MENU        0.00   # stock menu RTE delay

#
# Keying times  Don't change these unless doing special tests.  They need
# to be float values.
#
setenv NEWO_KEY         18.01   # new order keying time (18.0)
setenv PMT_KEY           3.01   # payment keying time (3.0)
setenv OS_KEY           2.01   # order status key time (2.0)
setenv DVRY_KEY         2.01   # delivery key time (2.0)
setenv STKL_KEY         2.01   # stock level key time (2.0)

#
# Think times.  Twiddle these as needed.  They need to be float values.
#
setenv NEWO_THINK       12.12   # new order keying time (12.20)
setenv PMT_THINK        12.05   # payment keying time (12.20)
setenv OS_THINK         10.10   # os keying time (10.25)
setenv DVRY_THINK        5.05   # delivery keying time (5.20)
setenv STKL_THINK        5.05   # stock level keying time (5.20)
```

D.2 Field Value Generation

Source/src/driver/generate.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 97/12/15 13:53:51 $

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

#include <stdio.h>
#include <values.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <fcntl.h>
#include <signal.h>
#include <math.h>

#include "shm_lookup.h"
#include "random.h"

#include <time.h>

int CLAST_CONST_C = 208;
int CID_CONST_C =  498;
int IID_CONST_C = 3415;

int trans_type = 0;   /* type of transaction 0 == all */

extern ID warehouse;
extern ID district;
```

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

neworder_gen(t)
   neworder_trans *t;
   {
   int i;

   t->W_ID = warehouse;

   t->D_ID = RandomNumber(1, no_dist_pw);
   t->C_ID = NURandomNumber( 1023, 1, no_cust_pd, CID_CONST_C);

   t->O_OL_CNT = RandomNumber(5, 15);

   for (i=0; i<t->O_OL_CNT; i++)
      {
      t->item[i].OL_I_ID = NURandomNumber(8191, 1, no_item, IID_CONST_C);
      t->item[i].OL_SUPPLY_W_ID = RandomWarehouse(warehouse, scale, 1);
      t->item[i].OL_QUANTITY = RandomNumber(1, 10);
      }

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

payment_gen(t)
   payment_trans *t;
   {

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

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

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

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

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

   }

ordstat_gen(t)
   ordstat_trans *t;
   {

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

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

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

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

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

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

   /* choose a random number between 0.0 and 1.0 */
   if (trans_type == 0) {
#ifdef USE_DRAND48
      r = drand48();
#else
      r = randy();
#endif

      /*
       * select one of STOCKLEV, DELIVERY, ORDSTAT and PAYMENT
       * based on weight
       */
      for (type = STOCKLEV;  type > NEWORDER;  type--) {
         r -= weight[type];
         if (r < 0) break;
      }
   } else {
            /* user wants only a certain type (say all stocklevel) so do that
      instead */
            type = trans_type;
   }
   /* return the value of the selected card, or NEWORDER if none selected */
   return type;
   }
```

Appendix E Disk Storage

The calculations used to determine the storage requirements for the 8 hours logical log and the 60-day space calculations are contained in this appendix.

Warehouses	11500	**tpmC**	140,239.97	**tpmC/W**	12.19	
Table	**Rows**	**Data**	**Index**	**5% Space**	**8H Space**	**Total Space**
Warehouse	11,500	46,004	84	2,304		48,392
District	115,000	46,276	96	2,319		48,691
Item	100,000	9,380	28	470		9,878
New-order	103,500,000	1,617,272	4,820		230,000	1,852,092
History	345,000,000	18,871,272	0		3,682,096	22,553,368
Orders	345,000,000	9,865,728	29,368		1,930,696	11,825,792
Customer	345,000,000	230,449,240	16,563,768	12,350,650		259,363,658
Order-line	3,450,000,000	207,576,596	669,612		40,632,267	248,878,475
Stock	1,150,000,000	354,882,820	966,992	17,792,491		373,642,303
Totals		823,364,588	18,234,768	30,148,234	46,475,060	918,222,650

Segment	**LogDev Cnt.**	**Seg. Size**	**Needed**	**Overhead**	**Not Needed**
wdi	1	204,800	106,961	1,070	96,769
new_order	1	4,096,000	1,852,092	18,521	2,225,387
history	1	24,576,000	22,553,368	225,534	1,797,098
orders	1	16,384,000	11,825,792	118,258	4,439,950
customer	23+1index	278,937,600	261,957,295	2,619,573	16,980,305
order_line	23	284,037,120	251,367,260	2,513,673	30,156,188
stock	23	410,511,360	377,378,726	3,773,787	29,358,847
tmpdb	1	60,416	0	604	59,812
Totals		1,018,807,296	927,041,494	9,271,019	85,114,356

Dynamic space	228,987,875	Sum of Data for Order, Order-Line and History (excluding free extents)
Static space	652,030,735	Data + Index + 5% Space + Overhead - Dynamic space
Free space	52,674,331	Total Seg. Size - Dynamic Space - Static Space - Not Needed
Daily growth	44,679,308	(Dynamic space/W * 62.5)* tpmC
Daily spread	(14,344,631)	Free space - 1.5 * Daily growth (zero if negative)
60 day (KB)	3,332,789,217	Static space + 60 (daily growth + daily spread)
60 day (GB)	3178.40	Excludes OS, Paging and RDBMS Logs
Log per N-O txn	0.91872	Number of 4K blocks per New-Order transaction
8 Hour Log (GB)	235.92	
OS+SWAP (GB)	72.71	
Total Space Needed	3487.02	GB

	Disk Size (MB)	# of Disks	Total Capacity
VA7100(15-36GB dri	252,812	1	252,812
VA7100(5/10 setup)	167,372	10	1,673,720
VA7100(15 setup)	124,652	13	1,620,476
18GB-SYS/Swap	17,366	2	34,732
Total Storage (GB)			3,498

Appendix F Price Quotes

The following pages contain the price quotes for the hardware included in this FDR.

Andreas Hotea
HP
Cupertino, CA 95014

HP Unix Sales Development
19111 Pruneridge Aveune
Cupertino, CA 95014
(408) 447-2320

February 22, 2002

hp server rp8400

TPC-C Rev 5

Report Date: February 22, 2002

Description	Part Number	Brand	Price Key	Unit Price	Qty	Price	3Year Main.Price
Server Hardware							
hp server rp8400 Enterprise Server	A6093A		1	40000	1	40,000	49,218
750MHz PA_RISC 8700 CPU-Dual	A6444A, Opt. 0D1		1	46000	8	368,000	69,216
Processor/Memory Cell Board	A6094A, Opt. 0D1		1	7000	4	28,000	
Core I/O	A6096A, Opt, 0D1		1	5000	1	5,000	
4GB SyncDRAM Memory Modules	A6098A, Opt. 0D1		1	28000	16	448,000	
18GB Ultra SCSI Internal Drive	A6723A, Opt 0D1		1	1764	2	3,528	
DVD Drive	A6180A, Opt. 0D1		1	460	1	460	
PCI Fibre Channel 2X	A5158A, Opt 0D1		1	2240	8	17,920	
PCI 1000BT Lan Adapter	A4926A, Opt. 0D1		1	2135	1	2,135	
Rack System/E Cabinet	J1528AZ		1	510	1	510	
4.5m, Server to UPS Power Cord	J1528AZ, Opt. A5F		1	0	1	0	
System Hotswap Power Supply	A6099A, Opt 0D1		1	560	4	2,240	
HPUX Opertating Environment	B9088AC		1	520	1	520	
6.5kW/9kVA HP UPS Rackmount	A6584A		1	8799	5	43,995	
Surestore VA7100 w/dual controllers 512/MB cache	A6262A		1	44250	24	1,062,000	101,064
18GB 15K RPM FC HDD	A6191A		1	914	295	269,630	
36GB 10K RPM FC HDD	A6192		1	1026	65	66,690	
2 meter Fibre Optic Cable	A3583A		1	175	24	4,200	
10 Port Short Wave Fibre Channel Hub	A3724A		1	9690	8	77,520	
16 meter Fibre Optic Cable	92227C		1	200	8	1,600	
HP9000 Std. Rack System E41	A4902AZ		1	2350	4	9,400	
Modular Power Dist.	A5137AZ		1	155	8	1,240	
200-240 Volts	A5137AZ, Opt AW4		1	115	8	920	
				Subtotal		**2,453,508**	**219,498**
Client Hardware							
hp workstation c3700 with 2GB Memory	A6057B		1	13390	8	107,120	31,048
1 GB Memory Moudle	A6016A, Opt. OD1		1	1395	48	66,960	
700/96 Console	C1064GX		1	550	1	550	
100BaseT 4-Port PCI Lan Adapter	A5506B		1	1830	8	14,640	
18 GB LVD 10K RPM Disk	A4998A, Opt. OD1		1	595	8	4,760	
				Subtotal		**194,030**	**31,048**
Client Software							
HP C/ANSI C Compiler	B3901BA, Option AH0		1	1,600	1	1,600	170
				Subtotal		**1,600**	**170**
User Connectivity							
HP ProCurve Switch 8000M	J4110A		1	1,699	1	1,699	821
HP ProCurve Switch 10/100Base-T Module	J4111A		1	509	1	509	
HP ProCurve Switch Gigabit-SX Module	J4113A		1	1,189	1	1,189	
				Subtotal		**3,397**	**821**
Large configuration Discount and Support Prepayment*						(1,326,268)	(93,050)
* All discounts are based on US list prices and for similar quantities and configurations				**Total**		**1,326,268**	**158,487**

All the components in the price list are currently available. Maintenance support price is for 24 hours, 7 days with 4 hour response time.

The eCommerce Transaction Platform

February 22, 2002

Ms. Lucille Boushey
TPC-C Performance Project Manager
Hewlett Packard
408 447 7364
408 447 5958 FAX

Dear Ms. Boushey:

Per your request I am enclosing the pricing information regarding TUXEDO 6.4 that you requested. This pricing applies to Tuxedo 6.4, 6.5, 7.1, and 8.0. Please note that Tuxedo 8.0 is our most recent version of Tuxedo. Core functionality services pricing is appropriate for your activities. As per the table below HP PA-RISC systems are classified as either a Tier 1, 2, 3, 4 or 5 systems depending on the performance and CPU capacity of the system. This quote is valid for 60 days from the date of this letter.

10.1.1 Tuxedo Core Functionality Services (CFS) Program Product Pricing and Description

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.4, 6.5, 7.1, and 8.0. Prices range from $3,000 for Tier 1 to $250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees. Note that a 5% discount will apply to total list price license purchases less than $100,000 (for instance 8 Tier 1 servers; C3700 Workstations – 8 * 3,000 = $24,000 - would be eligible for a 5% discount). Support is not discountable.

Very Truly Yours,

Robert J. Gieringer

Rob Gieringer,
Worldwide Pricing Manager

10.1.1.1 BEA Tux/CFS Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 8) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers	**Unlimited**	**$3,000.00**	**$480.00**	**$690.00**
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations with up to 2 CPUs	**Unlimited**	**$12,000.00**	**$1,920.00**	**$2,760.00**
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	**Unlimited**	**$30,000.00**	**$4,800.00**	**$6,900.00**
Tier 4 - Large (more than 8, less than 32 CPUs)	**Unlimited**	**$100,000.00**	**$16,000.00**	**$23,000.00**
Tier 5 - Massively Parallel Systems, > 32 processors	**Unlimited**	**$250,000.00**	**$40,000.00**	**$57,500.00**

	Tier 1	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Operating System						
HP/UX 9.X;10.X	**Uni-processor Workstation** **B Class - 132/180/2000** **C Class (3000/3600 / 3700)**	**9000/E25** **9000/E35** **9000/E45** **9000/E55** **9000/G30** **9000/G40** **9000/A180** **9000/A180C** **9000 /A400**	**9000/G50** **9000/G60** **Multi-Processor Workstations** **J Class (J282/J2240/J5600/J6000/J6700)** **9000/R380,390** **9000/D200,210** **220/30/50/60/80** **D310/20/30** **D350/60/70/80** **9000 /A500** **9000 – L1000** **9000 – R Class**	**9000/H20, 30** **9000/H40, 50** **9000/I30, 40** **9000/K1XX** **9000 – L2000/L3000** **9000/I50,60** **9000/H60** **9000/G70** **9000/H70** **9000/I70** **9000/K2XX** **9000/K3XX** **9000/K4XX** **9000/K5XX** **N4xxx Series**	**9000/T500, T520,T600** **1-16 CPUs** **S-Class**	**9000/V series all models** **X-Class** **9000 Series - Superdome**