SiliconGraphics
Computer Systems

Silicon Graphics Origin2000™ using IRIX™ 6.5 and Sybase Adaptive Server Enterprise® 11.5.1

TPC Benchmark™ C Full Disclosure Report

First Edition
April 23, 1998

First Edition, April 23, 1998.

Silicon Graphics, Inc and Sybase, Inc. believe that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. The sponsors assume 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, Silicon Graphics Inc. 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. Silicon Graphics Inc. 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., April 23, 1998.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark™ C test conducted on the Silicon Graphics Origin2000™ (two 250 MHz MIPS R10000 64-bit CPUs) in a client/server configuration using Sybase Adaptive Server Enterprise 11.5.1 and BEA TUXEDO 6.3 transaction monitor in conformance with the requirements of the TPC Benchmark™ C Standard Specification, Revision 3.3. The operating system used for the benchmark was IRIX™ 6.5.

TPC Benchmark™ C Metrics

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

Standard and Executive Summary Statements

Pages iii-v contain the Executive Summary of the benchmark results for the Silicon Graphics Origin2000™.

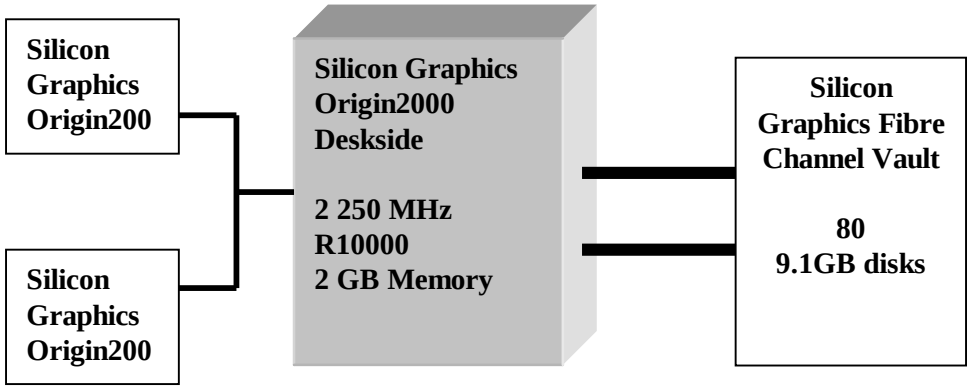
Auditor

The benchmark configuration, environment and methodology used to produce and validate the test results, and the pricing model used to calculate the cost per tpmC, were audited by Richard Gimarc of Performance Metrics, Inc. to verify compliance with the relevant TPC specifications.

Silicon Graphics, Inc. Sybase, Inc.	Silicon Graphics Origin2000™ with 2 Origin200™ clients	TPC-C® 3.3 REPORT DATE: **April 23, 1998**

TOTAL SYSTEM COST	TPC-C THROUGHPUT	PRICE/PERFORMANCE	AVAILABILITY DATE
$736,822	**6040.33 tpmC®**	**$121.98/tpmC®**	**July 31, 1998**

PROCESSORS	DATABASE MANAGER	OPERATING SYSTEM	OTHER SOFTWARE	NUMBER OF USERS
2 MIPS R10000 250 MHz	**Sybase Adaptive Server Enterprise 11.5.1**	**IRIX™ 6.5**	**BEA TUXEDO Transaction Monitor 6.3**	**5050**



SYSTEM COMPONENTS	SERVER SYSTEM	EACH CLIENT
Processors	2 250 MHz MIPS R10000 each with 32 KB I Cache, 32 KB D Cache and 4 MB Unified Secondary Cache	2 180 MHz MIPS R10000 32 KB I Cache 32 KB D Cache and 1 MB Unified Secondary Cache
Memory	2 GB	2 GB
Disk Controllers	1 Integral SCSI Adapters 2 2 port Fibre Channel Adapters	1 Integral SCSI Adapter
Disk Drives	1 9.1 GB SCSI Internal Disk 80 9.1 GB Fibre Channel Disks	1 4.5 GB SCSI Internal Disk
Total GB of Storage	686.88 GB	4.24 GB

Silicon Graphics, Inc. Sybase, Inc.	Silicon Graphics Origin2000™ with 2 Origin200™ clients					TPC-C® 3.3 REPORT DATE: April 23, 1998	
Description	**Part Number**	**Third Party**		**Unit Price**	**Qty**	**Extended Price**	**5 yr. Maint. Price**
Origin2000 Server Hardware		**Brand**	**Notes**				
O2000 DS 2xR10000 250Mhz,4MB Cache,10/100BaseT port, CDROM	D1-H6502-4	SGI	2	$46,427	1	$46,427	$17,304
Internal Disk Drives - 9GB Capacity	FTO-9GB	SGI	2	$0	1	$0	$0
Origin2000 Destination Kit	DK-LD1-001	SGI	2	$0	2	$0	$0
First 512MB in One Bank	OPN-512-D1	SGI	2	$6,050	1	$6,050	$638
Additional 512MB in One Bank	H4-N512-D1	SGI	2	$6,050	3	$18,150	$1,920
2 Port Fibre Channel XIO Adapters for O2000	XT-FC-2P	SGI	2	$3,150	2	$6,300	$728
10-Meter Copper Fibre Channel Cables	X-F-COP-10M	SGI	2	$100	4	$400	$0
0.3-Meter Copper Fibre Channel Cables	X-F-COP-0.3M	SGI	2	$50	4	$200	$0
Empty Fibre Channel Rack	P-F-RACK	SGI	2	$5,900	1	$5,900	$634
Rackmount FibreVault w/10x9.1GB Fibre Channel Disk Drives	P-F-B10X9-R	SGI	2	$22,070	8	$176,560	$18,684
4-port 100 Mbps FAST Ethernet Hub (incl. 2 spares)	NX-DH4	Netlux	3	$85	3	$255	$0
100 BaseT ethernet cable for client connectivity	9360005-10ft	SGI	2	$35	3	$105	$0
					Subtotal	$260,347	$39,908
Origin2000 Server Software							
IRIX Operating System (incl. In hardware quote)	SC4-IRIX-6.5	SGI	2	$0	1	$0	$0
CD-ROM Update Media Required (support only)	MO5-CD	SGI	2	$0	1	$0	$1,042
Data Base Accel. (incl. With IRIX 6.5)	SC4-DBA-2.0	SGI	2	$0	1	$0	$0
Sybase Open Client		Sybase		$755	1	$755	$636
Sybase SQL Server 11.5.1		Sybase		$142,500	1	$142,500	$120,000
					Subtotal	$143,255	$121,678
Origin200 Client Hardware							
2x180mhz cpu O200, 128MB memory, 1x100BaseT	L1-S6302-1	SGI	2	$14,158	2	$28,316	$12,980
Internal Disk Drives - 4GB Capacity	FTO-4GB	SGI	2	$0	2	$0	$0
Skins for free standing tower configuration	FTO-S1-TMOD	SGI	2	$0	2	$0	$0
Origin200 Destination Kit	DK-SL1-001	SGI	2	$0	2	$0	$0
First 512MB Memory in One Bank	OPN-512-D1	SGI	2	$6,052	2	$12,104	$638
Additional 512MB Memory in One Bank	H4-N512-D1	SGI	2	$6,052	6	$36,312	$3,840
PCI 1-Port Fast Ethernet Adapter	PCI-FE-TX-MK	SGI	2	$825	6	$4,950	$441
					Subtotal	$81,682	$17,899
Origin200 Client Software							
IRIX Operating System (bundled w/ hardware)	SC4-IRIX-6.5	SGI	2	$0	2	$0	$0
CD-ROM Update Media Required (support only)	MO5-CD	SGI	2	$0	2	$0	$1,042
Development option (compiler and development environment)	SC4-C-7.2	SGI	2	$570	1	$570	$441
BEA Tuxedo/CFS Version 6.3 Runtime/Unlimited User	6051033-01	BEA	1	$12,000	2	$24,000	$18,000
					Subtotal	$24,570	$19,483
User Connectivity							
Netlux 10 Base-T 8+A33+1 Port Hub (incl. 10% spares)	NX-H9+	Netlux	3	$40	700	$28,000	$0
					Subtotal	$28,000	$0
					Total	**$537,854**	**$198,968**
Notes:						**Five-Year Cost of Ownership:**	**$736,822**
1=BEA Systems., 2=Access Graphics, 3=Netlux						**tpmC Rating:**	**6040.33**
Audited by Performance Metrics Inc.						**$ / tpmC:**	**$121.98**

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

Silicon Graphics, Inc. Sybase, Inc.	Numerical Quantities Summary	TPC-C™ Rev 3.3 Report Date: April 23, 1998

MQTh, Computed Maximum Qualified Throughput: 6040.33 tpmC

Response Times (in seconds):

Transaction	90th %ile	Max	Avg.
New-Order	2.37	6.73	1.06
Payment	0.89	5.21	0.67
Order-Status	1.04	2.82	0.75
Delivery (interactive portion)	0.49	0.74	0.30
Delivery (deferred portion)	2.19	116.24	1.39
Stock-Level	9.00	16.23	4.00
Menu	0.42	1.35	0.33

Transaction Mix (% of Total Transactions):

New-Order	44.75
Payment	43.15
Order-Status	4.04
Delivery	4.02
Stock-Level	4.04

Keying/Think Times (seconds):

	Keying Time			Think Time		
	Min	Avg.	Max	Min.	Avg.	Max
New-Order	18.01	18.02	18.10	0.10	12.13	120.00
Payment	3.00	3.02	3.05	0.10	12.07	120.00
Order-Status	2.00	2.05	2.06	0.10	10.05	90.20
Delivery	2.00	2.05	2.05	0.10	5.05	43.00
Stock-Level	2.01	2.05	2.05	0.10	5.07	50.00

Test Duration:

Ramp-up time	50 minutes
Measurement interval	30 minutes
Transactions during measurement interval	404902
Ramp down time	3 minutes

Checkpoints:

Number of checkpoints in measurement interval	1
Checkpoint interval	30 minutes

Reproducibility:

Reported measurement (tpmC)	6040.33 tpmC
Reproducibility measurement (tpmC)	6018.17 tpmC
Reproducibility (%)	0.37 %

Table Of Contents

Abstract i
Overview i
TPC Benchmark™ C Metrics i
Standard and Executive Summary Statements i
Auditor i

Table Of Contents vi

Preface 1
Document Stucture 1
TPC Benchmark™ C Overview 1

General Items 3
1.1 Application Code and Definition Statements 3
1.2 Test Sponsor 3
1.3 Parameter Settings 3
1.4 Configuration Items 4

Clause 1 Related Items 7
2.1 Table Definitions 7
2.2 Physical Organization of Database 7
2.3 Insert and Delete Operations 7
2.4 Partitioning 8

Clause 2 Related Items 9
3.1 Random Number Generation 9
3.2 Input/Output Screen Layout 9
3.3 Priced Terminal Feature Verification 9

3.4 Presentation Manger or Intelligent Terminal 10
3.5 Transaction Statistics 10
3.6 Queueing Mechanism 10

Clause 3 Related Items 11
4.1 Transaction System Properties (ACID) 11
4.2 Atomicity 11
4.2.1 Completed Transaction 11
4.2.2 Aborted Transaction 12
4.3 Consistency 12
4.4 Isolation 13
4.4.1 Isolation Test 1 13
4.4.2 Isolation Test 2 13
4.4.3 Isolation Test 3 14
4.4.4 Isolation Test 4 14
4.4.5 Isolation Test 5 14
4.4.6 Isolation Test 6 15
4.4.7 Isolation Test 7 15
4.5 Durability 16
4.5.1 Loss of Data Disk 16
4.5.2 Loss of Log Disk 17
4.5.3 Instantaneous interruption and loss of memory 17

Clause 4 Related Items 18
5.1 Initial Cardinality of Tables 18
5.2 Database Layout 18
5.3 DBMS: Data Model and Interfaces 21
5.4 DBMS Partitions/Replications 21
5.5 DBMS Space Requirements 21

Clause 5 Related Items 23
6.1 Measured Throughput 23
6.2 Response Times 23
6.3 Keying and Think Times 23
6.4 Response Time Frequency Distribution Curves 24
6.5 Response Time Vs. Throughput 27
6.6 Think Time Frequency Distribution Curves 27
6.7 New-Order Throughput Vs. Time 28
6.8 Steady State Determination 29
6.9 Work Performed During Steady State 29
6.10 Reproducibility 30
6.11 Measurement Period Duration 30
6.12 Transaction Mix Regulation 30
6.13 Transaction Mix 30
6.14 Transaction Statistics 30
6.15 Checkpoints 31

Clause 6 Related Items 33

7.1 Remote Terminal Emulator Description....33
7.2 Emulated Components....33
7.3 Configuration Diagrams....33
7.4 Network Configuration....34
7.5 Network Bandwidth....34
7.6 Operator Intervention....34

Clause 7 Pricing Related Items....35
8.1 Priced System....35
8.2 Total Five Year Price....35
8.3 Availability Date....35
8.4 Measured tpmC and Price/tpmC....36

Clause 9 Related Items....37
9.1 Auditor's Report....37

Appendix A: Application Code....41

Appendix B: Database Design....79

Appendix C: Tunable Parameters....102

Appendix D: Disk Storage....113

Appendix E: RTE Configuration....115

Appendix F: Price Quotes....119

Preface

Document Stucture

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

This report is submitted to satisfy the specification's requirement for full disclosure. It documents the compliance of the benchmark implementation and execution reported for the Silicon Graphics Origin2000™ Server using Sybase Adaptive Server Enterprise 11.5.1.

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

Each section in this report begins with the text of the corresponding item from Clause 8 of the specification, printed in *italic* type. The following plain text explains how this benchmark complies with that specific portion of the specification. In sections where Clause 8 requires extensive listings, the appropriate appendix in this report is referenced.

TPC Benchmark™ C Overview

TPC Benchmark™ C was developed by the Transaction Processing Performance Council (TPC). It is the intent of the TPC to develop a suite of benchmarks to measure the performance of computer systems executing a wide range of applications. Silicon Graphics and Sybase are active participants in the TPC to define and develop such a suite of benchmarks.

TPC Benchmark™ C is an On-Line Transaction Processing (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 accesses through primary and secondary keys.
- Databases consisting of many tables with a variety of sizes, attributes, and relationships.
- Contention of data accesses 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 complaint 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.

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 application program used in this TPC Benchmark™ C.

1.2 Test Sponsor

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

Silicon Graphics, Inc. and Sybase, Inc. were joint 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 Tuning Options*
- *Recover/commit options*
- *Consistency/locking options*

- *Operating system and application configuration parameters*
- *Compilation and linkage options and run-time optimizations used to create/install applications, OS, and/or databases*
- *Compiler optimization options*

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 C contains the parameters used in this benchmark. Also included are all of the Sybase Adaptive Server Enterprise 11.5.1 database parameters and the TUXEDO transaction monitor parameters used.

1.4 Configuration Items

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 their protocol type.*
- *Number of LAN (e.g. Ethernet) Connections, including routers, workstations, terminals, etc. that were physically used in the test or are incorporated into the pricing structure.*
- *Type and the run-time execution location of software components (e.g., DBMS, query processing tools/languages, middle-ware components, software drivers, etc.).*

The server was an Origin2000™ with:

- Two 250 MHz MIPS R10000 64-bit CPUs, each with 4MB of L2 cache
- 2 GB of main memory
- One Base I/O card, which includes internal SE Ultra SCSI, external SE Ultra SCSI, 100Base-Tx ethernet port & two 460kbps serial ports.
- One 9.1 GB internal SCSI disk drive.
- 2 2-port fibre channel adapters.
- 80 9.1 GB fibre channel disk drives

The benchmark configuration used a Remote Terminal Emulator (RTE) to emulate TPC-C user sessions. The measured and the priced configurations are shown in Figure 1.1 and Figure 1.2 respectively.

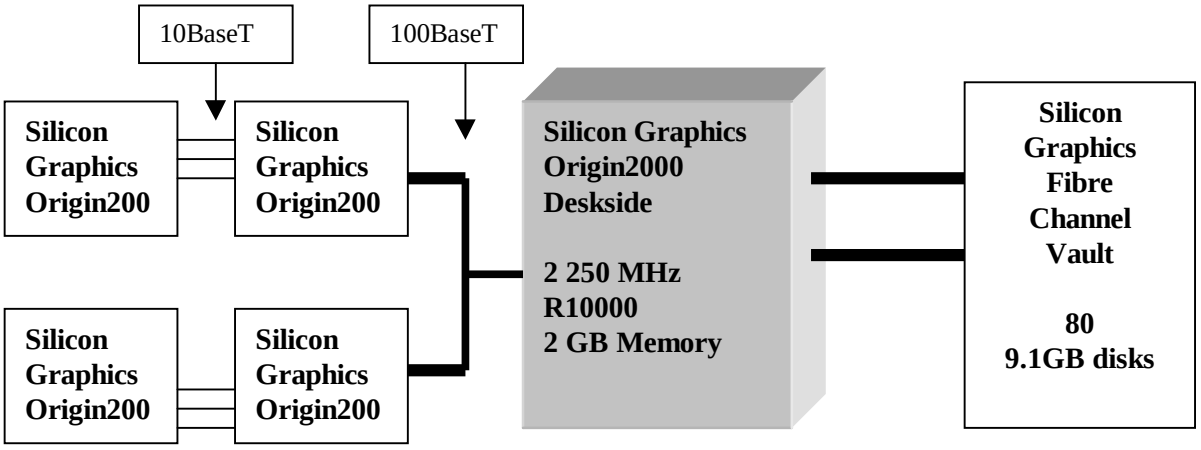


Figure 1.1 Benchmarked Configuration

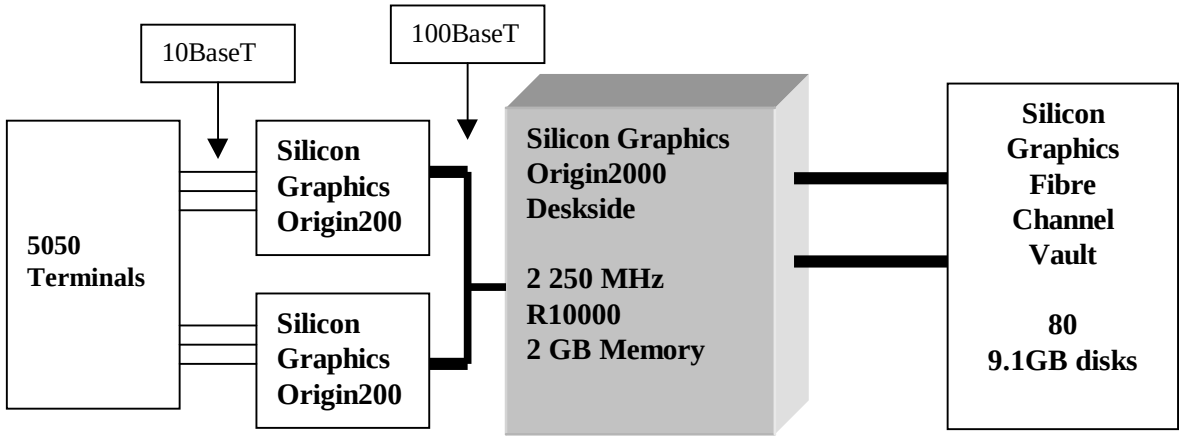


Figure 1.2 Priced Configuration

Clause 1 Related Items

2.1 Table Definitions

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

Appendix B describes the programs that define, create and populate a Sybase Adaptive Server Enterprise 11.5.1 database for TPC-C testing.

2.2 Physical Organization of Database

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

Disk space was allocated to Sybase Adaptive Server Enterprise 11.5.1 on the server according to the data in section 5.2. The size of the database table space on each disk drive was calculated to provide an 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, if used, 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.

Clause 2 Related Items

3.1 Random Number Generation

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

The RTE software, Empower, from Performix, Inc., generated the random numbers used in this benchmark. Empower computes random integers using an implementation of a linear congruential sequence as described in "The Art of Computer Programming, Volume 2/Seminumerical Algorithms" by Donald E. Knuth, copyright 1981 by Addison-Wesley Publishing Company.

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).

This was verified manually by the auditor by exercising each specification on a Silicon Graphics O2 workstation.

3.4 Presentation Manger or Intelligent Terminal

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

The TPC-C forms module was implemented using the capabilities of an xterm terminal emulator. A presentation manager was not used.

3.5 Transaction Statistics

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

Table 3.1: Transaction Statistics

Transaction Type	Statistics	Value
New Order	Home warehouse	98.95%
	Remote warehouse	1.05%
	Rolled back transactions	1.00%
	Average items per order	10.01
Payment	Home warehouse	84.89%
	Remote warehouse	15.11%
	Non-primary key access	60.18%
Order Status	Non-primary key access	60.18%
Delivery	Skipped transactions	0
Transaction Mix	New order	44.75%
	Payment	43.15%
	Order status	4.04%
	Delivery	4.02%
	Stock level	4.04%

3.6 Queueing Mechanism

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

Delivery transactions were submitted to servers using the same TUXEDO 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.

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

4.2.2 Aborted Transaction

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

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

4.3 Consistency

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

The TPC Benchmark C Standard Specification requires the System Under Test to meet the 12 consistency conditions listed in Clause 3.3.2. The specification requires explicit demonstration that the conditions are satisfied for the first four conditions only. In order to demonstrate the consistency of the application, the following steps were taken:

1. Prior to the start of the benchmark run, the consistency of the database was verified by applying the consistency conditions 1-4 described above.
2. After each measurement run, all consistency tests 1-4 were performed.

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

4.4 Isolation

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

Sybase Adaptive Server Enterprise 11.5.1 ensures isolation and full serializability by locking strategies that preserve data integrity when multiple users are accessing a database.

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

4.4.1 Isolation Test 1

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

1. At terminal 1, a New-Order transaction was started but not COMMITted.
2. At terminal 2, an Order-Status transaction was started for the same customer used on terminal 1. This transaction attempted to read the data for the order on terminal 1.
3. Terminal 2 transaction waited for the terminal 1 transaction to complete.
4. COMMITted the transaction on terminal 1. The transaction on terminal 2 now completed.
5. The results from the Order-Status transaction matched the data entered for the New-Order.

4.4.2 Isolation Test 2

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

1. Completed an Order-Status transaction.
2. At terminal 1, a New-Order transaction was started for the same customer used in step 1, but not COMMITted.
3. At terminal 2, an Order-Status transaction was started for the same customer used at terminal 1. This transaction attempted to read the data for the order at terminal 1.
4. The terminal 2 transaction waited for the terminal 1 transaction.

5. A ROLLBACK was executed for the transaction at terminal 1. The transaction at terminal 2 now completed.
6. The results from the Order-Status transaction matched the data returned in step 1.

4.4.3 Isolation Test 3

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

1. At terminal 1, a New-Order transaction was started but not committed.
2. At terminal 2, another New-Order transaction was started for the same customer used at terminal 1.
3. The terminal 2 transaction waited for the terminal 1 transaction to complete.
4. COMMITted the transaction at terminal 1. The transaction at terminal 2 completed.
5. The order number for the terminal 2 transaction was one greater than the terminal 1 transaction. The Next Order Number for the district reflected the results from both transactions.

4.4.4 Isolation Test 4

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

1. At terminal 1, a New-Order transaction was started but not COMMlTted.
2. At terminal 2, another New-Order transaction was started for the same customer used at terminal 1.
3. The terminal 2 transaction waited for the terminal 1 transaction to complete.
4. A ROLLBACK was executed for the transaction on terminal 1. The transaction on terminal 2 completed.
5. The order number for the terminal 2 transaction was one greater than the previous transaction. The Next Order Number for the district reflected the results from only the terminal 2 transaction. In other words, it had been incremented by one.

4.4.5 Isolation Test 5

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

1. At terminal 1, a Delivery transaction was started but not COMMITted.
2. At terminal 2, a Payment transaction was started for the same customer used at terminal 1.
3. Terminal 2 transaction waited for the terminal 1 transaction to complete.
4. COMMITted the transaction at terminal 1. The transaction at terminal 2 completed.

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

4.4.6 Isolation Test 6

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

1. At terminal 1, a Delivery transaction was started but not COMMITed.
2. At terminal 2, a Payment transaction was started for the same customer used on terminal 1.
3. Terminal 2 transaction waited for terminal 1 transaction to complete.
4. A ROLLBACK was executed for the transaction on terminal 1. The transaction on terminal 2 completed.
5. The customer balance reflected the result of the Payment transaction only.

4.4.7 Isolation Test 7

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

1. At terminal 1, New-Order transaction Tl was started and was queried for the price of two items, item x and item y. The transaction completed.
2. At terminal 2, New-Order transaction T2 was started for a group of items including item x twice and item y. The transaction was stopped immediately after querying the price of item x for the first time and before querying the price of x for the second time or item y.
3. At terminal 1, an interactive SQL transaction,T3, was started to increase the price of items x and y by ten percent. The transaction failed to complete.

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

4. Continued terminal 2 transaction, The price of items x (the second time) and y matched the values read by step 1. Completed the transaction.
5. Transaction T3 completed and COMMITted.
6. At terminal 1, another transaction was started to query the prices of items x and y. Completed the transaction.
7. The prices read by the transaction in step 6 matched those set by transaction T3.

4.5 Durability

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

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

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

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

Three durability tests were executed to satisfy the requirements of the specification. The test for loss of memory and instantaneous interruption was combined and performed with a fully scaled database with 5050 emulated users. The loss of log and loss of data tests were performed on a 10 warehouse database with 100 emulated users. To the best of our knowledge, these tests prove that the fully scaled configuration used for the throughput test would also pass all the durability tests.

4.5.1 Loss of Data Disk

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

1. The database was backed up (database dump) to disk.
2. 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).
3. The benchmark was executed with 100 users. On the driver system, the committed and rolled back New-Order transactions were recorded in a "success" file.
4. After 5 minutes into the measurement period one of the data disks was overwritten using the `dd` command.
5. When the Sybase dataserver recorded corruption errors, the test was aborted on the driver.
6. The transaction log was dumped to a file.
7. The original database was restored from the backup copy in step 1.
8. Sybase was restarted and its transaction log that was dumped in step 6 was used to roll forward the transactions that had completed since the backup.
9. The contents of the "success" file on the driver and the ORDERS table were compared to verify that records in the "success" file for committed New-Order transaction had corresponding records in the ORDERS table.
10. Step 2 was repeated to determine the total number of orders (count2). Count2-count1 was the same as the number of committed records in the "success" file.

4.5.2 Loss of Log Disk

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

1. The D_NEXT_O_ID fields for all rows in district table were added to determine the initial count of the total number of orders (count1).
2. A test was executed with 100 users. On the driver system, the committed and rolled back New-Order transactions were recorded in a "success" file.
3. After 5 minutes into the measurement period, one of the log disks was powered down.
4. Since the log disk is mirrored, the system continues to process transactions despite the missing disk.
5. The test was allowed to run until completion.
6. The contents of the "success" file on the driver and the ORDERS table were compared to verify that records in the "success" file for committed New-Order transaction had corresponding records in the ORDERS table.
7. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 was the same as the number of committed records in the "success" file.

4.5.3 Instantaneous interruption and loss of memory

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

1. The D_NEXT_O_ID fields for all rows in district table were summed up to determine the initial count of the total number of orders (count1).
2. A fully scaled test was executed. On the driver system, the committed and rolled back New-Order transactions were recorded in a "success" file.
3. After 5 minutes into the measurement period, the SUT's primary power was removed.
4. The test was aborted on the driver.
5. Power was restored to the SUT and a normal system recovery was done. A recovery was automatically performed by Sybase Adaptive Server Enterprise 11.5.1 when the database was restarted and brought on-line. The recovery restored the database to the consistent point just after the last committed transaction had occurred before the induced failure.
6. The contents of the "success" file on the driver and the ORDERS table were compared to verify that records in the "success" file for committed New-Order transactions had corresponding records in the ORDERS table. The number of transactions missed "in flight" were less than the number of users.
7. Step 1 was repeated to determine the total number of orders (count2). Count2-count1 was compared with the number of committed records in the "success" file.

Clause 4 Related Items

5.1 Initial Cardinality of Tables

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

The TPC-C database for this test was configured with 505 warehouses.

Table 5.1: Initial cardinality of Database Tables

Table	Occurrences
Warehouse	505
District	5050
Customer	15150000
History	15150000
Order	15150000
New Order	4545000
Order Line	151500001
Stock	50500000
Item	100000

5.2 Database Layout

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

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

Table 5.2: Disk Layout

Device Name	Allocated Space (MB)	Physical Disk Usage
stock01	390	29d70s1
stock02	390	30d90s1
stock03	390	29d71s1
stock04	390	30d91s1
stock05	390	27d32s1
stock06	390	28d52s1
stock07	390	29d72s1
stock08	390	30d92s1
stock09	390	27d33s1
stock10	390	28d53s1
stock11	390	29d73s1
stock12	390	30d93s1
stock13	390	27d34s1
stock14	390	28d54s1
stock15	390	29d74s1
stock16	390	30d94s1
stock17	390	27d35s1
stock18	390	28d55s1
stock19	390	29d75s1
stock20	390	30d95s1
stock21	390	27d36s1
stock22	390	28d56s1
stock23	390	29d76s1
stock24	390	30d96s1
stock25	390	27d37s1
stock26	390	28d57s1
stock27	390	29d77s1
stock28	390	30d97s1
stock29	390	27d38s1
stock30	390	28d58s1
stock31	390	29d78s1
stock32	390	30d98s1
stock33	390	27d39s1
stock34	390	28d59s1
stock35	390	29d79s1
stock36	390	30d99s1
stock37	390	27d40s1
stock38	390	28d60s1
stock39	390	29d80s1
stock40	390	30d100s1
stock41	390	27d41s1
stock42	390	28d61s1
stock43	390	29d81s1
stock44	1000	30d101s1
customer01	515	27d42s1
customer02	515	28d62s1
customer03	515	29d82s1

customer04	515	30d102s1
customer05	515	27d43s1
customer06	515	28d63s1
customer07	515	29d83s1
customer08	515	30d103s1
customer09	515	27d44s1
customer10	515	28d64s1
customer11	515	29d84s1
customer12	515	30d104s1
customer13	515	27d45s1
customer14	515	28d65s1
customer15	515	29d85s1
customer16	515	30d105s1
customer17	515	27d46s1
customer18	515	28d66s1
customer19	515	29d86s1
customer20	1000	30d106s1
cust_idx1	180	27d47s4
cust_idx2	180	28d67s4
cust_idx3	180	29d87s4
cust_idx4	500	30d107s4
orders1	109	27d48s4
orders2	109	28d68s4
orders3	109	29d88s4
orders4	500	5-way striped: 30d108s3 27d49s3 28d69s3 28d89s3 30d109s3
order_line1	1842	30d108s4
order_line2	1842	27d49s4
order_line3	1842	28d69s4
order_line4	1842	29d89s4
order_line5	1842	30d109s4
order_line6	2000	6-way striped: 28d67s3 29d87s3 30d107s3 27d48s3 28d68s3 29d88s3
w_d_no_i	130	12-way striped: 27d47s2 28d67s2 29d87s2 30d107s2 27d48s2 28d68s2 29d88s2 30d108s2 27d49s2 28d69s2 29d89s2 30d109s2
history	1125	12-way striped: 27d47s1 28d67s1 29d87s1 30d107s1 27d48s1 28d68s1 29d88s1 30d108s1 27d49s1 28d69s1 29d89s1 30d109s1
tpcc_log1	2000	27d30s1
tpcc_log2	2000	27d30s2
tpcc_log3	2000	27d31s1
tpcc_log4	2000	27d31s2
tpcc_log5	2000	27d31s11
tpcc_log6	2000	27d31s12

mirror_tpcc_log1	2000	28d50s1
mirror_tpcc_log2	2000	28d50s2
mirror_tpcc_log3	2000	28d51s1
mirror_tpcc_log4	2000	28d51s2
mirror_tpcc_log5	2000	28d51s11
mirror_tpcc_log6	2000	28d51s12
master	46	27d47s3

180-day storage growth requirements are met with the unused space of the tested configuration.

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

Sybase Adaptive Server Enterprise 11.5.1 is a relational DBMS. SQL stored procedures were invoked through the Sybase Open Client DB-Library.

5.4 DBMS Partitions/Replications

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

No table partitioning or replication was done.

5.5 DBMS Space Requirements

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

Appendix D lists the space requirements for the 180-day space as well as the logical log space for eight hours.

Clause 5 Related Items

6.1 Measured Throughput

Measured tpmC must be reported.

The measured tpmC was 6040.33.

6.2 Response Times

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

Table 6.1: Response Times (seconds)

Transaction	90th %ile	Max	Avg.
New-Order	2.37	6.73	1.06
Payment	0.89	5.21	0.67
Order-Status	1.04	2.82	0.75
Delivery (interactive portion)	0.49	0.74	0.30
Delivery (deferred portion)	2.19	116.24	1.39
Stock-Level	9.00	16.23	4.00
Menu	0.42	1.35	0.33

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 6.2: Keying/Think Times (seconds)

	Keying Time			Think Time		
	Min	**Avg.**	**Max**	**Min.**	**Avg.**	**Max**
New-Order	18.01	18.02	18.10	0.10	12.13	120.00
Payment	3.00	3.02	3.05	0.10	12.07	120.00
Order-Status	2.00	2.05	2.06	0.10	10.05	90.20
Delivery	2.00	2.05	2.05	0.10	5.05	43.00
Stock-Level	2.01	2.05	2.05	0.10	5.07	50.00

6.4 Response Time Frequency Distribution Curves

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

Figure 6.1: New Order Response Time Distribution

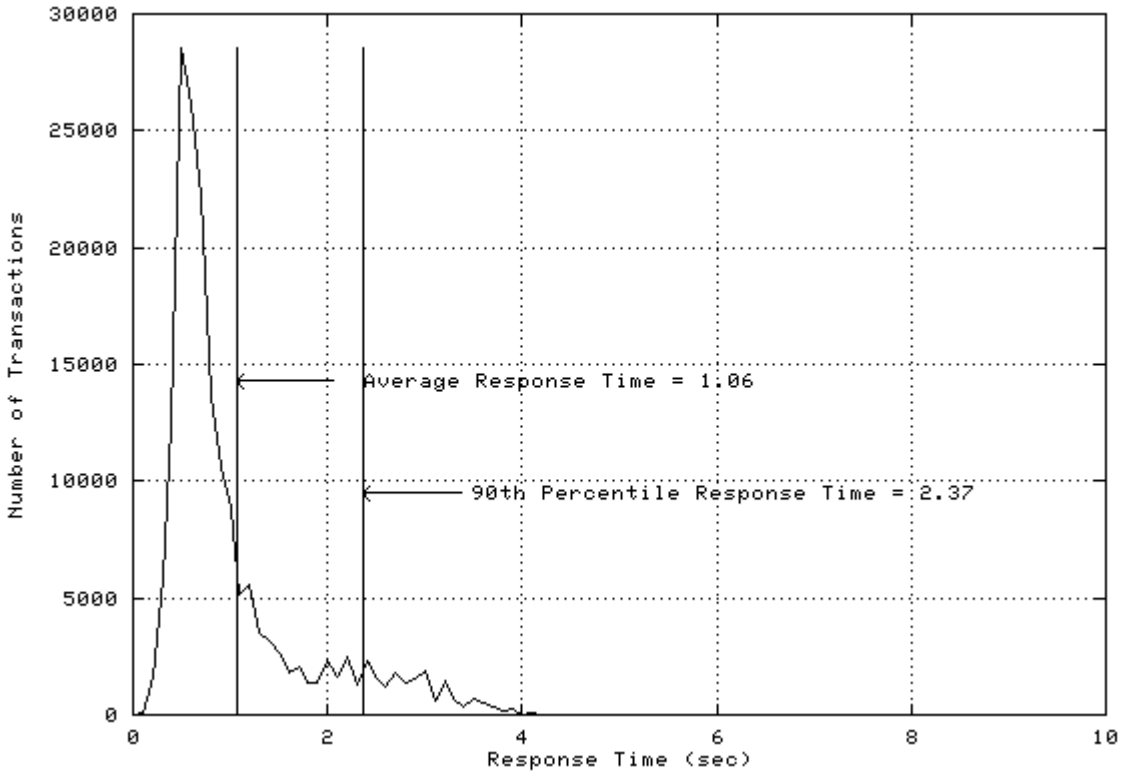


Figure 6.2: Payment Response Time Distribution

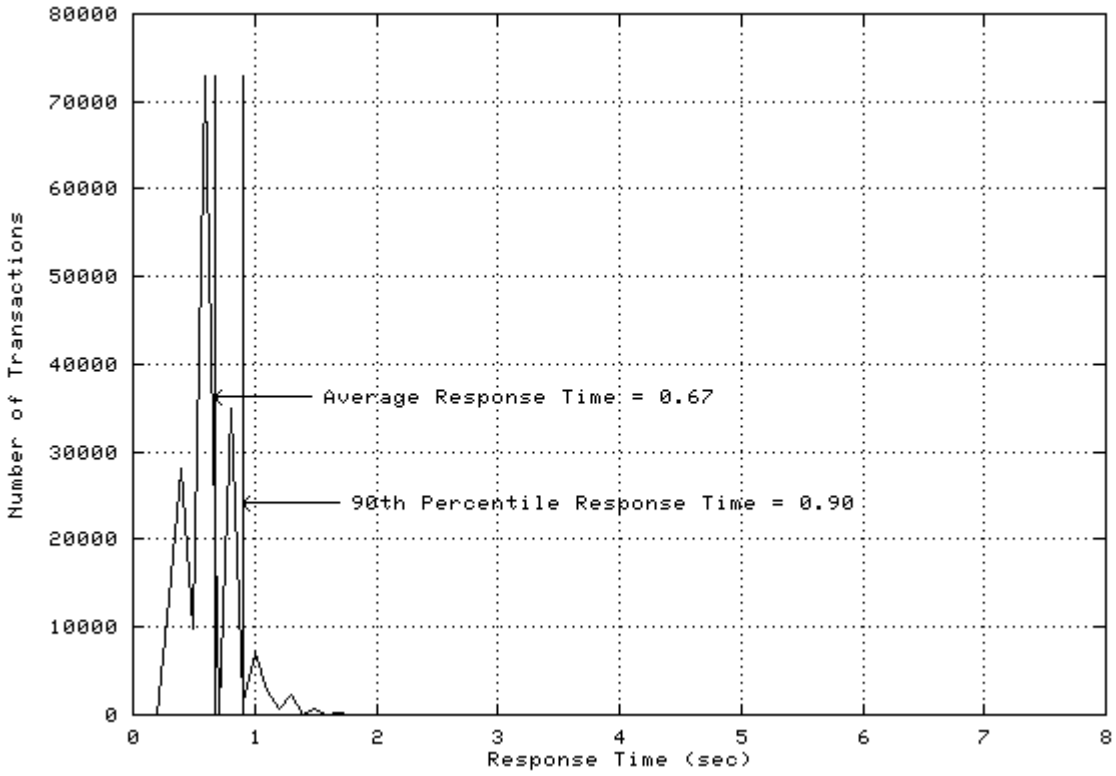


Figure 6.3: Order Status Response Time Distribution

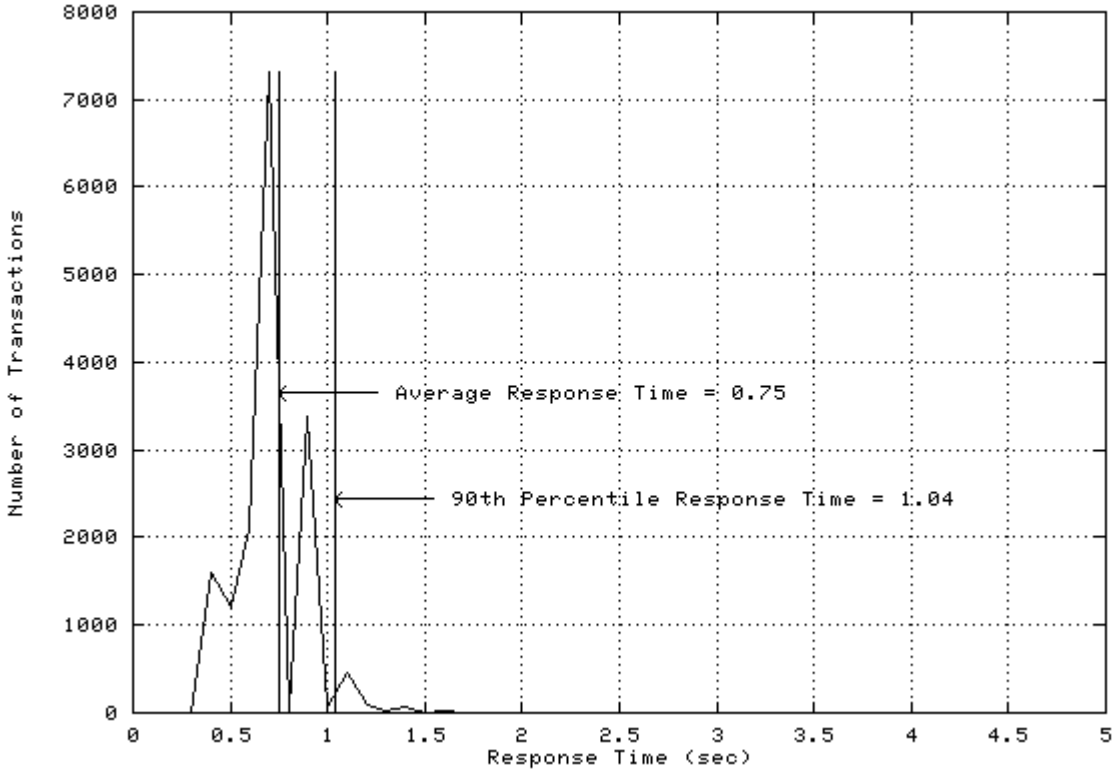


Figure 6.4: Delivery Response Time Distribution

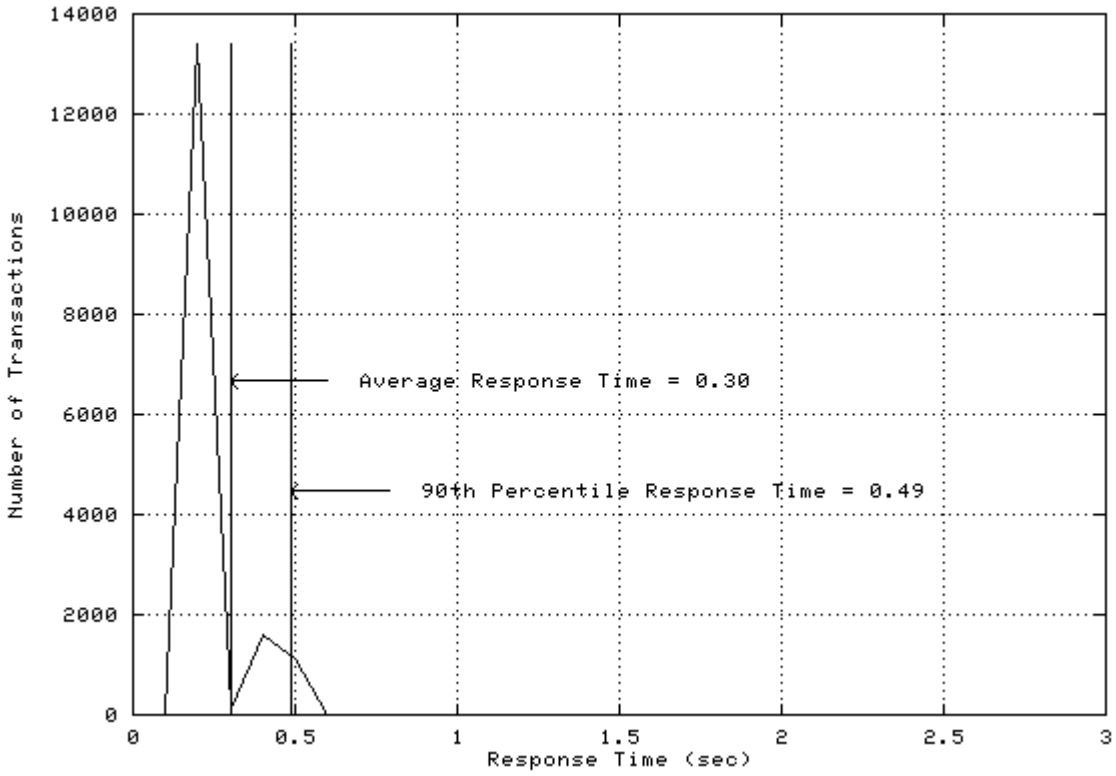
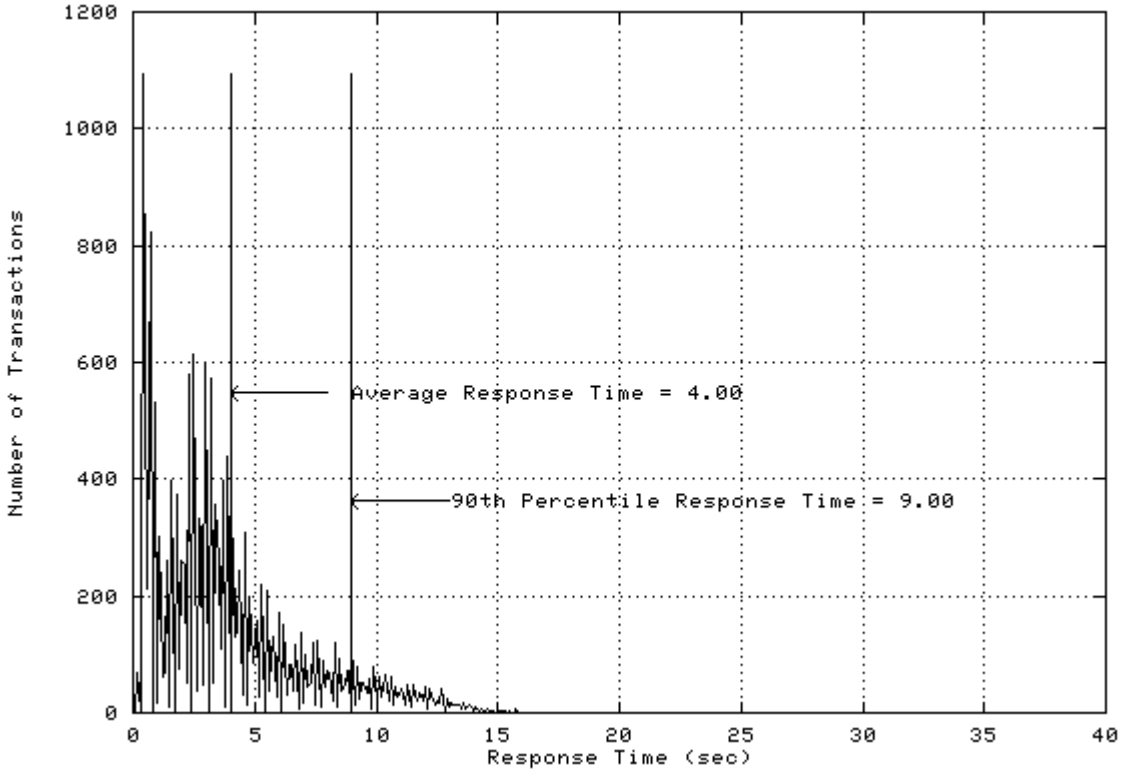


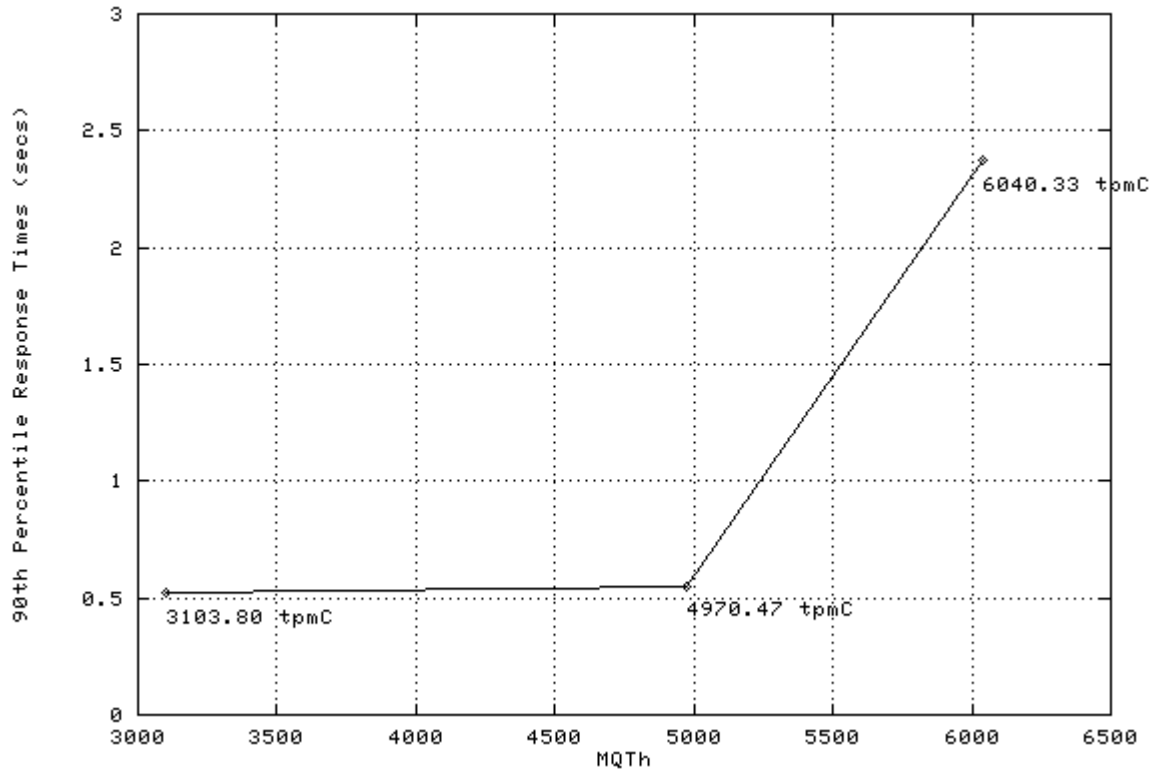
Figure 6.5: Stock Level Response Time Distribution



6.5 Response Time Vs. Throughput

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

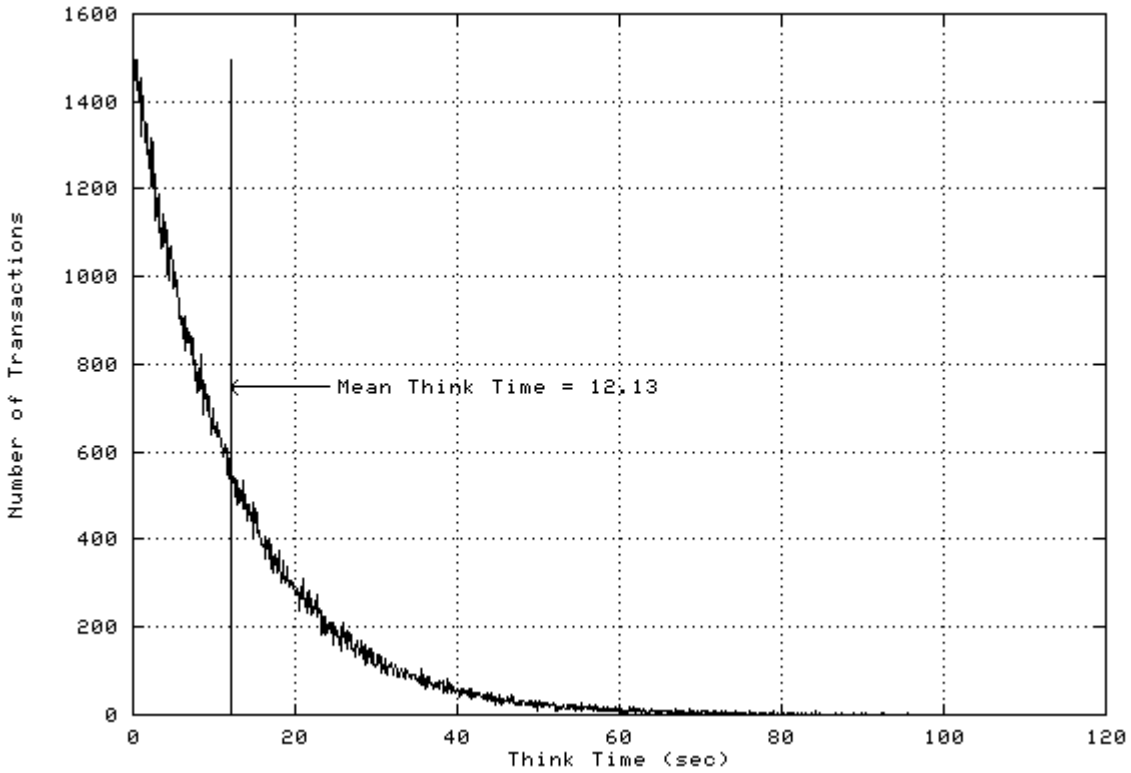
Figure 6.6: Response Time Vs. Throughput



6.6 Think Time Frequency Distribution Curves

A Think Time frequency distribution curve (see Clause 5.6.3) must be reported for the New-Order transaction.

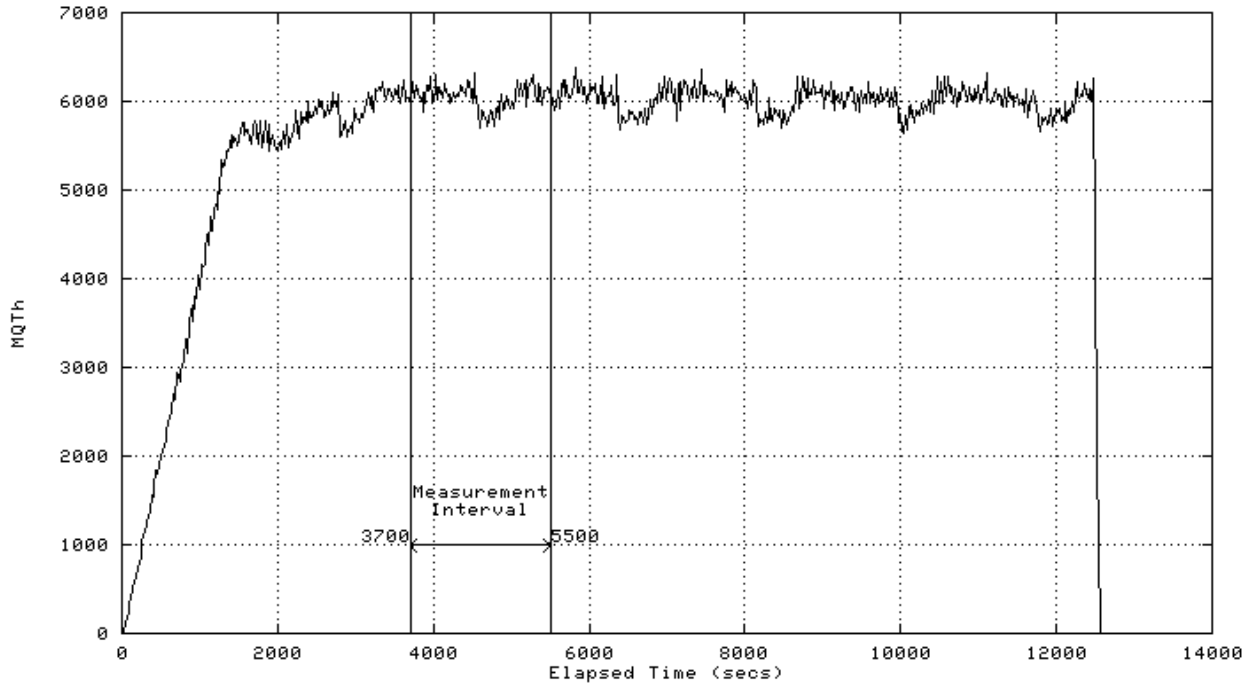
Figure 6.7: New Order Think Time Distribution



6.7 New-Order Throughput Vs. Time

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

Figure 6.8: Throughput Vs. Time



6.8 Steady State Determination

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

The transaction throughput rate (tpmC) and response times were relatively constant after the initial ramp up period. The throughput and response time were verified by examining the throughput (tpmC) graph reported at 20-second intervals for the duration of the benchmark. Ramp up, steady state, and ramp down are discernible in the graph presented in section 6.7.

6.9 Work Performed During Steady State

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

A Sybase Adaptive Server Enterprise 11.5.1 checkpoint writes all buffers in memory to disk so that data on disk matches what is in memory. Checkpoints are marked by a special record written into the logs. One checkpoint was implemented in the measurement run.

6.10 Reproducibility

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

In a second measurement, a throughput of 6018.17 tpmC was achieved.

6.11 Measurement Period Duration

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

The measurement interval was 30 minutes.

6.12 Transaction Mix Regulation

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

The weighted distribution algorithm as described in Clause 5.2.4.1 of the TPC-C specification was used to regulate the transaction mix. The weights were not adjusted during the run.

6.13 Transaction Mix

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

See Table 3.1.

6.14 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 transaction 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.

6.15 Checkpoints

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

One checkpoint occurred at 2790 seconds after the start of ramp-up and one at 880 seconds after the start of the measurement interval. The interval between the two checkpoints was 30 minutes.

Clause 6 Related Items

7.1 Remote Terminal Emulator 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.

Appendix E contains the portions of the RTE scripts that select the transactions and generate the transaction input data.

7.2 Emulated Components

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

There were no emulated components in the benchmark configuration other than the emulated users' workstations.

7.3 Configuration Diagrams

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

Figure 1.1 is a schematic of the benchmarked configuration and Figure 1.2 shows a schematic of the priced configuration. A description of the RTE and benchmark software is provided above.

7.4 Network Configuration

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

Figure 1.1 and Figure 1.2 show the network configurations of the benchmark and priced systems, and represent the Driver connected via LAN replacing the workstations that are directly connected via LAN.

7.5 Network Bandwidth

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

10BaseT and 100BaseT local area networks (LAN) were used in the benchmarked and priced systems as shown in configuration Figure 1.1 and Figure 1.2.

7.6 Operator Intervention

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

The tested configuration does not require operator intervention.

Clause 7 Pricing Related Items

8.1 Priced System

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

A detailed list of the hardware and software used in the priced system is included in the pricing sheet in the executive summary.

8.2 Total Five Year Price

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

The total 5-year price of the configuration is $736,822.

A detailed price sheet of all the hardware and software used in this configuration and the 5-year maintenance cost is included in the executive summary at the beginning of this document.

8.3 Availability Date

The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability

dates, the reported availability date for the priced system must be the date at which all components are committed to be available. This single date must be reported on the first page of the Executive Summary. All availability dates, whether for individual components or for the SUT as a whole, must be disclosed to a precision of one day.

All components are available as of the date reported in the executive summary.

8.4 Measured tpmC and Price/tpmC

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

The Maximum Qualified Throughput for the Silicon Graphics Origin2000™ Server was 6040.33 tpmC at $121.98 per tpmC. The availability date is reported in the executive summary.

Clause 9 Related Items

9.1 Auditor's Report

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

This implementation of the TPC Benchmark C was audited by:

Richard L. Gimarc
Performance Metrics, Inc.
2229 Benita Dr. Suite 101
Rancho Cordova, CA 95670.
Phone: (916) 635 2822
Fax: (916) 858-0109

The auditor's letter of attestation is included below:

PERFORMANCE METRICS INC.
TPC Certified Auditors

Jeff McDonald
System Performance Manager
Silicon Graphics Computer Systems
2011 N. Shoreline Blvd.
Mountain View, CA 94039

April 21, 1998

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

Platform: Silicon Graphics Origin2000 Server
Database Manager: Sybase Adaptive Server Enterprise 11.5.1
Operating System: IRIX 6.5
Transaction Manager: BEA Tuxedo 6.3 CFS

CPUs	Memory	Disks	New-Order Response Time @ 90%	tpmC
Server: Silicon Graphics Origin2000 Server				
2 MIPS R10000 @ 250 MHz	Main: 2 GB I-Cache: 32 KB D-Cache: 32 KB L2 Cache: 4 MB	81 @ 8.4 GB	2.37 sec.	6,040.33
2 Clients: Silicon Graphics Origin200 Server				
2 MIPS R10000 @ 180 MHz	Main: 2 GB I-Cache: 32 KB D-Cache: 32 KB L2 Cache: 1 MB	1 @ 4.2 GB	n/a	n/a

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, including error handling.
- The database files were properly sized and populated.
- The database was properly scaled with 505 warehouses.
- The ACID properties were met.
- The durability data loss and log loss tests were performed on a 10-warehouse database.

2229 Benita Dr., Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 Fax: (916) 858-0109 e-mail: Richard@PerfMetrics.com

- Input data was generated according to the specified percentages.
- Eight hours of mirrored log space was configured on the measured system.
- Data for the 180-day space calculation was verified. The measured system had sufficient storage space to satisfy this requirement.
- Measurement cycle times did not include any delays for emulated components.
- The steady state portion of the test was 30 minutes.
- One checkpoint was performed during the steady state portion of the test.
- Checkpoints were verified to be clear of the guard zones.
- There were 5,050 user contexts present on the system.
- Each emulated user had a different random number stream.
- The NURand constants used for database load and at run time were verified.
- System pricing was checked for major components and maintenance.

Additional Audit Notes: none

Regards,

Richard L. Gimarc

Richard L. Gimarc
Auditor

2229 Benita Dr., Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 Fax: (916) 858-0109 e-mail: Richard@PerfMetrics.com

Appendix A: Application Code

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

```
============================
Makefile.TMClient
============================

TUXEDO=/usr/people/sybase/tpcc-
kit/workdir/tuxedo/TUXEDO
SYBASE=/usr/sybase/11
WORKDIR=/usr/people/sybase/tpcc-kit/workdir/
INCLUDES = -I$(SYBASE)/include/ -
I$(TUXEDO)/include/
#CFLAGS = -32 $(INCLUDES) -DDEBUG
CFLAGS = -32 $(INCLUDES)
LIBS = -L$(SYBASE)/lib/ -L$(TUXEDO)/lib/ -ltux -
ltux2 -lfml
LIBS = -L$(SYBASE)/lib -L$(TUXEDO)/lib/ \
                -lsocket -ltux -ltux2 -lgp -lfml -
lfml32 -lsybdb
LDFLAGS = -32
PROF =
FOBJS =
# Use the following line for menu-driven
interactive queries
COBJS = clientlog.o client.o screen.o monitor.o
OBJS = $(COBJS)

all: client
client:    $(OBJS)
        $(TUXEDO)/bin/buildclient -v -o
$(WORKDIR)/bin/$@ \
        -f -32 -f -I$(SYBASE)/include \
        -f clientlog.o -f monitor.o -f client.o -f
screen.o \
        -f -L$(TUXEDO)/lib/ \
        -f -ltux -f -lsocket -f -ltux2 -f -lfml

============================
Makefile.TMServer
============================

TUXEDO=/usr/people/sybase/tpcc-
kit/workdir/tuxedo/TUXEDO
SYBASE=/usr/sybase/11
WORKDIR=/usr/people/sybase/tpcc-kit/workdir/
INCLUDES = -I$(SYBASE)/include/ -
I$(TUXEDO)/include/
#CFLAGS = -32 $(INCLUDES) -DDEBUG
CFLAGS = -32 $(INCLUDES)
LIBS = -L$(SYBASE)/lib/ -L$(TUXEDO)/lib/ -ltux -
ltux2 -lfml
LIBS = -L$(SYBASE)/lib -L$(TUXEDO)/lib/ \
                -lsocket -ltux -ltux2 -lgp -lfml -
lfml32 -lsybdb
LDFLAGS = -32
PROF =
FOBJS =

all: newo_srv ords_srv paym_srv stkl_srv dely_srv

newo_srv: newo_srv.o shared_tux_var.o
shared_rpc_var.o rpc.o error.o random.o
        $(TUXEDO)/bin/buildserver -v \
        -s NEWO -o $(WORKDIR)/bin/newo \
        -f -32 -f -I$(SYBASE)/include \
        -f newo_srv.o -f shared_tux_var.o \
        -f shared_rpc_var.o -f rpc.o -f error.o -f
random.o \
        -f -L$(SYBASE)/lib/ \
        -f -lsybdb

ords_srv: ords_srv.o shared_tux_var.o
shared_rpc_var.o rpc.o error.o random.o
        $(TUXEDO)/bin/buildserver -v \
        -s ORDS -o $(WORKDIR)/bin/ords \
        -f -32 -f -I$(SYBASE)/include \
        -f ords_srv.o -f shared_tux_var.o \
        -f shared_rpc_var.o -f rpc.o -f error.o -f
random.o \
        -f -L$(SYBASE)/lib/ \
        -f -lsybdb

paym_srv: paym_srv.o shared_tux_var.o
shared_rpc_var.o rpc.o error.o random.o
        $(TUXEDO)/bin/buildserver -v \
        -s PAYM -o $(WORKDIR)/bin/paym \
        -f -32 -f -I$(SYBASE)/include \
        -f paym_srv.o -f shared_tux_var.o \
        -f shared_rpc_var.o -f rpc.o -f error.o -f
random.o \
        -f -L$(SYBASE)/lib/ \
        -f -lsybdb

stkl_srv: stkl_srv.o shared_tux_var.o
shared_rpc_var.o rpc.o error.o random.o
        $(TUXEDO)/bin/buildserver -v \
        -s STOCK -o $(WORKDIR)/bin/stockl \
        -f -32 -f -I$(SYBASE)/include \
        -f stkl_srv.o -f shared_tux_var.o \
        -f shared_rpc_var.o -f rpc.o -f error.o -f
random.o \
        -f -L$(SYBASE)/lib/ \
        -f -lsybdb

dely_srv: dely_srv.o shared_tux_var.o
shared_rpc_var.o rpc.o error.o random.o
        $(TUXEDO)/bin/buildserver -v \
        -s DEL -o $(WORKDIR)/bin/dely \
        -f -32 -f -I$(SYBASE)/include \
        -f dely_srv.o -f shared_tux_var.o \
        -f shared_rpc_var.o -f rpc.o -f error.o -f
random.o \
        -f -L$(SYBASE)/lib/ \
        -f -lsybdb


============================
client.c
============================

/**
 ** client.c: Master control
 **/
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/procset.h>
#include <sys/param.h>
#include <limits.h>
#include <errno.h>
#include <stdlib.h>
#include <errno.h>
#include "client.h"

main(int argc, char **argv)
{
  int menu_selection;
  void do_transaction(int);

  if(argc != 2)  /** At least pass in the user-id
first! **/
    {
      fprintf(stderr, "Usage: %s <user-id>\n",
argv[0]);
      exit(1);
    }
```

```
  initialize(argv);
  Send_Menu();
  while ((menu_selection = sel_trans()) != 9) {
    if ((menu_selection < 1) || (menu_selection >
5))
      continue;
    do_transaction(menu_selection - 1);
    Send_Menu();
  }
  rundown();
}
initialize(char **argv)
{
  int menu_selection, start, m, n;
  char list[] =

"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ"
    ;
  tty_in = 0;
  tty_out = 1;
  if (Init_Monitor()) {
    fprintf(stderr, "\033[24;1H\033[mUnable to
connect to TP Monitor\n\1");
    exit(1);
  }
  get_wd(argv);
  set_display();
}
rundown()
{
  restore_terminal();
  Rundown_Monitor();
}
get_wd(char **argv)
{

  setup_wd(argv);

/*  display_screen(num);
  get_inputs(num);
*/
}
void
do_transaction(int num)
{
  int status;
  char c;
  FILE *fp;

  display_screen(num);
  status = get_inputs(num);
  if (status == 3)
    return;
#ifdef DEBUG
  fp = fopen("/usr/tmp/MYLOG", "a");
  fprintf(fp, "In <do_transaction()>:: Value of
status was: %d\n", status);
  fclose(fp);
#endif
  if ( Snd_Txn_To_Monitor(num) ){
    cleanup("\033[24;1H\033[mTransaction error
occured");
  }
  else
    display_output(num);
}
```

```
============================
client.h
============================

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

extern int tty_in;
extern int tty_out;

#define BOOLEAN int
#define LINEMAX 256
#define NEWORDER 0
#define PAYMENT 1
#define ORDSTAT 2
#define DELIVERY 3
#define STOCKLEV 4
#define WD 5
#define MAX_OL 15
#define TPM_ERROR 1
char tty_name[11];


============================
clientlog.c
============================

/************************************************
************************
clientlog.c
************************************************
************************/
/*
* ** clientlog.c -- Routine for writing out
messages form client processes - *
* useful for detailed error reporting and for
debugging
*/
#include <stdio.h>
#include <stdarg.h>
#define BACKTAB 2/** CTRL B **/
#define DELETE 127
#define ESCAPE 27
#define LF 10
#define QUIT 3 /** CNTRL-C **/
#define SPACE 32
#define SUBMIT 13/** CR **/
#define TAB 9
#define RTE_SYNCH_CHARACTER '\1'
static FILE *clientlog;
static int Clog_open = 0;
void
Clog(char *fmt, ...)
{
        va_list ap;
        va_start(ap, fmt);
        if (!Clog_open) {
                char fname[100];
                sprintf(fname, "%s/%s.%d",
getenv("TMPDIR"),
                        "CLIENTLOG", getpid());
                clientlog = fopen(fname, "w");
                Clog_open = 1;
        }

        fprintf(clientlog,fmt,ap);
        fflush(clientlog);
}

void
SCREENlog(int *flag, char *screen)
{
char fname[100];
int i, char_ct;
if (!Clog_open) {
sprintf(fname, "%s/%s.%d", getenv("TMPDIR"),
"CLIENTLOG",
getpid());
clientlog = fopen(fname, "w");
Clog_open = 1;
}
fprintf(clientlog, "** %d **\n", flag);
char_ct = 0;
fprintf(clientlog, "SCR: ");
for (i = 0; screen[i] != 0; char_ct++, i++) {
switch (screen[i]) {
```

```
case BACKTAB:
fprintf(clientlog, "<BACKTAB>");
break;
case DELETE:
fprintf(clientlog, "<DEL>");
break;
case ESCAPE:
fprintf(clientlog, "<ESC>");
break;
case LF:
fprintf(clientlog, "<LF>");
break;
case QUIT:
fprintf(clientlog, "<^C>");
break;
case SUBMIT:
fprintf(clientlog, "<CR>");
break;
case TAB:
fprintf(clientlog, "<TAB>");
break;
case RTE_SYNCH_CHARACTER:
fprintf(clientlog, "<^A>");
break;
default:
fprintf(clientlog, "%c", screen[i]);
}
if (char_ct > 192) {
char_ct = 0;
/* fprintf(screenlog,"\n"); */
}
}
fprintf(clientlog, "\n");
fflush(clientlog);
}
void
syserr(msg) /* print system call error message and
* terminate */
char *msg;
{
extern int errno, sys_nerr;
extern char *sys_errlist[];
extern char tty_name[];
fprintf(stderr, "\007ERROR: (%s) %s (%d", tty_name,
msg, errno);
if (errno > 0 && errno < sys_nerr)
fprintf(stderr, ";%s)\n", sys_errlist[errno]);
else
fprintf(stderr, ")\n");
exit(1);
}
void
cleanup(msg) /* print system call error message and
* terminate */
char *msg;
{
extern int tty_out;
extern int tty_in;
char c;
write(tty_out,msg,strlen(msg));
read(tty_in, &c, 1);
}

============================
dely_srv.c
============================

/**
 ** dely_srv.c : TUXEDO server for SYBASE delivery
Tx.
 **
 **/
#include <stdlib.h>
#include <signal.h>
#include <sys/utsname.h>
#include <errno.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>

/* SYBASE includes */
#include <sybfront.h>
#include <sybdb.h>
#include "tpcc.h"

/* TUXEDO includes */
#include "tuxstructs.h"

DBPROCESS *dbproc;
LOGINREC *login;

/**
 ** Setup for communication with DB.
 **/
int
dely_init()
{

  /* Install the error and message handler */
  dberrhandle(err_handler);
  dbmsghandle(msg_handler);

  /* Global Vars: deadlock and error handling */
  deadlock = 0;
  xact_type = XACT_BKEND;
  login = dblogin();
  DBSETLUSER(login, USER);
  DBSETLPACKET(login, 4096);
  DBSETLCHARSET(login, getenv("CHARSET"));

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

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

  return(0);
}

/**
 ** Parses the input 'req_struct' input from
tuxedo.
 ** Sets up the shared_var for rpc with DB.
 ** Returns shared_var results from rpc into
 **   the struct sent back to Tuxedo.
 **/
delivery_tx(tuxin)
TPSVCINFO *tuxin;
{
  struct timeval endq_t;

  /** Get the inputs into the rpc shared vars **/
  DelY = (struct req_struct *)(tuxin->data);
  w_id = DelY->w_id;
  o_carrier_id = DelY->o_carrier_id;
  tx_count++;
  sprintf(outbuf, "Starting transaction %d queued
at %d:%d\n",
          tx_count, DelY->startq_sec, DelY-
>startq_usec);

  /** Query the DB **/
  delivery_rpc();

  /** Log the return from the rpc **/
  gettimeofday(&endq_t);
  sprintf(outbuf+strlen(outbuf), "Transaction
completed at %d:%d\n",
          endq_t.tv_sec, endq_t.tv_usec);
```

```
  fwrite(outbuf, strlen(outbuf), 1, DelYfp);
  fflush(DelYfp);

  /** Finally, send it all back **/
  tpreturn(TPSUCCESS, 0, tuxin->data, sizeof(struct
req_struct), 0);
}

/* Tuxedo Prologue */
tpsvrinit(argc, argv)
  char **argv;
{
  char *p;
  char DelYLog[400];
  int SrvNo;
  struct utsname name;

  /* ServerNo used as unique identifier for log */
  SrvNo = atoi(argv[optind]);

  /* User Hostname + ServerNo. to name log */
  uname( &name);
  strcpy(DelYLog, "/usr/tmp");
  sprintf(DelYLog+strlen(DelYLog), "/%s.del%d",
name.nodename, SrvNo);
  userlog("Delivery Log = %s \n",DelYLog);

  /* Set up the delivery log for use */
  DelYfp = fopen(DelYLog, "w");
  if (DelYfp == NULL)
    userlog("Cannot create file %s\n", DelYLog);

  return(dely_init());
}

/** Tuxedo Epilogue **/
void
tpsvrdone()
{
  fclose(DelYfp);
  dbexit();
}

/** Service advertized by Tuxedo **/
DEL(tuxin)
  TPSVCINFO *tuxin;
{
  delivery_tx(tuxin);
}


/* TM Server: Tuxedo include files. Mostly for
message logging */
#include "atmi.h"
#include "userlog.h"

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

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

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


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

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

        /* Is this a deadlock message */
        if (msgno == 1205)
        {
                /* Set the deadlock indicator */
                /* *((DBBOOL *)
dbgetuserdata(dbproc)) = TRUE; */
                display_xction(msgtext);
                deadlock = 1;
                return(SUCCEED);
        }
        else if (msgno==0)
        {
                userlog("Message #%d - %s\n", msgno,
msgtext);
                userlog("Txn_type: %d Deadlock =
%d\n", xact_type, deadlock);
                return(SUCCEED);
        }
        else
        {
                userlog("Message #%d - %s\n", msgno,
msgtext);
                userlog("Txn_type: %d Deadlock =
%d\n", xact_type, deadlock);
                return(FAIL);
        }

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

============================
monitor.c
============================

/**
 ** monitor.c : General interface to and from the
TUXEDO TM.
 **/
#include <stdio.h>
#include <stdarg.h>
#include <atmi.h>
#include "monitor.h"
const char *svc_names[] = {"NEWO", "PAYM", "ORDS",
"DEL",
                           "STOCK"};
int
Snd_Txn_To_Monitor(int txn_type)
{
  int status;
  if (txn_type == DELIVERY) {
    if ( tpacall((char *)svc_names[txn_type],
tuxibuf, ilen,
                 TPNOREPLY) == -1){
#ifdef DEBUG
        Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
        svc_names[txn_type], tpstrerror(tperrno));
#endif
      return (TPM_ERROR);
    }
    return(0);
  } else {
```

```
    if ( tpcall((char *)svc_names[txn_type], (char
*)tuxibuf, ilen,
                &tuxobuf, &olen, 0) == -1){
      return (TPM_ERROR);
    }
    /* return user-defined failures */
    return (0);
  }
}
int Init_Monitor()
{
  char *text;
  ilen = sizeof(struct io_tpcc);
  olen = sizeof(struct io_tpcc);
  if (tpinit(NULL) == -1) {
    tpmerror("tpinit", tperrno);
    return -1;
  }
  if ((tuxibuf = tpalloc("CARRAY", NULL, ilen)) ==
NULL) {
    tpmerror("tpalloc", tperrno);
    return (-1);
  }
  if ((tuxobuf = tpalloc("CARRAY", NULL, ilen)) ==
NULL) {
    tpmerror("tpalloc", tperrno);
    return (-1);
  }

  /** Init the Server Statistics **/
  /* Tux_Init_Stats(); */

  return (NULL);
}
Rundown_Monitor()
{
  int status;
  tpfree(tuxibuf);
  status = tpterm();
#ifdef DEBUG
  Clog("terminated Tuxedo connection with status
%d\n", status);
#endif

  /** Collect and print out the Server Statistics
**/
  /* Tux_Print_Stats(); */

  return 0;
}
tpmerror(char *service_called, int errnum)
{
  char errmsg[256];
  fprintf(stderr, "\033[24;1H\033[mTUXEDO: Failed
%s with error: %s\n",
          service_called, tpstrerror(errnum));
  fprintf(stderr, "\n");
  return 0;
}
#ifdef DEBUG
print_input_data(int type)
{
  int i;
  time_t the_time;
  FILE *fp;


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

```
============================
monitor.h
============================

/**
 ** monitor.h: Vars and Defs. used in calls to TM
services
 **/
long ilen;
long olen;
int tty_in;
int tty_out;
char *tuxibuf;
char *tuxobuf;
extern void Clog(char *,...);

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

#define oNO (&((info_t *) tuxobuf)->neworder)
#define oPT (&((info_t *) tuxobuf)->payment)
#define oOS (&((info_t *) tuxobuf)->ordstat)
#define oDY (&((info_t *) tuxobuf)->delivery)
#define oSL (&((info_t *) tuxobuf)->stocklev)
#define iNO (&((info_t *) tuxibuf)->neworder)
#define iPT (&((info_t *) tuxibuf)->payment)
```

```
#define iOS (&((info_t *) tuxibuf)->ordstat)
#define iDY (&((info_t *) tuxibuf)->delivery)
#define iSL (&((info_t *) tuxibuf)->stocklev)
#define iWD (&((info_t *) tuxibuf)->wd)

/** Tuxedo Related Structs. **/
/* For NEWO: */
struct no_itm_struct {
  int ol_i_id;
  int ol_supply_w_id;
  int ol_quantity;
  char i_name[25];
  int s_quantity;
  char brand[2];
  double i_price;
  double ol_amount;
};
struct no_struct{
  int w_id;
  int d_id;
  int c_id;
  int o_ol_cnt;
  int o_all_local;
  struct no_itm_struct o_ol[15];

  char o_entry_d[31];
  char c_last[17];
  char c_credit[3];
  double c_discount;
  double w_tax;
  double d_tax;
  int o_id;
  char status[25];
  double total;
};

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

  char h_date[20];
  char w_street_1[21];
  char w_street_2[21];
  char w_city[21];
  char w_state[3];
  char w_zip[11];
  char d_street_1[21];
  char d_street_2[21];
  char d_city[21];
  char d_state[3];
  char d_zip[11];
  char c_first[17];
  char c_middle[3];
  char c_street_1[21];
  char c_street_2[21];
  char c_city[21];
  char c_state[3];
  char c_zip[11];
  char c_phone[17];
  char c_since[11];
  char c_credit[3];
  double c_credit_lim;
  double c_discount;
  double c_balance;
  char c_data[201];
};

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

struct ord_struct {
  int ol_cnt;
  int w_id;
  int d_id;
  int c_id;
  char c_last[17];

  char c_first[17];
  char c_middle[3];
  double c_balance;
  int o_id;
  char o_entry_d[31];
  int o_carrier_id;
  struct ord_itm_struct s_ol[15];
};

/* For queueing the DLVY Tx. */
struct del_struct {
  int w_id;
  int o_carrier_id;
  int startq_sec;    /** Time the Tx **/
  int startq_usec;   /** was queued **/
  int status;
};

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

struct menu_struct {
  int w_id;
  int d_id;
};
typedef union info {
  struct no_struct neworder;
  struct pay_struct payment;
  struct ord_struct ordstat;
  struct del_struct delivery;
  struct stock_struct stocklev;
  struct menu_struct wd;
} info_t;
struct io_tpcc {
  int type;
  info_t info;
};



============================
newo_srv.c
============================

/**
 ** newo_srv.c : TUXEDO server for SYBASE new_order
Tx.
 **
 **/
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

/* SYBASE includes */
#include <sybfront.h>
#include <sybdb.h>
#include "tpcc.h"

/* TUXEDO includes */
#include "tuxstructs.h"

char blank_mesg[25] = " ";
DBPROCESS *dbproc;
```

```
LOGINREC *login;

/**
 **  Setup for communication with DB.
 **/
int
newo_init()
{

  /* Install the error and message handler */
  dberrhandle(err_handler);
  dbmsghandle(msg_handler);

  /* Initialize global variable for error handling
*/
  deadlock = 0;
  xact_type = XACT_NEWO;
  login = dblogin();
  DBSETLUSER(login, USER);
  DBSETLPACKET(login, 4096);
  DBSETLCHARSET(login, getenv("CHARSET"));

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

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

  return(0);
}

/**
 ** Parses the input 'new_inf' struct from tuxedo.
 ** Sets up the shared_var for rpc with DB.
 ** Returns shared_var results from rpc into
 **   struct sent back to Tuxedo.
 **/
newo(tuxin)
TPSVCINFO *tuxin;
{
  int i;
  int rollback = 0;
  int linecnt;
  int ret;
  struct items_inf *item_ptr;

  NewO = (struct newo_inf *)(tuxin->data);
  linecnt = NewO->o_ol_cnt;
  strncpy(NewO->status, blank_mesg, 24);

  /* read warehouse, customer */
  NewO->total = 0;
  w_id = NewO->w_id;
  d_id = NewO->d_id;
  c_id = NewO->c_id;
  o_ol_cnt = NewO->o_ol_cnt;
  o_all_local = 1; /* Checked below */

  /** Copy over the orderlines from tuxstruct to
rpcvar **/
  for (i = 0; i < (int)o_ol_cnt ; i++) {
    item_ptr = &NewO->n_items[i];
    ol[i].i_id = item_ptr->ol_i_id;
    ol[i].supply_w_id = item_ptr->ol_supply_w_id;
    ol[i].quantity = item_ptr->ol_quantity;
    if (ol[i].supply_w_id != w_id)
      o_all_local = 0; /* non-local order */
  }

  /** Query DB **/
  new_order_rpc();

  /** Overall order information **/
  NewO->w_tax = w_tax*100;
  NewO->d_tax = d_tax*100;
  NewO->o_id = o_id;
  strcpy(NewO->c_last, c_last);
  NewO->c_discount = c_discount * 100;
  strcpy(NewO->c_credit, c_credit);
  strcpy(NewO->o_entry_d, o_entry_d);

  /* pick up total amount */
  NewO->total = (total_amount) *
                     (1 - c_discount)*
                      (1+w_tax+d_tax);
  tpreturn(TPSUCCESS, 0, tuxin->data, sizeof(struct
newo_inf), 0);
}

/* Tuxedo Prologue */
int
tpsvrinit(argc, argv)
char **argv;
{
  return(newo_init());
}

/** Tuxedo Epilogue */
void
tpsvrdone()
{
  dbexit();
}

/** Service advertized by Tuxedo **/
NEWO(tuxin)
TPSVCINFO *tuxin;
{
  newo(tuxin);
}

============================
ords_srv.c
============================

/**
 ** ords_srv.c : TUXEDO server for SYBASE
orderstatus Tx.
 **
 **/
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

/* Sybase includes */
#include <sybfront.h>
#include <sybdb.h>
#include "tpcc.h"

/* TUXEDO includes */
#include "tuxstructs.h"

DBPROCESS *dbproc;
LOGINREC *login;

/**
 **  Setup for communication with DB.
 **/
int
ords_init()
{

  /* Install the error and message handler */
  dberrhandle(err_handler);
```

```
  dbmsghandle(msg_handler);

  /* Initialize global variable for error handling
*/
  deadlock = 0;
  login = dblogin();
  DBSETLUSER(login, USER);
  DBSETLPACKET(login, 4096);
  DBSETLCHARSET(login, getenv("CHARSET"));

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

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

  return(0);
}

/**
 ** Parses the input 'ord_inf' struct from tuxedo.
 ** Sets up the shared_var for rpc with DB.
 ** Returns shared_var results from rpc into
 **   struct sent back to Tuxedo.
 **/
ords(tuxin)
TPSVCINFO *tuxin;
{
  int byid;

  OrdS = (struct ord_inf *)(tuxin->data);

  /** Check if ords_byname or ords_byid **/
  if (OrdS->c_id == 0) {
    byid = FALSE;
    xact_type = XACT_ORDS_NAME;
  }
  else {
    byid = TRUE;
    xact_type = XACT_ORDS_ID;
  }

  /** rpc shared vars **/
  c_w_id = OrdS->w_id;
  c_d_id = OrdS->d_id;

  /** Query the DB **/
  if (!byid) {
    strcpy(c_last, OrdS->c_last);
    order_status_byname_rpc();
    OrdS->c_id = c_id;
  }
  else {
    c_id = OrdS->c_id;
    order_status_byid_rpc();
    strcpy(OrdS->c_last, c_last);
  }

  /** Now get the rpcvar info into the tuxstruct
**/
  strcpy(OrdS->c_first, c_first);
  strcpy(OrdS->c_middle, c_middle);
  strcpy(OrdS->c_last, c_last);
  OrdS->c_balance = c_balance/100;
  OrdS->o_id = (int)o_id;
  strcpy(OrdS->o_entry_d, o_entry_d);
  OrdS->o_carrier_id = o_carrier_id;

  /** Finally, send it all back **/
  tpreturn(TPSUCCESS, 0, tuxin->data, sizeof(struct
ord_inf), 0);
}

/* Tuxedo Prologue */
int
tpsvrinit(argc, argv)
char **argv;
{
  return(ords_init());
}

/** Tuxedo Epilogue */
void
tpsvrdone()
{
  dbexit();
}

/** Service advertized by Tuxedo **/
ORDS(tuxin)
TPSVCINFO *tuxin;
{
  ords(tuxin);
}

============================
paym_srv.c
============================

/**
 ** paym_srv.c : TUXEDO server for SYBASE payment
Tx.
 **
 **/
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

/* SYBASE includes */
#include <sybfront.h>
#include <sybdb.h>
#include "tpcc.h"

/* TUXEDO includes */
#include "tuxstructs.h"

DBPROCESS *dbproc;
LOGINREC *login;

/**
 ** Setup for communication with DB.
 **/
int
paym_init()
{

  /* Install the error and message handler */
  dberrhandle(err_handler);
  dbmsghandle(msg_handler);

  /* Initialize global variable for error handling
*/
  deadlock = 0;
  login = dblogin();
  DBSETLUSER(login, USER);
  DBSETLPACKET(login, 4096);
  DBSETLCHARSET(login, getenv("CHARSET"));

  /* Open a dbproc */
  if ((dbproc = dbopen(login, (char *)SERVER )) ==
NULL)
    {
      userlog("SQL ERROR in dbopen: Could not open
connection\n");
```

```
      return(-1);
    }

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

  return(0);
}

/**
 ** Parses the input 'pay_inf' struct from tuxedo.
 ** Sets up the shared_var for rpc with DB.
 ** Returns shared_var results from rpc into
 **   the struct sent back to Tuxedo.
 **/
paym(tuxin)
TPSVCINFO *tuxin;
{
  int byid;

  PayM = (struct pay_inf *)(tuxin->data);
  w_id = PayM->w_id;
  c_w_id = PayM->c_w_id;
  h_amount = PayM->h_amount;
  d_id = PayM->d_id;
  c_d_id = PayM->c_d_id;

  /** Check if paym_byname or paym_byid **/
  if (PayM->c_id == 0) {
    byid = FALSE;
    xact_type = XACT_PAYM_NAME;
  }
  else {
    byid = TRUE;
    xact_type = XACT_PAYM_ID;
  }

  /** Query the DB **/
  if (byid) {
    c_id = PayM->c_id;
    payment_byid_rpc();
  }
  else {
    strcpy(c_last, PayM->c_last);
    payment_byname_rpc();
    PayM->c_id = c_id;
  }

  /** Now get the rpcvar info into the tuxstruct
**/
  strcpy(PayM->h_date, h_date);
  strcpy(PayM->w_street_1, w_street_1);
  strcpy(PayM->w_street_2, w_street_2);
  strcpy(PayM->w_city, w_city);
  strcpy(PayM->w_state, w_state);
  strcpy(PayM->w_zip, w_zip);
  strcpy(PayM->d_street_1, d_street_1);
  strcpy(PayM->d_street_2, d_street_2);
  strcpy(PayM->d_city, d_city);
  strcpy(PayM->d_state, d_state);
  strcpy(PayM->d_zip, d_zip);
  strcpy(PayM->c_first, c_first);
  strcpy(PayM->c_middle, c_middle);
  strcpy(PayM->c_last, c_last);
  strcpy(PayM->c_street_1, c_street_1);
  strcpy(PayM->c_street_2, c_street_2);
  strcpy(PayM->c_city, c_city);
  strcpy(PayM->c_state, c_state);
  strcpy(PayM->c_zip, c_zip);
  strcpy(PayM->c_phone, c_phone);
  strcpy(PayM->c_since, c_since);
  strcpy(PayM->c_credit, c_credit);
  strcpy(PayM->c_data, c_data);
  PayM->c_credit_lim = c_credit_lim/100;
  PayM->c_discount = c_discount * 100;
  PayM->c_balance = c_balance/100;

  /** Finally, send it all back **/
  if ( ( *c_last == '\0' ) || ( PayM->c_id == 0 )
|| (PayM->d_state == NULL))
    tpreturn(TPFAIL, 0, tuxin->data, sizeof(struct
pay_inf), 0);
  else
    tpreturn(TPSUCCESS, 0, tuxin->data,
sizeof(struct pay_inf), 0);
}

/* Tuxedo Prologue */
int
tpsvrinit(argc, argv)
char **argv;
{
  return(paym_init());
}

/** Tuxedo Epilogue */
void
tpsvrdone()
{
  dbexit();
}

/** Service advertized by Tuxedo **/
PAYM(tuxin)
TPSVCINFO *tuxin;
{
  paym(tuxin);
}


============================
random.c
============================

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

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

#define Denominator          16807
#define MaxRandom       2147483647
#define Quotient            127773     /* MaxRandom
div Denominator */
```

```
#define Remainder             2836     /* MaxRandom
mod Denominator */

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

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

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

/*************************************************
**************************
*
*
* irand - returns a 32 bit integer pseudo random
number with a period of     *
*       1 to 2 ^ 32 - 1.
*
*
*
* parameters:
*
*       none.
*
*
*
* returns:
*
*       32 bit integer - defined as I32 ( see above
).                        *
*
*
* side effects:
*
*       seed get recomputed.
*
*************************************************
************************/
I32
irand()
{
        register I32   hi = Seed / Quotient;
        register I32   lo = Seed % Quotient;

        Seed = Denominator * lo - Remainder * hi;

        if (Seed <= 0) Seed += MaxRandom;

        return Seed;
}

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

```
============================
rpc.c
============================

/**
 ** rpc.c : RPCs to talk to database
 **/
#include <stdio.h>
#include <sybfront.h>
#include <sybdb.h>
#include "tpcc.h"
#include "tuxstructs.h"

/* date conversion routines */
DBDATETIME SYBDatTim;
DBDATETIME SYBDat;

/* V 3.3 Addition*/
int invalid_xact;

void
sybdate2datetime(DBDATETIME *date, char *datetime)
{
  DBDATEREC dbdate;
  dbdatecrack(NULL, &dbdate, date);

  sprintf(datetime,"%02d/%02d/%04d %02d:%02d:%02d",
    dbdate.datedmonth,
    dbdate.datemonth+1,
    dbdate.dateyear,
    dbdate.datehour,
    dbdate.dateminute,
    dbdate.datesecond);
}
void
sybdate2date(DBDATETIME * sybdate, char * date)
{
  DBDATEREC daterec;
  dbdatecrack(NULL, &daterec, sybdate);
  sprintf(date, "%02d/%02d/%04d",
          daterec.datedmonth,
          daterec.datemonth+1,
          daterec.dateyear);
}

void
new_order_rpc()
{
    int        try;

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

       deadlock = 0;
       total_amount = 0;
       if (new_order_body() != TRUE) break;
       dbcancel(dbproc);
       sleep_before_retry();
    }
    if (try >= MaxTries)
      {
       display_xction("Failed");
/*     tpreturn(TPFAIL, 0, (char *)NewOrqst->data,
NewOrqst->len, 0); */
      }

}

int
new_order_body()
{
    int i;
    DBINT retcode;
```

```
    struct items_inf *i_inf_ptr; /* Current Item
Pointer */

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

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

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

    for (i = 0; i < o_ol_cnt; i++)
    {
        if (dbresults(dbproc) != SUCCEED ||
deadlock)
            return TRUE;
        else
        {
            dbbind(dbproc, 1, NTBSTRINGBIND,
sizeof(i_name), i_name);
            dbbind(dbproc, 2, SMALLBIND,      0,
&s_quantity);
            dbbind(dbproc, 3, FLT8BIND,       0,
&i_price);
            dbbind(dbproc, 4, NTBSTRINGBIND,
sizeof(b_g), b_g);
            if (dbnextrow(dbproc) != REG_ROW) return
TRUE;
            if(*i_name == '\0')
            {
/*              display_xction("Invalid item in");
        */
              strcpy(NewO->status, "Item number is
not valid");
              bad_items++;
              commit_flag = FALSE;
            }
            if (dbcanquery(dbproc) != SUCCEED ||
deadlock) return TRUE;
        }
        if (dbhasretstat(dbproc))
        {
            if ((retcode = dbretstatus(dbproc)) == -
3)
            {
                deadlock = 1;
                display_xction("Deadlock victim:");
            }
            else if (retcode<0)
            {
                userlog( "Unknown return status
%d:", retcode);
                display_xction("");
            }
            return TRUE;
        }
        i_inf_ptr = &NewO->n_items[i];
        strcpy(i_inf_ptr->i_name, i_name);
        i_inf_ptr->i_price = i_price/100;
        i_inf_ptr->s_quantity = s_quantity;
        strcpy(i_inf_ptr->brand, b_g);
        i_inf_ptr->ol_amount = (i_price *
ol[i].quantity)/100;
        total_amount += i_inf_ptr->ol_amount; /*
Display ol_amount[i] (= rhs) */
    }

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

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

void
payment_byid_rpc ()
{
    int try;

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

        if (payment_byid_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (payment_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }
    if (try >= MaxTries)
      {
        display_xction("Failed");
/*      tpreturn(TPFAIL, 0, (char *)PayMrqst->data,
PayMrqst->len, 0); */
      }
}

int
payment_byid_begin ()
{
    deadlock = 0;
    dbrpcinit(dbproc, "payment_byid", 0);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,
&w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,
&c_w_id);
    dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1,
&h_amount);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1,
&d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1,
&c_d_id);
```

```
    dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1,
&c_id);
    return (dbrpcsend(dbproc) == SUCCEED ? FALSE :
TRUE);
}

void
payment_byname_rpc ()
{
    int try;

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

        if (payment_byname_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (payment_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }
    if (try >= MaxTries)
      {
        display_xction("Failed");
/*      tpreturn(TPFAIL, 0, (char *)PayMrqst->data,
PayMrqst->len, 0); */
      }
}

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

int
payment_end ()
{
    if (dbsqlok(dbproc) != SUCCEED) return TRUE;
    if (dbresults(dbproc) != SUCCEED || deadlock)
            return TRUE;
    else
    {
        dbbind(dbproc,  1, INTBIND,     0, &c_id);
        dbbind(dbproc,  2, NTBSTRINGBIND,
sizeof(c_last), c_last);
        dbbind(dbproc,  3, DATETIMEBIND,  0,
&SYBDatTim);
        dbbind(dbproc,  4, NTBSTRINGBIND,
sizeof(w_street_1), w_street_1);
        dbbind(dbproc,  5, NTBSTRINGBIND,
sizeof(w_street_2), w_street_2);
        dbbind(dbproc,  6, NTBSTRINGBIND,
sizeof(w_city), w_city);
        dbbind(dbproc,  7, NTBSTRINGBIND,
sizeof(w_state), w_state);
        dbbind(dbproc,  8, NTBSTRINGBIND,
sizeof(w_zip), w_zip);

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

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

    /* TPC-C V3.3 Error Checking for invalid input
data */
    if (dbretstatus(dbproc) == -6)
      {
        userlog( "This is an invalid
transaction\n");
        invalid_xact++;
      }
    return FALSE;
}

void
order_status_byid_rpc()
{
    int try;

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

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

```
            sleep_before_retry();
            continue;
        }
        break;
    }
    if (try >= MaxTries)
      {
       display_xction("Failed");
/*     tpreturn(TPFAIL, 0, (char *)OrdSrqst->data,
OrdSrqst->len, 0); */
      }
}

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

void
order_status_byname_rpc ()
{
    int try;

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

        if (order_status_byname_begin() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        if (order_status_end() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }
    if (try >= MaxTries)
      {
       display_xction("Failed");
/*     tpreturn(TPFAIL, 0, (char *)OrdSrqst->data,
OrdSrqst->len, 0); */
      }
}

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

int
order_status_end ()
{
    int NoOLCount;

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

    if (dbresults(dbproc) != SUCCEED || deadlock)
        return TRUE;
    else {
         /* V 3.3 Error Checking
          ** checks whether current command returned
any rows
          */
         if (dbrows(dbproc) != SUCCEED )
           invalid_xact++;
        dbbind(dbproc, 1, SMALLBIND,     0,
&ol_supply_w_id);
        dbbind(dbproc, 2, INTBIND,       0,
&ol_i_id);
        dbbind(dbproc, 3, SMALLBIND,     0,
&ol_quantity);
        dbbind(dbproc, 4, FLT8BIND,      0,
&ol_amount);
        dbbind(dbproc, 5, DATETIMEBIND,  0,
        &SYBDat);
        NoOLCount = 0;
        while ((code = dbnextrow(dbproc)) == REG_ROW
&& !deadlock)
        {
          /*
           ** Print order_line information on RTE
           */
          OrdS->o_items[NoOLCount].ol_supply_w_id =
ol_supply_w_id;
          OrdS->o_items[NoOLCount].ol_i_id =
ol_i_id;
          OrdS->o_items[NoOLCount].ol_quantity =
ol_quantity;
          OrdS->o_items[NoOLCount].ol_amount =
ol_amount/100;
          sybdate2date(&SYBDat, ol_delivery_d);
          strcpy(OrdS-
>o_items[NoOLCount].ol_delivery_d, ol_delivery_d);
          ++NoOLCount;
        }
        OrdS->item_cnt = NoOLCount;

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

    if (dbresults(dbproc) != SUCCEED || deadlock)
        return TRUE;
    else
    {
       /* V 3.3 Error Checking
       ** checks whether current command returned
any rows
       */
       if (dbrows(dbproc) != SUCCEED )
          invalid_xact++;

        dbbind(dbproc, 1, INTBIND,     0,
&c_id);
        dbbind(dbproc, 2, NTBSTRINGBIND,
sizeof(c_last), c_last);
        dbbind(dbproc, 3, NTBSTRINGBIND,
sizeof(c_first), c_first);
        dbbind(dbproc, 4, NTBSTRINGBIND,
sizeof(c_middle), c_middle);
        dbbind(dbproc, 5, FLT8BIND,      0,
&c_balance);
        dbbind(dbproc, 6, INTBIND,       0,
&o_id);
        dbbind(dbproc, 7, DATETIMEBIND,  0,
        &SYBDatTim);
        dbbind(dbproc, 8, SMALLBIND,     0,
&o_carrier_id);
        if (dbnextrow(dbproc) != REG_ROW ||
deadlock) return TRUE;
        if (dbcanquery(dbproc) != SUCCEED ||
deadlock) return TRUE;
         sybdate2datetime(&SYBDatTim, o_entry_d);
    }
```

```
    return FALSE;
}

void
delivery_rpc ()
{
/*
  Called by delivery processes.
*/
    int try;

    d_id = 1;
    for (try = 0; try < MaxTries; try++)
    {
       if (try > 0) display_xction("Resuming
Delivery");

        if (delivery_body() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }
    if (try >= MaxTries)
      {
        display_xction("Failed");
        fwrite(outbuf, strlen(outbuf), 1, DelYfp);
        fflush(DelYfp);
        tpreturn(TPFAIL, 0, (char *)DelYrqst->data,
DelYrqst->len, 0);
      }
}

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

    for (; d_id <= 10; d_id++)
    {
       if (dbresults(dbproc) != SUCCEED) return
TRUE;
       dbbind(dbproc, 1, INTBIND,       0, &o_id);
       if (dbnextrow(dbproc) != REG_ROW ||
deadlock) return TRUE;
       if (dbcanquery(dbproc) != SUCCEED ||
deadlock) return TRUE;
       if (dbhasretstat(dbproc) &&
dbretstatus(dbproc) != 0) return TRUE;
       /*
       ** Print delivery information
       */
       if (o_id == NULL)
         sprintf(outbuf+strlen(outbuf),
                 "Delivery transaction skipped for
District %d\n", d_id);
       else
         sprintf(outbuf+strlen(outbuf),
                 "Delivered Order %d for District
%d, Warehouse %d, Carrier %d\n",
                 o_id, d_id, w_id, o_carrier_id);
    }
    return FALSE;
}

void
stock_level_rpc()
{
    int try;

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

        if (stock_level_body() == TRUE)
        {
            dbcancel(dbproc);
            sleep_before_retry();
            continue;
        }
        break;
    }
    if (try >= MaxTries)
      {
        display_xction("Failed");
/*      tpreturn(TPFAIL, 0, (char *)StkLrqst->data,
StkLrqst->len, 0); */
      }
}

int
stock_level_body ()
{
    int dup, iid, unique[500], i, rowcount;

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

    if (dbresults(dbproc) != SUCCEED || deadlock)
return TRUE;
    dbbind(dbproc, 1, INTBIND,  0, &iid);
    /* sort to eliminate duplicates */
    rowcount = 0;
    while (dbnextrow(dbproc) == REG_ROW &&
!deadlock)
      {
       dup = 0;
       for (i=0; i<rowcount; i++)
       {
         if (iid == unique[i])
           {
             dup = 1;
             break;    /** Been there...continue
**/
           }
       }
      if (dup == 0)
       {
         if (rowcount >= 500)
           display_xction("Too many rows in sort
within");
         else
           unique[rowcount++] = iid;
       }
    }
    if (dbcanquery(dbproc) != SUCCEED || deadlock)
return TRUE;
    low_count = rowcount;
    return FALSE;
}

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

```
void
sleep_before_retry ()
{
    sleep(irand()%3+1);
    userlog( "called sleep_before_retry\n");
fflush(stderr); /* marcoz */
}


void
display_xction(msg) char * msg;
{
    int i;
    userlog("Client #%d: %s %s ", clientid, msg,
xction[xact_type].name);

    switch(xact_type)
    {

    case 0:    /* new_order */
       userlog("w=%d, d=%d, c=%d, %d lines: \n[",
                     w_id, d_id, c_id, o_ol_cnt);
       for (i=0; i<(int)o_ol_cnt; i++)
           userlog(" %d", ol[i].i_id);
       userlog("]\n");
       break;

    case 1:    /* payment_byid */
       userlog( "w=%d/%d, d=%d/%d, c=%d\n",
                     w_id, c_w_id, d_id/c_d_id,
c_id);
       break;

    case 2:    /* payment_byname */
       userlog( "w=%d/%d, d=%d/%d, l=%s\n",
                     w_id, c_w_id, d_id, c_d_id,
c_last);
       break;

    case 3:    /* order_status_byid */
       userlog( "cw=%d, cd=%d, c=%d\n", c_w_id,
c_d_id, c_id);
       break;

    case 4:    /* order_status_byname */
       userlog( "cw=%d, cd=%d, l=%s\n", c_w_id,
c_d_id, c_last);
       break;

    case 5:    /* delivery_qu */
       userlog( "w=%d, carrier=%d\n", w_id,
o_carrier_id);
       break;

    case 6:    /* stock level */
       userlog( "w=%d, d=%d, th=%d\n", w_id, d_id,
threshold);
       break;

    case 7:    /* delivery */
       userlog( "w=%d, d=%d, carrier=%d\n", w_id,
d_id, o_carrier_id);
       break;

    default:
       userlog( "Unknown xact_type\n");
    }
}

#ifdef SORT_LINES
sort_order_lines ()
{
/*
** Sort order_lines in a new_order by i_id. Reduces
possibility of deadlock.
** Brute force insertion sort -- works OK for <= 15
rows.
*/
       int            i, j;
       ORDER_LINE     temp;

       for (j=1; j<o_ol_cnt; j++) {
              if (ol[j-1].i_id > ol[j].i_id) {
                     temp = ol[j];
                     ol[j] = ol[j-1];
                     for (i=j-2; i>=0 && temp.i_id
< ol[i].i_id; i--) {
                            ol[i+1] = ol[i];
                     }
                     ol[i+1] = temp;
              }
       }
}
#endif

============================
screen.c
============================

/************************************************
************************
screen.c
************************************************
************************/
#include <stdio.h>
#include <sys/termio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include "client.h"
#include "screen.h"

static int screen_bufindex;
static char screen_buf[SCRBUF_LEN];
extern void Clog(char *,...);
extern void SCREENlog(int, char *);
const char blanks[1802] = " ";

void
setraw()
{
  /** put screen in raw mode **/
  extern struct termio tbufsave;
  struct termio tbuf;
  int status;
  if (ioctl(tty_in, TCGETA, &tbuf) == -1)
    return;
  tbufsave = tbuf;
  tbuf.c_iflag &= ~(INLCR | ICRNL | IUCLC | ISTRIP
| IXON |
                    BRKINT);
  tbuf.c_oflag &= ~OPOST;
  tbuf.c_lflag &= ~(ICANON | ISIG | ECHO);
  tbuf.c_cc[VMIN] = LEAVE_SCREEN_MIN;
  tbuf.c_cc[VTIME] = LEAVE_SCREEN_TIMEOUT;
  if (ioctl(tty_out, TCSETAF, &tbuf) == -2)
    syserr("ioctl_ERROR#2 - setting raw mode for
STDIN error");
}

void
restore_terminal()
{ /** restore terminal flags **/
  extern struct tbufsave;
  struct termio tbuf;
  int status;
  if (ioctl(tty_out, TCSETAF, &tbufsave) == -1)
    syserr("ioctl_ERROR#3 - restoring original
input terminal settings error");
  tbuf = tbufsave;
  if (ioctl(tty_out, TCSETAF, &tbuf) == -1)
    syserr("ioctl_ERROR#4 - Forcing the original
settings back for STDIN error");
}

int
sel_trans()
{
  int c, read_count;
```

```
  static char inbuf[2] = "\0\0";
  int i = 0;
  read_count = read(tty_in, inbuf, 1);
  if (read_count == 0)
    syserr("TTY lost connection");
  if (inbuf[0] == QUIT)
    return 9;
  switch (inbuf[0]) {
  case '1':
    c = 1; break;
  case '2':
    c = 2; break;
  case '3':
    c = 3; break;
  case '4':
    c = 4; break;
  case '5':
    c = 5; break;
  case '9':
    c = 9; break;
  }
  return c;
}
int newo_val(int *);
int paym_val(int *);
int ords_val(int *);
int del_val(int *);
int stock_val(int *);
int wd_val(int *);
int(*p_check_function[]) () = {
  &newo_val,
  &paym_val,
  &ords_val,
  &del_val,
  &stock_val,
  &wd_val
};
int
get_inputs(int txn_type)
{
  int done = FALSE;
  int i, returned_key;
  io_elem *ioptr;
  int last_input;
  FILE *fp;
  float float_h_amount = 0.0;
  memset(tuxibuf, '\0', sizeof(info_t ));
  int_h_amount = 0;
  last_input = Forms[txn_type].num_input_elems - 1;
#ifdef DEBUG
  fp = fopen("/usr/tmp/MYLOG", "a");
  fprintf(fp, "DBG: The value of <txn_type> was:
%d\n", txn_type);
  fprintf(fp, "DBG: The value of <num_input_elems>
was: %d\n", Forms[txn_type].num_input_elems);
  fprintf(fp, "DBG: The <last_input> value was:
%d\n", last_input);
  fclose(fp);
#endif
  i = 0;
  while (done == FALSE) {
    ioptr = &Forms[txn_type].input_elems[i];
    if (txn_type == PAYMENT){
      if (i == 5)
        payment_input = TRUE;
      else
        payment_input = FALSE;
    }
    returned_key = (ioptr->fptr) (ioptr->x, ioptr-
>y, ioptr->
                                  len,ioptr->flags,
ioptr->dptr);
    switch (returned_key) {
    case BACKTAB:
      if (i == 0)
        i = last_input;
      else
        i--;
      break;
    case TAB:
      if (i == last_input)
        i = 0;
      else
        i++;
      break;
    case QUIT:
      done = TRUE;
      break;
    case SUBMIT:
    case LF:
      if (screen_bufindex) {
        PAINTSCRLEN(screen_buf, screen_bufindex);
        screen_bufindex = 0;
      }
      payment_input = FALSE;
      done = (p_check_function[txn_type]) (&i);
      break;
    }
  }
  return returned_key;
}

int
newo_val(int *pos)
{
  int done = FALSE;
  struct no_itm_struct *ol_ptr;
  int blank_line = 0, i;
  iNO->w_id = w_id;
  if (iNO->d_id <= 0) {
    *pos = 0;
    message = TRUE;
    PAINTSCR(MANDATORY_MSG);
  } else if (iNO->c_id <= 0) {
    *pos = 1;
    message = TRUE;
    PAINTSCR(MANDATORY_MSG);
  } else {
    ol_ptr = iNO->o_ol;
    for (i = 0; i < MAX_OL; i++, ol_ptr++) {
      if (ol_ptr->ol_i_id || ol_ptr->ol_supply_w_id
          || ol_ptr->ol_quantity)
        {
          /* and is that data complete */
          if (ol_ptr->ol_i_id && ol_ptr-
>ol_supply_w_id
              && ol_ptr->ol_quantity)
            {
              if (blank_line == 0){
                iNO->o_ol_cnt++;
              }else{
                *pos = 2;
                PAINTSCR(INCOMPLINE_MSG);
                message = TRUE;
                iNO->o_ol_cnt = 0;
                return FALSE;
              }
            } else {
              *pos = 2 + 3 * i;
              PAINTSCR(INCOMPLINE_MSG);
              message = TRUE;
              iNO->o_ol_cnt = 0;
              return FALSE;
            }
        } else blank_line=1;
    }
    if (!iNO->o_ol_cnt) {
      *pos = 2;
      PAINTSCR(MANDATORY_MSG);
      message = TRUE;
      iNO->o_ol_cnt = 0;
      return FALSE;
    }
    done = TRUE;
  }
  return done;
}

int paym_val(int *pos)
{
```

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

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

int del_val(int *pos)
{
  struct timeval endq_t;

  int done = FALSE;
  iDY->w_id = w_id;
  if (iDY->o_carrier_id <= 0) {
    message = TRUE;
    PAINTSCR(MANDATORY_MSG);
  } else {

    gettimeofday(&endq_t);
    iDY->startq_sec = endq_t.tv_sec;
    iDY->startq_usec = endq_t.tv_usec;
    done = TRUE;
  }
  return done;
}

int stock_val(int *pos)
{
  int done = FALSE;
  iSL->w_id = w_id;
  iSL->d_id = d_id;
  if (iSL->threshold <= 0) {
    message = TRUE;
    PAINTSCR(MANDATORY_MSG);
  } else
    done = TRUE;
  return done;
}

int wd_val(int *pos)
{
  int done = FALSE;
  if (iWD->w_id == 0 || iWD->d_id == 0) {
    message = TRUE;
    PAINTSCR(MANDATORY_MSG);
  } else {
    w_id = iWD->w_id ;
    d_id = iWD->d_id;
    done = TRUE;
  }
  return done;
}

void setup_wd(char **argv)
{
  io_elem *p;
  char buf[128];
  void setup_io_elems();
  setraw();

  w_id = (atoi(argv[1])-1)/10 + 1;
  d_id = (atoi(argv[1])-1)%10 + 1;

/*  setup_screen_buffer(&Forms[5], 5);
  p = Forms[WD].input_elems;
  p++->dptr = &iWD->w_id;
  p++->dptr = &iWD->d_id;
*/  CLRSCN(buf);
  PAINTSCR(buf);
}

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

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

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

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

```
  p++->dptr = &iPT->c_d_id;
  p++->dptr = (int *) &iPT->c_last[0];
  p->dptr = &int_h_amount;
  p = Forms[ORDSTAT].input_elems;
  p++->dptr = &iOS->d_id;
  p++->dptr = &iOS->c_id;
  p->dptr = (int *) &iOS->c_last[0];
  p = Forms[DELIVERY].input_elems;
  p->dptr = &iDY->o_carrier_id;
  p = Forms[STOCKLEV].input_elems;
  p->dptr = &iSL->threshold;
}

int
setup_screen_buffer(struct form_info * form_ptr,
int txn_type)
{
  FILE *ifile;
  const text_elem *tbuf;
  char *bufp;
  int ct;
  char input_display_buf[64];
  io_elem *io_ptr;
  bufp = screen_buf;
  bufp += CLRSCN(bufp);
  tbuf = form_ptr->tp;
  while (tbuf->text) {
    bufp += DISPLAY(bufp, tbuf->y, tbuf->x, tbuf-
>text);
    tbuf++;
  }
  bufp += SWITCH_TO_UNDERL(bufp);
  ct = 0;
  for (io_ptr = form_ptr->input_elems; io_ptr->y !=
999; io_ptr++) {
    strncpy(input_display_buf, blanks, io_ptr-
>len);
    input_display_buf[io_ptr->len] = '\0';
    bufp += DISPLAY(bufp, io_ptr->x, io_ptr->y,
input_display_buf);
    ct++;
  }
  form_ptr->num_input_elems = ct;
  bufp += SWITCH_TO_NORMAL(bufp);
  if (txn_type == PAYMENT)
    bufp += DISPLAY_INT(bufp, 4, 12, 4, w_id);
  else if (txn_type != 5)
    bufp += DISPLAY_INT(bufp, 4, 12, 2, w_id);
  if (txn_type == STOCKLEV)
    bufp += DISPLAY_INT(bufp, 2, 29, 2, d_id);
  bufp += SWITCH_TO_UNDERL(bufp);
  *bufp++ = '\1';
  *bufp = '\0';
  form_ptr->blank_formlen = bufp - screen_buf + 1;
  if (!form_ptr->blank_form &&
      ((form_ptr->blank_form = malloc(form_ptr-
>blank_formlen)) == NULL))
    {
      Clog("setup_screen_buffer: malloc failed\n");
      exit(1);
    }
  memcpy(form_ptr->blank_form, screen_buf,
form_ptr->blank_formlen);
  memset(screen_buf, '\0', form_ptr-
>blank_formlen);
}

int
read_integer(col, row, size, flags, data)
  int col, row, size, flags, *data;
{
  int exit_read_function = FALSE,
previous_data_exists = FALSE;
  int return_status = TAB, bytes_read = 0, i = 0, j
= 0, k = 0,
    size1 = 0, cur_col = col;
  char *bufp, temp[50];
  float q;
  char erase_field[20];
  strncpy(temp, " " , 1);
  bufp = screen_buf + screen_bufindex;
  /* Position cursor at start of field */
  if (curbuf_read == read_count || curbuf_read ==
0) {
    screen_buf[0] = '\0';
    bufp += GOTOXY(bufp, col + size - 1, row);
    PAINTSCRLEN(screen_buf, bufp - screen_buf);
    bufp = screen_buf;
  }
  size1 = size;
  if (*data > 0)
    previous_data_exists = TRUE;
  while (exit_read_function == FALSE) {
    /*
     * Below we read from standard input into the
array curbuf.
     * curbuf_read is the pointer to the array
curbuf indicating
     * the position upto which the curbuf has been
parsed.
     * curbuf_consumed is the number of elements in
the buffer
     * temp that holds the array that is to be
displayed.
     * Elements of curbuf_consumed is selectively
copied from
     * curbuf Note:read_count is the total number
of characters
     * in the buffer curbuf. curbuf_read is always
less than or
     * equal to read_count.
     */
    if (curbuf_read == read_count || curbuf_read ==
0) {
      curbuf_read = 0;
      read_count = read(tty_in, curbuf,
sizeof(curbuf));
      if (read_count == 0)
        syserr("TTY lost connection");
    }
    /*
     * int message prevents unnecessary display of
warning
     * messages
     */
    if (message == TRUE) {
      bufp += DISPLAY(bufp, MESSAGE_COL,
                      MESSAGE_ROW, ERASE_MSG);
      message = FALSE;
    }
    if (previous_data_exists == TRUE) {
      if (curbuf[curbuf_read] == DELETE) {
        previous_data_exists = FALSE;
        strncpy(erase_field, blanks, size);
        erase_field[size] = '\0';
        bufp += DISPLAY(bufp, col, row,
erase_field);
        bufp += GOTOXY(bufp, col + size - 1, row);
      } else {
        if (curbuf[curbuf_read] < '0' ||
curbuf[curbuf_read]
            > '9') {
          exit_read_function = TRUE;
          previous_data_exists = FALSE;
          return_status = curbuf[curbuf_read];
          curbuf[curbuf_read] = '\0';
        } else {
          previous_data_exists = FALSE;
          strncpy(erase_field, blanks, size);
          erase_field[size] = '\0';
          bufp += DISPLAY(bufp, col, row,
erase_field);
          /*
           * bufp = screen_buf;
           */
        }
      }
    } /* if previous_data_exists */
    while ((curbuf_read < read_count) &&
(exit_read_function ==
```

```
                                        FALSE)) {
      /*
       * intermediate variable size1 for cases when
       * floating point field whose size is less
than
       * actual size by 1 because of decimal.
       */
      if (payment_input == TRUE)
        size1 = size - 1;
      /*
       * Test for integer
       */
      if ((curbuf[curbuf_read] >= '0' &&
curbuf[curbuf_read] <=
           '9') || (curbuf[curbuf_read] == '.')) {
        /*
         * Consume all integers in buffer
         */
        for (; curbuf_read < read_count &&
               ((curbuf[curbuf_read] >= '0'
                 && curbuf[curbuf_read] <= '9') ||
                curbuf[curbuf_read] == '.');
curbuf_read++) {
          /*
           * below we fill up temp making sure
           * the size limit is not exceeded
           */
          if (curbuf_consumed < size1) {
            temp[curbuf_consumed] =
              curbuf[curbuf_read];
            curbuf_consumed++;
          }
          /*
           * number of elements typed in is
           * more than the size of the field
           */
          else
            OVERFLOW = TRUE;
          /*
           * ensure the character is removed
           * after it is read
           */
          curbuf[curbuf_read] = '\0';
        } /* end of for curbuf is legitimate
           * number */
        temp[curbuf_consumed] = '\0';/* terminate
temp
                                      string */
        if (payment_input == TRUE) {/* floating
point field */
          /* convert the ascii to float */
          q = (atof(temp)) ;
          bufp += DISPLAY_FLOAT(bufp, 2, (col +
                                        size - 4),
row, q);
          /*
            if (curbuf_consumed < 3)
            else
            bufp += DISPLAY_FLOAT(bufp, 2, (col +
            size - curbuf_consumed - 1), row, q);
            */
        } else {
          if (curbuf_consumed < size + 1)
            bufp += DISPLAY(bufp, (col + size -
                                   curbuf_consumed),
row,
                            temp);
          return_status = curbuf[curbuf_read];
          cur_col++;
        }
      }
      /* if curbuf[] between "0" and "9" */
      /** if not integer, then test for movement
character
       */
      else if (curbuf[curbuf_read] == TAB
               || curbuf[curbuf_read] == LF
               || curbuf[curbuf_read] == BACKTAB
               || curbuf[curbuf_read] == SUBMIT) {
        if (message == TRUE) {
          bufp += DISPLAY(bufp, MESSAGE_COL,
                          MESSAGE_ROW, ERASE_MSG);
          message = FALSE;
        }
        temp[curbuf_consumed] = '\0';
        if (payment_input == TRUE) {
          q = atof(temp);
          *data = q*100;
        }
        else {
          *data = atoi(temp);
        }
        exit_read_function = TRUE;
        return_status = curbuf[curbuf_read];
        curbuf[curbuf_read] = '\0';
        curbuf_read++;
        curbuf_consumed = 0;
      }
      /* if curbuf[] a movement character */
      /*
       * if not integer of movement, test for
DELETE
       */
      else if (curbuf[curbuf_read] == DELETE) {
        if (payment_input == TRUE) {/* for floating
                                       point
                                       * field */
          if (curbuf_consumed != 0)
            curbuf_consumed--;
          if (message == TRUE) {
            bufp += DISPLAY(bufp,
                            MESSAGE_COL,
                            MESSAGE_ROW,
                            ERASE_MSG);
            message = FALSE;
          }
          OVERFLOW = FALSE;
          PAINTSCR(screen_buf);
          temp[curbuf_consumed] = '\0';
          q = atof(temp);
          curbuf[curbuf_read] = '\0';
          strncpy(erase_field, blanks, size);
          erase_field[size] = '\0';
          bufp = screen_buf;
          screen_bufindex = 0;
          bufp += DISPLAY(bufp, col, row,
                          erase_field);
          if (curbuf_consumed < 3)
            bufp += DISPLAY_FLOAT(bufp, 2, (col +
size - 4), row, q);
          else
            bufp += DISPLAY_FLOAT(bufp, 2,
                                  (col + size -
curbuf_consumed - 1), row, q);
          if (cur_col != 0)
            cur_col--;
          if (curbuf_read < 40)
            curbuf_read++;/* pressed key overflow
                           * situations */
          bufp += GOTOXY(bufp, col + size, row);
        } else {
          if (curbuf_consumed != 0)
            curbuf_consumed--;
          curbuf[curbuf_read] = '\0';
          curbuf_read++;
          if (message == TRUE) {
            bufp += DISPLAY(bufp, MESSAGE_COL,
                            MESSAGE_ROW,
                            ERASE_MSG);
            message = FALSE;
          }
          OVERFLOW = FALSE;
          PAINTSCR(screen_buf);
          temp[curbuf_consumed] = '\0';
          strncpy(erase_field, blanks, size);
          erase_field[size] = '\0';
          bufp = screen_buf;
          screen_bufindex = 0;
          bufp += DISPLAY(bufp, col, row,
erase_field);
```

```
          bufp += DISPLAY(bufp, (col + size -
                                 curbuf_consumed),
row, temp);
          if (cur_col != 0)
            cur_col--;
          bufp += GOTOXY(bufp, col + size, row);
        }
      }
      /* end of if DELETE */
      /* could be a ^C */
      else if (curbuf[curbuf_read] == QUIT) {
        temp[0] = '\0';
        return_status = QUIT;
        curbuf[curbuf_read] = '\0';
        exit_read_function = TRUE;
      } else {
        /** Any other character entered at the
keyboard ... **/
        if (message == FALSE) {
          bufp += DISPLAY(bufp, MESSAGE_COL,
                          MESSAGE_ROW, INVALID_MSG);
          bufp += GOTOXY(bufp, col + size, row);
          PAINTSCR(screen_buf);
          bufp = screen_buf;
          screen_bufindex = 0;
          message = TRUE;
        }
        curbuf_read++;
      }
    } /** End of the WHILE loop **/
    if (OVERFLOW == TRUE && exit_read_function ==
FALSE) {
      /*
       * if number of characters are exceeding the
field
       * limit beep and warning message is
necessary
       */
      if (message == FALSE) {
        bufp += DISPLAY(bufp, MESSAGE_COL,
                        MESSAGE_ROW,
EXC_FLD_LIM_MSG);
        PAINTSCR(screen_buf);
        bufp = screen_buf;
        screen_bufindex = 0;
        message = TRUE;
      }
      *data = atoi(temp);
      return_status = curbuf[curbuf_read];
      curbuf[curbuf_read] = '\0';
      curbuf_read = 0;
      OVERFLOW = FALSE;
    } else {
      screen_bufindex = bufp - screen_buf;
      if ((curbuf_read == read_count) ||
(curbuf_read == 0)
          || (screen_bufindex > SCRBUF_LEN -
CURBUFLEN))
        {
          PAINTSCRLEN(screen_buf, screen_bufindex);
          screen_bufindex = 0;
          bufp = screen_buf;
        }
    }
  }
  /* ensuring unnecessary warning messages are
removed */
  if (message == TRUE) {
    bufp += DISPLAY(bufp, MESSAGE_COL,
                    MESSAGE_ROW, ERASE_MSG);
    message = FALSE;
    PAINTSCR(screen_buf);
    bufp = screen_buf;
    screen_bufindex = 0;
  }
  return (return_status);
}
int
read_string(col, row, size, flags, data)
  int col, row, size, flags;
char *data;
{
  int exit_read_function = FALSE,
previous_data_exists = FALSE,
    data_full = FALSE;
  int return_status = TAB, bytes_read = 0, i = 0, j
= 0,
    size_tot = 0;
  char *bufp, temp[80];
  char erase_field[20];
  strncpy(temp, "\0", 1);
  curbuf_consumed = 0;
  bufp = screen_buf + screen_bufindex;
  /* Position cursor at start of field */
  if (curbuf_read == read_count || curbuf_read ==
0) {
    screen_buf[0] = '\0';
    bufp += GOTOXY(bufp, col, row);/* Goto input
area */
    PAINTSCRLEN(screen_buf, bufp - screen_buf);
    bufp = screen_buf;
  }
  if ((*(char *) data) != '\0')
    previous_data_exists = TRUE;
  while (exit_read_function == FALSE) {
    /*
     * Below we read from standard input into the
array curbuf.
     * curbuf_read is the pointer to the array
curbuf indicating
     * the position upto which the curbuf has been
parsed.
     * curbuf_consumed is the number of elements in
the buffer
     * temp that holds the array that is to be
displayed.
     * Elements of curbuf_consumed is selectively
copied from
     * curbuf Note:read_count is the total number
of characters
     * in the buffer curbuf. curbuf_read is always
less than or
     * equal to read_count.
     */
    if (curbuf_read == read_count) {
      curbuf_read = 0;
      read_count = read(tty_in, curbuf, size -
size_tot);
      if (read_count == 0)
        syserr("TTY lost connection");
    }
    if (message == TRUE) {
      bufp += DISPLAY(bufp, MESSAGE_COL,
                      MESSAGE_ROW, ERASE_MSG);
      message = FALSE;
    }
    if (previous_data_exists == TRUE) {
      if (curbuf[curbuf_read] == DELETE) {
        previous_data_exists = FALSE;
        strncpy(erase_field, blanks, size);
        erase_field[size] = '\0';
        bufp += DISPLAY(bufp, col, row,
erase_field);
        bufp += GOTOXY(bufp, col, row);
      } else {
        if (curbuf[curbuf_read] < ' ' ||
curbuf[curbuf_read] >
            '~') {
          exit_read_function = TRUE;
          previous_data_exists = FALSE;
          return_status = curbuf[curbuf_read];
          curbuf[curbuf_read] = '\0';
        } else {
          previous_data_exists = FALSE;
          strncpy(erase_field, blanks, size);
          erase_field[size] = '\0';
          bufp += DISPLAY(bufp, col, row,
erase_field);
          bufp += GOTOXY(bufp, col, row);
        }
```

```
      }
    }
    while ((curbuf_read < read_count) &&
(exit_read_function ==
                                        FALSE)) {
      if (curbuf[curbuf_read] >= ' ' &&
curbuf[curbuf_read] <=
          '~') { /** if between ASCII space (040)
through ~ (0176) **/
        for (; curbuf[curbuf_read] >= ' '
               && curbuf[curbuf_read] <= '~';
curbuf_read++) {
          /*
           * ensuring the curbuf_consumed is
           * not more than field size
           */
          if (curbuf_consumed < size) {
            temp[curbuf_consumed] =
              curbuf[curbuf_read];
            curbuf_consumed++;
          }
          /* else overflow condition */
          else
            OVERFLOW = TRUE;
          curbuf[curbuf_read] = '\0';/* erasing
characters
                                      * already read
from the
                                      * buffer */
        }
        temp[curbuf_consumed] = '\0';/* terminate
temp
                                        string */
        bufp += DISPLAY(bufp, col, row, temp);
        return_status = curbuf[curbuf_read];
      } else if (curbuf[curbuf_read] == TAB
                 || curbuf[curbuf_read] == LF
                 || curbuf[curbuf_read] == BACKTAB
                 || curbuf[curbuf_read] == SUBMIT) {
        if (curbuf_consumed > 0) {
          if (message == TRUE) {
            bufp += DISPLAY(bufp,
                            MESSAGE_COL,
                            MESSAGE_ROW,
                            ERASE_MSG);
            message = FALSE;
          }
          temp[curbuf_consumed] = '\0';
          strcpy(data, temp);
          exit_read_function = TRUE;
          return_status = curbuf[curbuf_read];
          curbuf[curbuf_read] = '\0';
          curbuf_read++;
          curbuf_consumed = 0;
        } else {
          if (message == TRUE) {
            bufp += DISPLAY(bufp,
                            MESSAGE_COL,
                            MESSAGE_ROW,
                            ERASE_MSG);
            message = FALSE;
          }
          temp[curbuf_consumed] = '\0';
          strcpy(data, temp);
          exit_read_function = TRUE;
          return_status = curbuf[curbuf_read];
          curbuf[curbuf_read] = '\0';
          curbuf_read++;
        }
      } else if (curbuf[curbuf_read] == DELETE) {
        for (curbuf_read = curbuf_read;
             curbuf[curbuf_read] ==
               DELETE
             ; curbuf_read++) {
          curbuf[curbuf_read] = '\0';
          temp[curbuf_consumed - 1] = '\0';
          if (curbuf_consumed != 0)
            curbuf_consumed--;
        }
        if (curbuf_consumed >= 0) {
          bufp += BLANK_UNDERLINE(bufp, col,
                                  row, " ");
          bufp += DISPLAY(bufp, col, row, temp);
          PAINTSCR(screen_buf);
          bufp = screen_buf;
          screen_bufindex = 0;
        } else {
          if (message == FALSE) {
            bufp += DISPLAY(bufp,
                            MESSAGE_COL,
                            MESSAGE_ROW,
                            EXC_FLD_LIM_MSG);
            bufp += BEEP(bufp);
            PAINTSCR(screen_buf);
            bufp = screen_buf;
            screen_bufindex = 0;
            message = TRUE;
          }
          curbuf[curbuf_read] = '\0';
          curbuf_read = 0;
        }
      } else if (curbuf[curbuf_read] == QUIT) {
        temp[0] = '\0';
        return_status = QUIT;
        curbuf[curbuf_read] = '\0';
        exit_read_function = TRUE;
      } else {/** Any other character entered at
the keyboard ... **/
        if (message == FALSE) {
          bufp += DISPLAY(bufp, MESSAGE_COL,
                          MESSAGE_ROW, INVALID_MSG);
          bufp += GOTOXY(bufp, col, row);
          message = TRUE;
        }
        curbuf_read++;
      }
    } /** End of the WHILE loop **/
    if (OVERFLOW == TRUE && exit_read_function ==
FALSE)
        /** If read enough to fill the size already
**/
      {
        if (message == FALSE) {
          bufp += DISPLAY(bufp, MESSAGE_COL,
                          MESSAGE_ROW,
EXC_FLD_LIM_MSG);
          PAINTSCR(screen_buf);
          bufp = screen_buf;
          screen_bufindex = 0;
          message = TRUE;
        }
        OVERFLOW = FALSE;
        temp[curbuf_consumed] = '\0';
        strcpy(data, temp);
        curbuf_consumed--;
        return_status = curbuf[curbuf_read];
      } else {
        screen_bufindex = bufp - screen_buf;
        if ((curbuf_read == read_count) ||
(curbuf_read == 0)
            || (screen_bufindex > SCRBUF_LEN -
CURBUFLEN)) {
          PAINTSCRLEN(screen_buf, screen_bufindex);
          screen_bufindex = 0;
          bufp = screen_buf;
        }
      }
  }
  if (message == TRUE) {
    bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW,
                    ERASE_MSG);
    message = FALSE;
    PAINTSCR(screen_buf);
    screen_bufindex = 0;
  }
  return (return_status);
}
void display_newo();
void display_paym();
void display_ords();
```

```
void display_del();
void display_stock();
void (*p_print_function[]) () = {
  &display_newo,
  &display_paym,
  &display_ords,
  &display_del,
  &display_stock
};
display_output(int txn_type)
{
  char c;
  (p_print_function[txn_type]) ();
/*  read(tty_in, &c, 1); */
}
void
display_newo()
{
  struct no_itm_struct *ol_ptr, *ool;
  char *bufp;
  int i, r;
  double total = 0.0;
  bufp = output_screen;
  if (oNO->status == '\0') {
    PAINTSCR(EXECUTION_STATUS_MSG);
    return;
  } else {
    bufp += SWITCH_TO_NORMAL(bufp);
    bufp += DISPLAY(bufp, 61, 2, oNO->o_entry_d);
    bufp += DISPLAY(bufp, 25, 3, oNO->c_last);
    bufp += DISPLAY(bufp, 52, 3, oNO->c_credit);
    bufp += DISPLAY_FLOAT(bufp, 5, 64, 3, oNO-
>c_discount);
    bufp += DISPLAY_INT(bufp, 8, 15, 4, oNO->o_id);
    bufp += DISPLAY_INT(bufp, 2, 42, 4, oNO-
>o_ol_cnt);
    bufp += DISPLAY_FLOAT(bufp, 5, 59, 4, oNO-
>w_tax);
    bufp += DISPLAY_FLOAT(bufp, 5, 74, 4, oNO-
>d_tax);
    ol_ptr = iNO->o_ol;
    ool = oNO->o_ol;
    for (i = 0, r = FIRST_OL_ROW; i < iNO-
>o_ol_cnt;
         r++, i++, ol_ptr++, ool++) {
      bufp += DISPLAY(bufp, 19, r, ool->i_name);
      bufp += DISPLAY_INT(bufp, 3, 51, r, ool-
>s_quantity);
      bufp += DISPLAY(bufp, 58, r, ool->brand);
      bufp += DISPLAY_MONEY(bufp, 6, 62, r, ool-
>i_price);
      bufp += DISPLAY_MONEY(bufp, 7, 71, r, ool-
>ol_amount);
    }
    bufp += DISPLAY_MONEY(bufp, 8, 70, 22, oNO-
>total);
    bufp += DISPLAY(bufp, 19, 22, oNO->status);
    bufp += DISPLAY(bufp, 23, 75, "SGI");
    *bufp++ = '\0';
    PAINTSCRLEN(output_screen, bufp -
output_screen);
  }
#ifdef DEBUG
  Clog("DBG: Screen output chars = %d\n", (bufp -&

output_screen[0]));
#endif
}
void
display_paym()
{
  char *bufp, temp[51], tempbuf2[201];
  char *make_phone(char *), *make_zip(char *);
  FILE *fp;

  bufp = output_screen;
  bufp += SWITCH_TO_NORMAL(bufp);/* jr */
  bufp += DISPLAY(bufp, 7, 2, oPT->h_date);
  bufp += DISPLAY(bufp, 1, 5, oPT->w_street_1);
  bufp += DISPLAY(bufp, 1, 6, oPT->w_street_2);
  bufp += DISPLAY(bufp, 1, 7, oPT->w_city);
  bufp += DISPLAY(bufp, 22, 7, oPT->w_state);
  bufp += DISPLAY(bufp, 25, 7, make_zip(oPT-
>w_zip));
  bufp += DISPLAY(bufp, 42, 5, oPT->d_street_1);
  bufp += DISPLAY(bufp, 42, 6, oPT->d_street_2);
  bufp += DISPLAY(bufp, 42, 7, oPT->d_city);
  bufp += DISPLAY(bufp, 63, 7, oPT->d_state);
  bufp += DISPLAY(bufp, 66, 7, make_zip(oPT-
>d_zip));
  bufp += DISPLAY_INT(bufp, 4, 11, 9, oPT->c_id);
  bufp += DISPLAY(bufp, 29, 10, oPT->c_last);
  bufp += DISPLAY(bufp, 9, 10, oPT->c_first);
  bufp += DISPLAY(bufp, 26, 10, oPT->c_middle);
  bufp += DISPLAY(bufp, 9, 11, oPT->c_street_1);
  bufp += DISPLAY(bufp, 9, 12, oPT->c_street_2);
  bufp += DISPLAY(bufp, 9, 13, oPT->c_city);
  bufp += DISPLAY(bufp, 30, 13, oPT->c_state);
  bufp += DISPLAY(bufp, 33, 13, make_zip(oPT-
>c_zip));
  bufp += DISPLAY(bufp, 58, 10, oPT->c_since);
  bufp += DISPLAY(bufp, 58, 11, oPT->c_credit);
  bufp += DISPLAY_FLOAT(bufp, 5, 58, 12, oPT-
>c_discount);
  bufp += DISPLAY(bufp, 58, 13, make_phone(oPT-
>c_phone));
  bufp += DISPLAY_MONEY(bufp, 14, 55, 15, oPT-
>c_balance);
  bufp += DISPLAY_MONEY(bufp, 13, 17, 16, oPT-
>c_credit_lim);

#ifdef DEBUG
  fp = fopen("/usr/tmp/MYLOG" ,  "a");
  fprintf(fp, "DBG: Screen output chars = %d\n",
(bufp - &output_screen[0]));
  fprintf(fp, "DBG: c_data output = %s\n", &(oPT-
>c_data[0]));
  fclose(fp);
#endif

  if (oPT->c_data[0] != ' ') {
    bufp += DISPLAY50(bufp, 12, 18, oPT->c_data);
    bufp += DISPLAY50(bufp, 12, 19, oPT->c_data +
50 );
    bufp += DISPLAY50(bufp, 12, 20, oPT->c_data +
100);
    bufp += DISPLAY50(bufp, 12, 21, oPT->c_data +
150);
  }
  if (!oPT->h_date)
    bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW
- 2,
                    BAD_INPUTS);
  bufp += DISPLAY(bufp, 23, 75, "SGI");
  *bufp++ = '\0';
  PAINTSCRLEN(output_screen, bufp - output_screen);
}
void
display_ords()
{
  struct ord_itm_struct *sol;
  char *bufp;
  int i = 0, r = 8;
  bufp = output_screen;
  bufp += SWITCH_TO_NORMAL(bufp);
  bufp += DISPLAY_INT(bufp, 4, 11, 3, oOS->c_id);
  bufp += DISPLAY(bufp, 44, 3, oOS->c_last);
  bufp += DISPLAY(bufp, 24, 3, oOS->c_first);
  bufp += DISPLAY(bufp, 41, 3, oOS->c_middle);
  bufp += DISPLAY_MONEY(bufp, 9, 15, 4, oOS-
>c_balance);
  bufp += DISPLAY_INT(bufp, 8, 15, 6, oOS->o_id);
  bufp += DISPLAY(bufp, 38, 6, oOS->o_entry_d);
  bufp += DISPLAY_INT(bufp, 2, 76, 6, oOS-
>o_carrier_id);
  for (i = 0; i < oOS->ol_cnt; i++) {
    sol = &oOS->s_ol[i];
    if (sol->ol_supply_w_id > 0) {
      bufp += DISPLAY_INT(bufp, 4, 3, r, sol->
                          ol_supply_w_id);
```

```
      bufp += DISPLAY_INT(bufp, 6, 14, r, sol-
>ol_i_id);
      bufp += DISPLAY_INT(bufp, 2, 25, r, sol->
                          ol_quantity);
      bufp += DISPLAY_MONEY(bufp, 8, 32, r, sol->
                            ol_amount);
      bufp += DISPLAY(bufp, 47, r, sol-
>ol_delivery_d);
      r++;
    }
  }
  if (!oOS->ol_cnt)
    bufp += DISPLAY(bufp, MESSAGE_COL, MESSAGE_ROW
                    - 2, BAD_INPUTS);
  bufp += DISPLAY(bufp, 23, 75, "SGI");
  *bufp++ = '\0';
  PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
  Clog("DBG: Screen output chars = %d\n", (bufp -&

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

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

&output_screen[0]));
#endif
}
char *
make_phone(char *data)
{
  static char tempphone[20];
  strncpy(tempphone, data, 6);
  tempphone[6] = '-';
  strncpy(&tempphone[7], &data[6], 3);
  tempphone[10] = '-';
  strncpy(&tempphone[11], &data[9], 3);
  tempphone[14] = '-';
  strncpy(&tempphone[15], &data[12], 4);
  tempphone[19] = '\0';
  return tempphone;
}
char *
make_zip(char *data)
{
  static char temp[10];
  strncpy(temp, data, 5);
  temp[5] = '-';
  strncpy(&temp[6], &data[5], 4);
  temp[10] = '\0';
  return temp;
}
```

```
============================
screen.h
============================

/**
 ** screen.h: All vars for screen to/from tux
 **/
#include <sys/termio.h>
#include <sybfront.h>
#include <sybdb.h>
#include "monitor.h"

int w_id;
int d_id;

extern int tty_in;
extern int tty_out;

char *tuxibuf;
char *tuxobuf;

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

#define MAX_FORMS 5
#define MESSAGE_ROW 24
#define MESSAGE_COL 1
#define RTE_SYNCH_CHARACTER '\1'
#define SCRBUF_LEN 1536
#define FIRST_OL_ROW 7
#define CLRSCN(buf) sprintf(buf,"\033[0m\033[2J")
#define DISPLAY_INT(buf,wid,x,y,ip)
sprintf(buf,"\033[%d\;%dH%*.1d", y, x,wid,ip)
#define DISPLAY_MONEY(buf,wid,x,y,fp)
sprintf(buf,"\033[%d\;%dH$%#*.2f",y,x,wid,fp)
#define DISPLAY_FLOAT(buf,wid,x,y,fp)
sprintf(buf,"\033[%d\;%dH%#*.2f", y, x,wid,fp)
#define DISPLAY(buf,x,y,txt)
sprintf(buf,"\033[%d\;%dH%s", y, x,txt)
#define DISPLAY50(buf,x,y,txt)
sprintf(buf,"\033[%d\;%dH%50.50s", y, x,txt)
#define PAINTSCR(buf)
write(tty_out,buf,strlen(buf))
#define PAINTSCRLEN(buf, len)
write(tty_out,buf,len)
#define SWITCH_TO_NORMAL(buf) sprintf(buf,"\033[m")
#define SWITCH_TO_UNDERL(buf)
sprintf(buf,"\033[4m")
#define GOTOXY(buf,x,y) sprintf(buf,"\033[%d\;%dH",
y, x)
#define BEEP(buf) sprintf(buf,"\007")
#define BLANK_UNDERLINE(buf,x,y,txt)
sprintf(buf,"\033[4m;\033[%d;\%dH%s",y,x,txt);
#define CLRSCN_STR "\033[0m\033[2J"
#define DISPLAY_STR(x,y,txt)
'\033[/**/y;/**/xH/**/txt'

/** Possible status values returned by read
functions **/
#define CANCELLED 3
#define PREVIOUS_FIELD 4
/**
 ** Possible key strokes read in by the read
functions.
 ** Some are also returned as status from the read
functions.
 **/
#define BACKTAB 2               /** CTRL B **/
```

```
/** was **/
/**   #define DELETE 8  **/
/** is **/
#define DELETE 127
#define ESCAPE 27
#define LF 10
#define QUIT 3                    /** CNTRL-C Key
**/
#define SPACE 32
#define SUBMIT 13                 /** CR Submit **/
#define TAB 9
#define UNDERLINE 95

#define LEAVE_SCREEN_MIN 300      /** Minimum # of
characters to leave screen **/
#define LEAVE_SCREEN_TIMEOUT 2    /** Minimum time
to leave screen, 10=1sec **/

static int curbuf_consumed = 0;
static int curbuf_read = 0;
static int read_count = 0;
#define CURBUFLEN 300
static char curbuf[CURBUFLEN];
static BOOLEAN OVERFLOW = FALSE;

static BOOLEAN message;
BOOLEAN payment_input = FALSE;

static struct termio tbufsave;

extern void syserr();
void Init_Screen();
void display_screen_array(int);
void Send_Menu();
int Get_Menu_Input();

typedef struct {
int y;              /** Y-axis scren pos. **/
int x;              /** X-axis scren pos. **/
int len;            /** Datafield size in Bytes **/
int flags;          /** Indicates mandatory data
fields: 1 => required **/
int *dptr;          /** Data pointer **/
int (*fptr) ();     /** Pointer to func. to read
data of type *dptr **/
} io_elem;

int int_h_amount;

/* All the possible messages to print out */
const static char MANDATORY_MSG[] =
"\033[24;1H\033[mMandatory data field! Please enter
data.\033[K\033[4m\1";
const static char INVALID_MSG[] =
"\007\033[24;1HAn invalid character was entered.
Please enter again.\033[K\033[4m\1";
const static char ERASE_MSG[] =
"\033[24\;1H\033[K\033[4m";
const static char MIN1DIGIT_MSG[] =
"\033[24;1H\033[0mYou must enter atleast 1 digit.
Please reenter.\033[4m\1";
const static char BAD_INPUTS[] = " #### Bad input
data was entered -- Select again ####
\1\033[2;30H";
const static char INCOMPLINE_MSG[] =
"\033[24;1H\033[0mOrder line is incomplete. Please
complete the whole line.\033[4m\1";
const static char ID_OR_LAST_MSG[] =
"\033[24;1H\033[0mYou must enter either the Last
Name or the Customer Number.\033[4m\1";
const static char EXC_MAX_LFT_DEC_DGT_MSG[]
="\033[24;1H\033[0mMaximum digits left of decimal
point already entered. '.' expected\033[4m\1";
const static char EXC_FLD_LIM_MSG[] =
"\007\033[24;1H\033[0mMaximum digits already
entered. Tab or <CR> expected\033[4m\1";/* jr */
const static char EXECUTION_STATUS_MSG[] =
"\033[0m\033[22\;18HItem number is not valid. Tx
rolled back.";
const static char DELIVERY_QUEUED_MSG[] =
"\033[0m\033[6\;19HDelivery has been queued";

int read_integer(int, int, int, int, int *);
int read_money(int, int, int, int, float *);
int read_string(int, int, int, int, char *);

char menu_buf[] = "\033[0m\033[23\;1HNew-Order(1)
Payment(2) Order-Status(3) Delivery(4) Stock-
Level(5) Exit(9)\1";
int menu_buflen = sizeof (menu_buf);

io_elem neworder_inputs[] = {
/* y x len flags ptr to data ptr to read function
*/
/* - - --- ----- ----------- ------------------- */
2, 29, 2, 0, 0, &read_integer,
3, 12, 4, 0, 0, &read_integer,
7, 3, 4, 0, 0, &read_integer,
7, 10, 6, 0, 0, &read_integer,
7, 45, 2, 0, 0, &read_integer,
8, 3, 4, 0, 0, &read_integer,
8, 10, 6, 0, 0, &read_integer,
8, 45, 2, 0, 0, &read_integer,
9, 3, 4, 0, 0, &read_integer,
9, 10, 6, 0, 0, &read_integer,
9, 45, 2, 0, 0, &read_integer,
10, 3, 4, 0, 0, &read_integer,
10, 10, 6, 0, 0, &read_integer,
10, 45, 2, 0, 0, &read_integer,
11, 3, 4, 0, 0, &read_integer,
11, 10, 6, 0, 0, &read_integer,
11, 45, 2, 0, 0, &read_integer,
12, 3, 4, 0, 0, &read_integer,
12, 10, 6, 0, 0, &read_integer,
12, 45, 2, 0, 0, &read_integer,
13, 3, 4, 0, 0, &read_integer,
13, 10, 6, 0, 0, &read_integer,
13, 45, 2, 0, 0, &read_integer,
14, 3, 4, 0, 0, &read_integer,
14, 10, 6, 0, 0, &read_integer,
14, 45, 2, 0, 0, &read_integer,
15, 3, 4, 0, 0, &read_integer,
15, 10, 6, 0, 0, &read_integer,
15, 45, 2, 0, 0, &read_integer,
16, 3, 4, 0, 0, &read_integer,
16, 10, 6, 0, 0, &read_integer,
16, 45, 2, 0, 0, &read_integer,
17, 3, 4, 0, 0, &read_integer,
17, 10, 6, 0, 0, &read_integer,
17, 45, 2, 0, 0, &read_integer,
18, 3, 4, 0, 0, &read_integer,
18, 10, 6, 0, 0, &read_integer,
18, 45, 2, 0, 0, &read_integer,
19, 3, 4, 0, 0, &read_integer,
19, 10, 6, 0, 0, &read_integer,
19, 45, 2, 0, 0, &read_integer,
20, 3, 4, 0, 0, &read_integer,
20, 10, 6, 0, 0, &read_integer,
20, 45, 2, 0, 0, &read_integer,
21, 3, 4, 0, 0, &read_integer,
21, 10, 6, 0, 0, &read_integer,
21, 45, 2, 0, 0, &read_integer,
999
};
io_elem payment_inputs[] = {
/* y x len flags ptr to data ptr to read function
*/
/* - - --- ----- ----------- ------------------- */
4, 52, 2, 0, 0, &read_integer,
9, 11, 4, 0, 0, &read_integer,
9, 33, 4, 0, 0, &read_integer,
9, 54, 2, 0, 0, &read_integer,
10, 29, 16, 0, 0, &read_string,
15, 24, 8, 0, 0, &read_integer,
999
};
io_elem ordstat_inputs[] = {
/* y x len flags ptr to data ptr to read function
*/
```

```
/* - - --- ----- ----------- ------------------- */
2, 29, 2, 0, 0, &read_integer,
3, 11, 4, 0, 0, &read_integer,
3, 44, 16, 0, 0, &read_string,
999
};
io_elem delivery_inputs[] = {
/* y x len flags ptr to data ptr to read function
*/
/* - - --- ----- ----------- ------------------- */
4, 17, 2, 0, 0, &read_integer,
999
};
io_elem stocklev_inputs[] = {
/* y x len flags ptr to data ptr to read function
*/
/* - - --- ----- ----------- ------------------- */
4, 24, 2, 0, 0, &read_integer,
999
};
io_elem wd_inputs[] = {
/* y x len flags ptr to data ptr to read function
*/
/* - - --- ----- ----------- ------------------- */
2, 16, 4, 0, 0, &read_integer,
2, 43, 4, 0, 0, &read_integer,
999
};
typedef struct {
int x;
int y;
char *text;
} text_elem;

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

struct form_info {
const text_elem *tp;
char *blank_form;
int blank_formlen;
io_elem *input_elems;
int num_input_elems;
};

char output_screen[SCRBUF_LEN];

struct form_info Forms[MAX_FORMS] = {
{NO_text_elem, 0, 0, neworder_inputs, 0},
{PT_text_elem, 0, 0, payment_inputs, 0},
{OS_text_elem, 0, 0, ordstat_inputs, 0},
{DY_text_elem, 0, 0, delivery_inputs, 0},
{SL_text_elem, 0, 0, stocklev_inputs, 0}/*,
{WD_text_elem, 0, 0, wd_inputs, 0} */
};


#ifdef Multiple_blank_form
const char WD_blank_form[SCRBUF_LEN] =
CLRSCN_STR/**/DISPLAY_STR(2,1,'Warehouse:')/**/DISP
LAY_STR(2,26,'District:')";
#endif


============================
shared_rpc_var.c
============================

/** Mandatory SYBASE includes **/
#include <sybfront.h>
#include <sybdb.h>

#define XCTION_COUNT   7

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

typedef struct
{
       int     total, deadlock, overlimit;
       double  sumResp, sumRespSq;
} STATS;

XCTION xction[XCTION_COUNT+1] =
```

```
{
  {"new_order"},
  {"payment_byid"},
  {"payment_byname"},
  {"order_status_byid"},
  {"order_status_byname"},
  {"delivery_qu"},
  {"stock_level"},
  {"NULL"}
};

char            * server       = NULL;
char            * database     = "tpcc";
char            * user         = "sa";
char            * password     = "";
char            * trans        = NULL;
int             deliveryRatio  = 4;
int             nclients       = 36;
int             nlocals        = -1;
int             duration       = 600;
int             nengines       = 0;
int             rampup          = 0;
int             runtime         = 0;
int             nwares         = 0;
int             deadlock       = 0;
int             prev_xact_type = -9999;
int             packet         = 4096;
int             clientid       = 0;
int             lines_per_call = 15;
int             rollback_pct   = 1;
int             key_percent    = 0;
int             queue          = 1234;
int             xact_type;

/*
** Database and Txn. related vars.
*/
/*
** Structure for each line of an order
*/

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

char b_g[2];
DBFLT8 total_amount;
DBTINYINT commit_flag;
int bad_items;
RETCODE code;
DBSMALLINT w_id;
DBTINYINT d_id;
ORDER_LINE ol[15];

/*
** Variables for the customer table
*/
DBINT c_id;
DBTINYINT c_d_id;
DBSMALLINT c_w_id;
char c_first[17];
char c_middle[3];
char c_last[17];
char c_street_1[21];
char c_street_2[21];
char c_city[21];
char c_state[3];
char c_zip[10];
char c_phone[17];
char c_since[31];
char c_credit[3];
DBFLT8 c_credit_lim;
DBREAL c_discount;
DBFLT8 c_balance;
char c_data[201];
/*
** Variables for warehouse
*/
char w_name[11];
char w_street_1[21];
char w_street_2[21];
char w_city[21];
char w_state[3];
char w_zip[10];
DBREAL w_tax;
/*
** Variables for district
*/
DBTINYINT d_id;
DBSMALLINT d_w_id;
char d_name[11];
char d_street_1[21];
char d_street_2[21];
char d_city[21];
char d_state[3];
char d_zip[10];
DBREAL d_tax;
/*
** Variables for item table
*/
int i_id;
DBFLT8 i_price;
char i_name[25];
/*
** Variables for the stock table
*/
DBSMALLINT s_quantity;
DBSMALLINT threshold;
DBINT low_count;
char s_dist[25];
/*
** Variables for order table
*/
int o_id;
DBTINYINT o_d_id;
DBSMALLINT o_w_id;
DBSMALLINT o_c_id;
char o_entry_d[31];
DBSMALLINT o_carrier_id;
DBSMALLINT o_ol_cnt, o_ol_now, o_ol_done;
DBTINYINT o_all_local;
/*
** Variables for order_line
*/
int ol_o_id;
DBTINYINT ol_d_id;
DBSMALLINT ol_w_id;
DBSMALLINT ol_number;
DBINT ol_i_id;
DBSMALLINT ol_supply_w_id;
char ol_delivery_d[31];
DBSMALLINT ol_quantity;
DBFLT8 ol_amount;
/*
** Variables for new_order tble
*/
int no_o_id;
DBTINYINT no_d_id;
DBSMALLINT no_w_id;
/*
** Variables for history table
*/
DBFLT8 h_amount;
char h_date[20];

============================
shared_tux_var.c
============================

/** Standard includes **/
#include <stdio.h>

/** Mandatory TUXEDO includes **/
```

```
#include "atmi.h"
#include "tuxstructs.h"

/* For NewOrder Txn. */
struct newo_inf *NewO;
TPSVCINFO *NewOrqst;

/* For Payment Txn. */
struct pay_inf *PayM;
TPSVCINFO *PayMrqst;

/* For OrderStat Txn. */
struct ord_inf *OrdS;
TPSVCINFO *OrdSrqst;

/* For STOCK: */
struct stock_inf *StkL;
TPSVCINFO *StkL_rqst;

/* For Delivery Txn. */
struct req_struct *DelY;
char outbuf[1024];          /* Buffer for results
file */
int tx_count = 0;           /* Transaction counter
*/
FILE *DelYfp;               /* FP to write out
DeliveryLog */
TPSVCINFO *DelYrqst;


============================
stkl_srv.c
============================

/**
 ** stkl_srv.c : TUXEDO server for SYBASE
stocklevel Tx.
 **
 **/
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <signal.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

/* SYBASE includes */
#include <sybfront.h>
#include <sybdb.h>
#include "tpcc.h"

/* TUXEDO includes */
#include "tuxstructs.h"

DBPROCESS *dbproc;
LOGINREC *login;

/**
 ** Setup for communication with DB.
 **/
int
stkl_init()
{
  /* Install the error and message handler */
  dberrhandle(err_handler);
  dbmsghandle(msg_handler);

  /* Initialize global variable for error handling
*/
  deadlock = 0;
  xact_type = XACT_STOCK;
  login = dblogin();
  DBSETLUSER(login, USER);
  DBSETLPACKET(login, 4096);
  DBSETLCHARSET(login, getenv("CHARSET"));

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

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

  return(0);
}

/**
 ** Parses the input 'stock_inf' struct from
tuxedo.
 ** Sets up the shared_var for rpc with DB.
 ** Returns shared_var results from rpc into
 **   the struct sent back to Tuxedo.
 **/
stkl(tuxin)
TPSVCINFO *tuxin;
{
  StkL = (struct stock_inf *)(tuxin->data);
  w_id = StkL->w_id;
  d_id = StkL->d_id;
  threshold = StkL->threshold;

  /** Query the DB **/
  stock_level_rpc();

  /** Now get the rpcvar info into the tuxstruct
**/
  StkL->low_stock = low_count;

  /** Finally, send it all back **/
  tpreturn(TPSUCCESS, 0, tuxin->data, sizeof(struct
stock_inf), 0);
}

/* Tuxedo Prologue */
int
tpsvrinit(argc, argv)
char **argv;
{
  return(stkl_init());
}

/** Tuxedo Epilogue */
void
tpsvrdone()
{
  dbexit();
}

/** Service advertized by Tuxedo **/
STOCK(tuxin)
TPSVCINFO *tuxin;
{
  stkl(tuxin);
}

============================
tpcc.h
============================

#define SERVER "ROCKY"
#define DATABASE "tpcc"
#define USER "sa"
#define MAX_ERROR       1
#define MAXDIST        10
#define MaxTries       5
#define smaller(x,y)   (x<y ? x : y)
#define XCTION_COUNT   7
```

```
#define XACT_NEWO 0
#define XACT_PAYM_ID 1
#define XACT_PAYM_NAME 2
#define XACT_ORDS_ID 3
#define XACT_ORDS_NAME 4
#define XACT_DEL 5
#define XACT_STOCK 6
#define XACT_BKEND 7

extern char    * server;
extern char    * database;
extern char    * user;
extern char    * password;
extern char    * trans;
extern LOGINREC       * login;
extern DBPROCESS * dbproc;
extern int     nclients, nlocals, duration,
nengines, nwares, key_percent;
extern int     deadlock, xact_type, prev_xact_type;
extern int     packet, rollback_pct, delta;
extern int     clientid, deliveryRatio, queue;

#define IDoDelivery(x) (x %  deliveryRatio ==
deliveryRatio-1)

double  drand();
int     err_handler();
int     msg_handler();
void    init_time();
long    gettime();
void    disaster(char * msg);

/**
 **  Functions are defined in the SYBASE rpc.c file
 **/

void    new_order_rpc();
void    payment_byid_rpc();
void    payment_byname_rpc();
void    order_status_byid_rpc();
void    order_status_byname_rpc();
void    stock_level_rpc();
void    delivery_qu_add();
void    delivery_rpc();

void    display_xction();
void    sleep_before_retry ();

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

typedef struct
{
       int     total, deadlock, overlimit;
       double  sumResp, sumRespSq;
} STATS;

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

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

extern XCTION xction[XCTION_COUNT+1];
extern STATS  stats[XCTION_COUNT];

extern char           b_g[2];
extern DBFLT8         total_amount;
extern DBTINYINT        commit_flag;
extern int            bad_items;
extern RETCODE        code;

extern DBSMALLINT     w_id;
extern DBTINYINT      d_id;
extern ORDER_LINE     ol[15];

/*
** Variables for the customer table
*/

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

/*
** Variables for warehouse
*/

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

/*
** Variables for district
*/

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

/*
** Variables for item table
*/

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

/*
** Variables for the stock table
*/

extern DBSMALLINT     s_quantity;
extern DBSMALLINT     threshold;
extern DBINT          low_count;
extern char           s_dist[25];

/*
** Variables for order table
*/

extern int            o_id;
extern DBTINYINT      o_d_id;
extern DBSMALLINT     o_w_id;
```

```
extern DBSMALLINT      o_c_id;
extern char            o_entry_d[31];
extern DBSMALLINT      o_carrier_id;
extern DBSMALLINT      o_ol_cnt, o_ol_now,
o_ol_done;
extern DBTINYINT       o_all_local;

/*
** Variables for order_line
*/

extern int             ol_o_id;
extern DBTINYINT       ol_d_id;
extern DBSMALLINT      ol_w_id;
extern DBTINYINT       ol_number;
extern DBINT           ol_i_id;
extern DBSMALLINT      ol_supply_w_id;
extern char            ol_delivery_d[31];
extern DBSMALLINT      ol_quantity;
extern DBFLT8          ol_amount;

/*
** Variables for new_order tble
*/

extern int             no_o_id;
extern DBTINYINT       no_d_id;
extern DBSMALLINT      no_w_id;

/*
** Variables for history table
*/

extern DBFLT8          h_amount;
extern char            h_date[20];

============================
tuxstructs.h
============================

/**
 ** Tuxedo Tx. Buffer Defns. Used by the TM servers
to receive data
 ** from the TM client side programs. These
variables are then read
 ** out into the shared variables defined in tpcc.h
for use by the
 ** Sybase rpc.c routines.
 **
 **/
/** Mandatory TUXEDO includes **/
#include "atmi.h"
#include "userlog.h"

/* For NEWO: */
struct items_inf {
  int ol_i_id;
  int ol_supply_w_id;
  int ol_quantity;
  char i_name[25];
  int s_quantity;
  char brand[2];
  double i_price;
  double ol_amount;
};
struct newo_inf {
  int w_id;
  int d_id;
  int c_id;
  int o_ol_cnt;
  int o_all_local;
  struct items_inf n_items[15];

  char o_entry_d[31];
  char c_last[17];
  char c_credit[3];
  double c_discount;
  double w_tax;
  double d_tax;
  int o_id;
  char status[25];
  double total;
};

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

  char h_date[20];
  char w_street_1[21];
  char w_street_2[21];
  char w_city[21];
  char w_state[3];
  char w_zip[11];
  char d_street_1[21];
  char d_street_2[21];
  char d_city[21];
  char d_state[3];
  char d_zip[11];
  char c_first[17];
  char c_middle[3];
  char c_street_1[21];
  char c_street_2[21];
  char c_city[21];
  char c_state[3];
  char c_zip[11];
  char c_phone[17];
  char c_since[11];
  char c_credit[3];
  double c_credit_lim;
  double c_discount;
  double c_balance;
  char c_data[201];
};

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

struct ord_inf {
  int item_cnt;
  int w_id;
  int d_id;
  int c_id;
  char c_last[17];

  char c_first[17];
  char c_middle[3];
  double c_balance;
  int o_id;
  char o_entry_d[31];
  int o_carrier_id;
  struct ord_itm_inf o_items[15];
};

/* For queueing the DLVY Tx. */
struct req_struct {
  int w_id;
  int o_carrier_id;
  int startq_sec;    /** Time the Tx **/
  int startq_usec;   /** was queued **/
  int status;
};

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

struct stock_inf {
  int w_id;
```

```
  int d_id;
  int threshold;
  int low_stock;
};

/**
 ** Shared defns. of TUXEDO vars.
 ** These are defined in shared_tux_vars.c
 **/
/* For NewOrder Txn. */
extern struct newo_inf *NewO;
extern TPSVCINFO *NewOrqst;

/* For Payment Txn. */
extern struct pay_inf *PayM;
extern TPSVCINFO *PayMrqst;

/* For OrderStat Txn. */
extern struct ord_inf *OrdS;
extern TPSVCINFO *OrdSrqst;

/* For STOCK: */
extern struct stock_inf *StkL;
extern TPSVCINFO *StkLrqst;

/* For Delivery Txn. */
extern struct req_struct *DelY;
extern char outbuf[1024];          /* Buffer for
results file */
extern int tx_count;           /* Transaction
counter */
extern FILE *DelYfp;               /* FP to write
out DeliveryLog */
extern TPSVCINFO *DelYrqst;
```

Stored Procedures:

```
#####################################################
##############################
#
# 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       tinyint,

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

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

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

        @s_quantity     smallint,
        @s_ytd          int,            @s_order_cnt
        int,
        @s_dist         char(24),       @s_data
        char(50),

        @ol_number      tinyint,        @o_id
        int,
        @o_entry_d      datetime,       @b_g
        char(1)

declare @0  tinyint,   @1  tinyint,   @2  tinyint,
        @one smallint, @ten smallint,
        @ol_qty_smallint smallint

declare c_no_wdc CURSOR FOR
        SELECT  w_tax, d_tax, d_next_o_id,
                c_last, c_discount, c_credit
                ,0,1,1,10,1,0
                ,getdate()
                FROM    district HOLDLOCK,
                        warehouse HOLDLOCK,
                        customer (index c_clu
prefetch 2 lru) HOLDLOCK
                WHERE   d_w_id  = @w_id
                AND     d_id    = @d_id
                AND     w_id    = d_w_id
                AND     c_w_id  = d_w_id
                AND     c_d_id  = d_id
                AND     c_id    = @c_id
                FOR UPDATE OF d_next_o_id
```

```
begin

        begin transaction NO

        OPEN c_no_wdc
        FETCH c_no_wdc INTO
               @w_tax, @d_tax, @o_id,
               @c_last, @c_discount, @c_credit
               ,@0, @1, @one, @ten, @commit_flag,
@ol_number
               ,@o_entry_d
        UPDATE  district
               SET     d_next_o_id = @o_id + 1
               WHERE CURRENT OF c_no_wdc
        CLOSE c_no_wdc

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

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

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

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

              if (@@rowcount = 0)
                 begin
                   select @commit_flag = 0
                   select NULL, NULL, NULL, NULL
                   continue
                 end
          /*Otherwise if the item is found */
                 update stock
                     set s_ytd = s_ytd + @ol_qty,
                          @s_quantity = s_quantity -
@ol_qty +
                                case when
(s_quantity - @ol_qty < 10)
                                then 91 else 0 end,
                                s_quantity = s_quantity -
@ol_qty +
                                      case when
(s_quantity - @ol_qty < 10)
                                      then 91 else 0 end,
                               s_order_cnt = s_order_cnt
+ 1,
                               @s_data = s_data,
                               @s_dist = case @d_id
                                     when 1 then
s_dist_01
                                     when 2 then
s_dist_02
                                     when 3 then
s_dist_03
                                     when 4 then
s_dist_04
                                     when 5 then
s_dist_05
                                     when 6 then
s_dist_06
                                     when 7 then
s_dist_07
                                     when 8 then
s_dist_08
                                     when 9 then
s_dist_09
                                     when 10 then
s_dist_10
                                     end
                               where s_w_id = @w_id and
                                     s_i_id = @i_id
               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_qty * @i_price, @s_dist)


          /* send line-item data to client */
          select
             @i_name,
             @s_quantity,
             @i_price,
             b_g= case when((patindex("%ORIGINAL%",
@i_data) > 0) and

(patindex("%ORIGINAL%", @s_data) > 0))
                          then "B" else "G" end

       end /* while */

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

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

```
                rollback transaction NO

        select          /* Return to client */
                @w_tax, @d_tax, @o_id, @c_last,
                @c_discount, @c_credit, @o_entry_d
end
go

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

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

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

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

        @s_quantity     smallint,
        @s_ytd          int,            @s_order_cnt
        int,
        @s_dist         char(24),       @s_data
        char(50),
        @s_remote_cnt   int,            @remote
        int,

        @ol_number      tinyint,        @o_id
        int,
        @o_entry_d      datetime,       @b_g
        char(1)

declare @0  tinyint,   @1  tinyint,
        @one smallint, @ten smallint,
        @ol_qty_smallint smallint
```

```
declare c_no_wdc CURSOR FOR
        SELECT  w_tax, d_tax, d_next_o_id,
                c_last, c_discount, c_credit
                ,0,1,1,10,1,0
                ,getdate()
                FROM    district HOLDLOCK,
                        warehouse HOLDLOCK,
                        customer (index c_clu
prefetch 2 lru) HOLDLOCK
                WHERE   d_w_id  = @w_id
                AND     d_id    = @d_id
                AND     w_id    = d_w_id
                AND     c_w_id  = d_w_id
                AND     c_d_id  = d_id
                AND     c_id    = @c_id
                FOR UPDATE OF d_next_o_id

begin

        begin transaction NO

        OPEN c_no_wdc
        FETCH c_no_wdc INTO
                @w_tax, @d_tax, @o_id,
                @c_last, @c_discount, @c_credit
                ,@0, @1, @one, @ten, @commit_flag,
@ol_number
                ,@o_entry_d
        UPDATE  district
                SET     d_next_o_id = @o_id + 1
                WHERE CURRENT OF c_no_wdc
        CLOSE c_no_wdc

        while (@ol_number < @o_ol_cnt) begin
                SELECT @ol_number = @ol_number + 1
                ,@i_id = case @ol_number
                 when 1 then @i_id2
                  when 2 then @i_id3
                  when 3 then @i_id4
                  when 4 then @i_id5
                  when 5 then @i_id6
                  when 6 then @i_id7
                  when 7 then @i_id8
                  when 8 then @i_id9
                  when 9 then @i_id10
                  when 10 then @i_id11
                  when 11 then @i_id12
                  when 12 then @i_id13
                  when 13 then @i_id14
                  when 14 then @i_id15
                  else @i_id
                  end
                , @ol_qty = case @ol_number
                  when 1 then @ol_qty2
                  when 2 then @ol_qty3
                  when 3 then @ol_qty4
                  when 4 then @ol_qty5
                  when 5 then @ol_qty6
                  when 6 then @ol_qty7
                  when 7 then @ol_qty8
                  when 8 then @ol_qty9
                  when 9 then @ol_qty10
                  when 10 then @ol_qty11
                  when 11 then @ol_qty12
                  when 12 then @ol_qty13
                  when 13 then @ol_qty14
                  when 14 then @ol_qty15
                  else @ol_qty
                  end
                ,@s_w_id = case @ol_number
                 when 1 then @s_w_id2
                  when 2 then @s_w_id3
                  when 3 then @s_w_id4
                  when 4 then @s_w_id5
                  when 5 then @s_w_id6
                  when 6 then @s_w_id7
                  when 7 then @s_w_id8
                  when 8 then @s_w_id9
                  when 9 then @s_w_id10
                  when 10 then @s_w_id11
```

```
                when 11 then @s_w_id12
                when 12 then @s_w_id13
                when 13 then @s_w_id14
                when 14 then @s_w_id15
                else @s_w_id
                end

        /* convert c_no_is cursor to a simple
select */
        /* get item data (no one update item) */
               select @i_price = i_price,
                      @i_name = i_name ,
                      @i_data = i_data
                 from item HOLDLOCK
                 where i_id = @i_id

              if (@@rowcount = 0)
                 begin
                       select @commit_flag = 0
                       select NULL, NULL, NULL,
NULL
                       continue
                 end
         /* Otherwise if the item is found */
              update stock
                set s_ytd = s_ytd + @ol_qty,
                    @s_quantity = s_quantity -
@ol_qty +
                                  case when
(s_quantity - @ol_qty < 10)
                                  then 91 else 0 end,
                 s_quantity = s_quantity - @ol_qty
+
                 case when (s_quantity - @ol_qty
< 10)
                   then 91 else 0 end,
                 @s_data = s_data,
                 @s_dist = case @d_id
                         when 1 then s_dist_01
                         when 2 then s_dist_02
                         when 3 then s_dist_03
                         when 4 then s_dist_04
                         when 5 then s_dist_05
                         when 6 then s_dist_06
                         when 7 then s_dist_07
                         when 8 then s_dist_08
                         when 9 then s_dist_09
                         when 10 then s_dist_10
                         end,
                 s_order_cnt = s_order_cnt + 1,
                 s_remote_cnt = s_remote_cnt +
                 case when (@s_w_id = @w_id)
                        then 0 else 1 end
              where s_w_id = @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_qty * @i_price, @s_dist)

               /* send line-item to client */
               select
                  @i_name,
                @s_quantity,
                @i_price,
                b_g = case when
((patindex("%ORIGINAL%", @i_data) > 0) and

(patindex("%ORIGINAL%", @s_data) > 0))
                      then "B" else "G" end
       end /* while */

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

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

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

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

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

declare @c_first       char(16),      @c_middle
       char(2),
       @c_street_1    char(20),      @c_street_2
       char(20),
       @c_city        char(20),      @c_state
       char(2),
       @c_zip         char(9),       @c_phone
       char(16),
       @c_since       datetime,      @c_credit
       char(2),
       @c_credit_lim  numeric(12,0), @c_balance
       float,
       @c_discount    real,
       @1             smallint,
       @data1         char(250),     @data2
       char(250),
       @c_data_1      char(250),     @c_data_2
       char(250)

declare @screen_data   char(200),     @today
       datetime
```

```
declare c_pay_wd CURSOR FOR
        SELECT  w_street_1, w_street_2, w_city,
                w_state, w_zip, w_name, w_ytd,
                d_street_1, d_street_2, d_city,
                d_state, d_zip, d_name, d_ytd
        FROM    district HOLDLOCK,
                warehouse HOLDLOCK
        WHERE   d_w_id  = @w_id
        AND     d_id    = @d_id
        AND     w_id    = d_w_id
        FOR UPDATE OF w_ytd, d_ytd

BEGIN TRANSACTION PID
        OPEN c_pay_wd
        FETCH c_pay_wd INTO
                @w_street_1, @w_street_2, @w_city,
                @w_state, @w_zip, @w_name, @w_ytd,
                @d_street_1, @d_street_2, @d_city,
                @d_state, @d_zip, @d_name, @d_ytd
        UPDATE district
                SET     d_ytd  = @d_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        UPDATE warehouse
                SET w_ytd = @w_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        CLOSE c_pay_wd

        /* Customer data */
        UPDATE customer SET
                  @c_first = c_first
                , @c_middle = c_middle
                , @c_last = c_last
                , @c_street_1 = c_street_1
                , @c_street_2 = c_street_2
                , @c_city = c_city
                , @c_state = c_state
                , @c_zip = c_zip
                , @c_phone = c_phone
                , @c_credit = c_credit
                , @c_credit_lim = c_credit_lim
                , @c_discount = c_discount
                , c_balance = c_balance - @h_amount
                , @c_balance = c_balance - @h_amount
                , c_ytd_payment = c_ytd_payment +
@h_amount
                , c_payment_cnt = c_payment_cnt + 1
                , @c_since = c_since
                , @data1 = c_data1
                , @data2 = c_data2
                , @today = getdate()
        where
                c_id = @c_id
                and c_w_id = @c_w_id
                and c_d_id = @c_d_id

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

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

        /* Create the history record */
        INSERT INTO history (
                h_c_id, h_c_d_id, h_c_w_id, h_d_id,
h_w_id,
                h_date, h_amount, h_data)
        VALUES (
                @c_id, @c_d_id, @c_w_id, @d_id,
@w_id,
                @today, @h_amount, (@w_name + "    "
+ @d_name))
 if (@today = NULL or @d_state = NULL)
        begin
        rollback transaction PID
        select @screen_data = "Invalid Transaction:
Rolled Back"
        end
 else
COMMIT TRANSACTION PID

        select          /* Return to client */
                @c_id,
                @c_last,
                @today,
                @w_street_1,
                @w_street_2,
                @w_city,
                @w_state,
                @w_zip,

                @d_street_1,
                @d_street_2,
                @d_city,
                @d_state,
                @d_zip,

                @c_first,
                @c_middle,
                @c_street_1,
                @c_street_2,
                @c_city,
                @c_state,
                @c_zip,
                @c_phone,
                @c_since,
                @c_credit,
                @c_credit_lim,
                @c_discount,
                @c_balance,
                @screen_data
go
if exists (select * from sysobjects where name =
'payment_byname')
        DROP PROC payment_byname
go
CREATE PROC payment_byname
        @w_id           smallint,       @c_w_id
        smallint,
        @h_amount       float,
        @d_id           tinyint,        @c_d_id
        tinyint,
        @c_last char(16)
as
declare @n      int,    @c_id   int

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

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

```
declare @c_first       char(16),      @c_middle
        char(2),
        @c_street_1    char(20),      @c_street_2
        char(20),
        @c_city        char(20),      @c_state
        char(2),
        @c_zip         char(9),       @c_phone
        char(16),
        @c_since       datetime,      @c_credit
        char(2),
        @c_credit_lim  numeric(12,0), @c_balance
        float,
        @c_discount    real,
        @1             smallint,
        @data1         char(250),     @data2
        char(250),
        @c_data_1      char(250),     @c_data_2
        char(250)

declare @screen_data   char(200),     @today
        datetime

declare c_pay_wd CURSOR FOR
        SELECT  w_street_1, w_street_2, w_city,
                w_state, w_zip, w_name, w_ytd,
                d_street_1, d_street_2, d_city,
                d_state, d_zip, d_name, d_ytd
        FROM    district HOLDLOCK,
                warehouse HOLDLOCK
        WHERE   d_w_id  = @w_id
        AND     d_id    = @d_id
        AND     w_id    = d_w_id
        FOR UPDATE OF w_ytd, d_ytd

declare c_find CURSOR FOR
        SELECT  c_id
        FROM customer (index c_non1 prefetch 2 lru)
HOLDLOCK
        WHERE   c_w_id  =       @c_w_id
        AND     c_d_id  =       @c_d_id
        AND     c_last  =       @c_last
        ORDER BY c_w_id, c_d_id, c_last, c_first,
c_id
        FOR READ ONLY

BEGIN TRANSACTION PNM
        SELECT @n = (count(*)+1)/2
        FROM customer (index c_non1 prefetch 2 lru)
HOLDLOCK
        WHERE   c_w_id = @c_w_id and
                c_d_id = @c_d_id and
                c_last = @c_last
        OPEN c_find
        while (@n>0) begin
                FETCH c_find INTO @c_id
                SELECT @n = @n-1
        end
        CLOSE c_find

        OPEN c_pay_wd
        FETCH c_pay_wd INTO
                @w_street_1, @w_street_2, @w_city,
                @w_state, @w_zip, @w_name, @w_ytd,
                @d_street_1, @d_street_2, @d_city,
                @d_state, @d_zip, @d_name, @d_ytd
        UPDATE district
                SET     d_ytd  = @d_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        UPDATE warehouse
                SET w_ytd = @w_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        CLOSE c_pay_wd

        /* Customer data */
        UPDATE customer SET
                  @c_first = c_first
                , @c_middle = c_middle
                , @c_last = c_last
                , @c_street_1 = c_street_1
                , @c_street_2 = c_street_2
                , @c_city = c_city
                , @c_state = c_state
                , @c_zip = c_zip
                , @c_phone = c_phone
                , @c_credit = c_credit
                , @c_credit_lim = c_credit_lim
                , @c_discount = c_discount
                , c_balance = c_balance - @h_amount
                , @c_balance = c_balance - @h_amount
                , c_ytd_payment = c_ytd_payment +
@h_amount
                , c_payment_cnt = c_payment_cnt + 1
                , @c_since = c_since
                , @data1 = c_data1
                , @data2 = c_data2
                , @today = getdate()
        where
                c_id = @c_id
                and c_w_id = @c_w_id
                and c_d_id = @c_d_id

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

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

        /* Create the history record */
        INSERT INTO history (
                h_c_id, h_c_d_id, h_c_w_id, h_d_id,
h_w_id,
                h_date, h_amount, h_data)
        VALUES (
                @c_id, @c_d_id, @c_w_id, @d_id,
@w_id,
                @today, @h_amount, (@w_name + "    "
+ @d_name))
 if (@today = NULL or @d_state = NULL)
        begin
        rollback transaction PNM
        select @screen_data = "Invalid Transaction:
Rolled Back"
        end
 else
COMMIT TRANSACTION PNM

        select          /* Return to client */
                @c_id,
                @c_last,
                @today,
                @w_street_1,
                @w_street_2,
                @w_city,
                @w_state,
                @w_zip,

                @d_street_1,
                @d_street_2,
                @d_city,
```

```
                @d_state,
                @d_zip,

                @c_first,
                @c_middle,
                @c_street_1,
                @c_street_2,
                @c_city,
                @c_state,
                @c_zip,
                @c_phone,
                @c_since,
                @c_credit,
                @c_credit_lim,
                @c_discount,
                @c_balance,
                @screen_data
go
if exists (select * from sysobjects where name =
'order_status_byid')
        DROP PROC order_status_byid
go
CREATE PROC order_status_byid
        @w_id           smallint,
        @d_id           tinyint,
        @c_id           int
as

DECLARE @o_id           int,
        @o_entry_d      datetime,
        @o_carrier_id   smallint

BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
        SELECT  @o_id = o_id, @o_carrier_id =
o_carrier_id,
                @o_entry_d = o_entry_d
        FROM    orders (index o_clu prefetch 16 lru)
HOLDLOCK
        WHERE   o_w_id  = @w_id
        AND     o_d_id  = @d_id
        AND     o_c_id  = @c_id
        /* ORDER BY o_w_id, o_d_id, o_id */

/* Select order lines for the current order */
        select  /* Return multiple rows to client */
                ol_supply_w_id,
                ol_i_id,
                ol_quantity,
                ol_amount,
                ol_delivery_d
        FROM    order_line HOLDLOCK
        WHERE   ol_o_id = @o_id
        AND     ol_d_id = @d_id
        AND     ol_w_id = @w_id

        select  /* Return single row to client */
                @c_id, c_last, c_first, c_middle,
c_balance,
                @o_id,
                @o_entry_d,
                @o_carrier_id
        FROM    customer (index c_clu prefetch 2
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
declare c_find CURSOR FOR
        SELECT  c_id
        FROM customer (index c_non1 prefetch 2 lru)
HOLDLOCK
        WHERE   c_w_id  =       @w_id
        AND     c_d_id  =       @d_id
        AND     c_last  =       @c_last
        ORDER BY c_w_id, c_d_id, c_last, c_first,
c_id
        FOR READ ONLY

BEGIN TRANSACTION OSNM
        SELECT @n = (count(*)+1)/2
        FROM customer (index c_non1 prefetch 2 lru)
HOLDLOCK
        WHERE   c_w_id = @w_id and
                c_d_id = @d_id and
                c_last = @c_last
        OPEN c_find
        while (@n>0) begin
                FETCH c_find INTO @c_id
                SELECT @n = @n-1
        end
        CLOSE c_find

/* Get the latest order made by the customer */
        SELECT  @o_id = o_id, @o_carrier_id =
o_carrier_id,
                @o_entry_d = o_entry_d
        FROM    orders (index o_clu prefetch 16 lru)
HOLDLOCK
        WHERE   o_w_id  = @w_id
        AND     o_d_id  = @d_id
        AND     o_c_id  = @c_id
        /* ORDER BY o_w_id, o_d_id, o_id */

/* Select order lines for the current order */
        select  /* Return multiple rows to client */
                ol_supply_w_id,
                ol_i_id,
                ol_quantity,
                ol_amount,
                ol_delivery_d
        FROM    order_line HOLDLOCK
        WHERE   ol_o_id = @o_id
        AND     ol_d_id = @d_id
        AND     ol_w_id = @w_id

        select  /* Return single row to client */
                @c_id, c_last, c_first, c_middle,
c_balance,
                @o_id,
                @o_entry_d,
                @o_carrier_id
        FROM    customer (index c_clu prefetch 2
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

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 @ten = 10

    while (@d_id <= @ten) begin

        BEGIN TRANSACTION DEL
        OPEN c_del_no
        FETCH c_del_no INTO @no_o_id

        if (@@sqlstatus != 0)
        begin
                COMMIT TRANSACTION DEL
                select NULL
                CLOSE c_del_no
                select @d_id = @d_id + 1
                continue
        end

        DELETE  FROM new_order
        WHERE   CURRENT OF c_del_no
        CLOSE   c_del_no

/* Using the 'update' enhancement */
        UPDATE  orders
        SET     o_carrier_id = @o_carrier_id
                ,@o_c_id     = o_c_id
                ,@ol_total = 0.0
         WHERE  o_id    = @no_o_id
         AND    o_d_id  = @d_id
         AND    o_w_id  = @w_id

        UPDATE  order_line
        SET     ol_delivery_d = getdate()
                ,@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  customer
        SET     c_balance      = c_balance +
@ol_total,
                        c_delivery_cnt =
c_delivery_cnt + 1
        WHERE   c_id   = @o_c_id
        AND     c_d_id = @d_id
        AND     c_w_id = @w_id

        COMMIT TRANSACTION DEL

        select      /* Return to client */
                    @no_o_id
        select @d_id = @d_id + 1
    end /* while @d_id... */
end
go
if exists ( SELECT name FROM sysobjects WHERE name
= 'stock_level')
    DROP PROC stock_level
go

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

Appendix B: Database Design

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

```
###################
devices.sql
###################
disk init
  name = 'tpcc_log1',
  physname = '/usr/sybase/dbf/ROCKY/dev/tpcc_log1',
  vdevno = 2,
  size = 1024000
go
disk init
  name = 'stock01',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock01',
  vdevno = 3,
  size = 199680
go
disk init
  name = 'stock02',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock02',
  vdevno = 4,
  size = 199680
go
disk init
  name = 'stock03',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock03',
  vdevno = 5,
  size = 199680
go
disk init
  name = 'stock04',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock04',
  vdevno = 6,
  size = 199680
go
disk init
  name = 'stock05',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock05',
  vdevno = 7,
  size = 199680
go
disk init
  name = 'stock06',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock06',
  vdevno = 8,
  size = 199680
go
disk init
  name = 'stock07',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock07',
  vdevno = 9,
  size = 199680
go
disk init
  name = 'stock08',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock08',
  vdevno = 10,
  size = 199680
go
disk init
  name = 'stock09',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock09',
  vdevno = 11,
  size = 199680
go
disk init
  name = 'stock10',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock10',
  vdevno = 12,
  size = 199680
go
disk init
  name = 'stock11',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock11',
  vdevno = 13,
  size = 199680
go
disk init
  name = 'stock12',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock12',
  vdevno = 14,
  size = 199680
go
disk init
  name = 'stock13',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock13',
  vdevno = 15,
  size = 199680
go
disk init
  name = 'stock14',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock14',
  vdevno = 16,
  size = 199680
go
disk init
  name = 'stock15',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock15',
  vdevno = 17,
  size = 199680
go
disk init
  name = 'stock16',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock16',
  vdevno = 18,
  size = 199680
go
disk init
  name = 'stock17',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock17',
  vdevno = 19,
  size = 199680
go
disk init
  name = 'stock18',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock18',
  vdevno = 20,
  size = 199680
go
disk init
  name = 'stock19',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock19',
  vdevno = 21,
  size = 199680
go
disk init
  name = 'stock20',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock20',
  vdevno = 22,
  size = 199680
go
disk init
  name = 'stock21',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock21',
  vdevno = 23,
  size = 199680
go
disk init
  name = 'stock22',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock22',
  vdevno = 24,
  size = 199680
go
disk init
  name = 'stock23',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock23',
  vdevno = 25,
  size = 199680
```

```
go
disk init
  name = 'stock24',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock24',
  vdevno = 26,
  size = 199680
go
disk init
  name = 'stock25',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock25',
  vdevno = 27,
  size = 199680
go
disk init
  name = 'stock26',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock26',
  vdevno = 28,
  size = 199680
go
disk init
  name = 'stock27',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock27',
  vdevno = 29,
  size = 199680
go
disk init
  name = 'stock28',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock28',
  vdevno = 30,
  size = 199680
go
disk init
  name = 'stock29',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock29',
  vdevno = 31,
  size = 199680
go
disk init
  name = 'stock30',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock30',
  vdevno = 32,
  size = 199680
go
disk init
  name = 'stock31',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock31',
  vdevno = 33,
  size = 199680
go
disk init
  name = 'stock32',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock32',
  vdevno = 34,
  size = 199680
go
disk init
  name = 'stock33',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock33',
  vdevno = 35,
  size = 199680
go
disk init
  name = 'stock34',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock34',
  vdevno = 36,
  size = 199680
go
disk init
  name = 'stock35',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock35',
  vdevno = 37,
  size = 199680
go
disk init
  name = 'stock36',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock36',
  vdevno = 38,
  size = 199680
go
disk init
  name = 'stock37',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock37',
  vdevno = 39,
  size = 199680
go
disk init
  name = 'stock38',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock38',
  vdevno = 40,
  size = 199680
go
disk init
  name = 'stock39',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock39',
  vdevno = 41,
  size = 199680
go
disk init
  name = 'stock40',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock40',
  vdevno = 42,
  size = 199680
go
disk init
  name = 'stock41',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock41',
  vdevno = 43,
  size = 199680
go
disk init
  name = 'stock42',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock42',
  vdevno = 44,
  size = 199680
go
disk init
  name = 'stock43',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock43',
  vdevno = 45,
  size = 199680
go
disk init
  name = 'stock44',
  physname = '/usr/sybase/dbf/ROCKY/dev/stock44',
  vdevno = 46,
  size = 512000
go
disk init
  name = 'customer01',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer01',
  vdevno = 47,
  size = 263680
go
disk init
  name = 'customer02',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer02',
  vdevno = 48,
  size = 263680
go
disk init
  name = 'customer03',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer03',
  vdevno = 49,
  size = 263680
go
disk init
  name = 'customer04',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer04',
  vdevno = 50,
  size = 263680
go
disk init
  name = 'customer05',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer05',
  vdevno = 51,
```

```
  size = 263680
go
disk init
  name = 'customer06',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer06',
  vdevno = 52,
  size = 263680
go
disk init
  name = 'customer07',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer07',
  vdevno = 53,
  size = 263680
go
disk init
  name = 'customer08',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer08',
  vdevno = 54,
  size = 263680
go
disk init
  name = 'customer09',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer09',
  vdevno = 55,
  size = 263680
go
disk init
  name = 'customer10',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer10',
  vdevno = 56,
  size = 263680
go
disk init
  name = 'customer11',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer11',
  vdevno = 57,
  size = 263680
go
disk init
  name = 'customer12',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer12',
  vdevno = 58,
  size = 263680
go
disk init
  name = 'customer13',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer13',
  vdevno = 59,
  size = 263680
go
disk init
  name = 'customer14',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer14',
  vdevno = 60,
  size = 263680
go
disk init
  name = 'customer15',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer15',
  vdevno = 61,
  size = 263680
go
disk init
  name = 'customer16',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer16',
  vdevno = 62,
  size = 263680
go
disk init
  name = 'customer17',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer17',
  vdevno = 63,
  size = 263680
go
disk init
  name = 'customer18',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer18',
  vdevno = 64,
  size = 263680
go
disk init
  name = 'customer19',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer19',
  vdevno = 65,
  size = 263680
go
disk init
  name = 'customer20',
  physname =
'/usr/sybase/dbf/ROCKY/dev/customer20',
  vdevno = 66,
  size = 512000
go
disk init
  name = 'cust_idx1',
  physname = '/usr/sybase/dbf/ROCKY/dev/cust_idx1',
  vdevno = 67,
  size = 92160
go
disk init
  name = 'cust_idx2',
  physname = '/usr/sybase/dbf/ROCKY/dev/cust_idx2',
  vdevno = 68,
  size = 92160
go
disk init
  name = 'cust_idx3',
  physname = '/usr/sybase/dbf/ROCKY/dev/cust_idx3',
  vdevno = 69,
  size = 92160
go
disk init
  name = 'cust_idx4',
  physname = '/usr/sybase/dbf/ROCKY/dev/cust_idx4',
  vdevno = 70,
  size = 256000
go
disk init
  name = 'orders1',
  physname = '/usr/sybase/dbf/ROCKY/dev/orders1',
  vdevno = 71,
  size = 55808
go
disk init
  name = 'orders2',
  physname = '/usr/sybase/dbf/ROCKY/dev/orders2',
  vdevno = 72,
  size = 55808
go
disk init
  name = 'orders3',
  physname = '/usr/sybase/dbf/ROCKY/dev/orders3',
  vdevno = 73,
  size = 55808
go
disk init
  name = 'orders4',
  physname = '/usr/sybase/dbf/ROCKY/dev/orders4',
  vdevno = 74,
  size = 256000
go
disk init
  name = 'order_line1',
  physname =
'/usr/sybase/dbf/ROCKY/dev/order_line1',
  vdevno = 75,
```

```
  size = 943104
go
disk init
  name = 'order_line2',
  physname =
'/usr/sybase/dbf/ROCKY/dev/order_line2',
  vdevno = 76,
  size = 943104
go
disk init
  name = 'order_line3',
  physname =
'/usr/sybase/dbf/ROCKY/dev/order_line3',
  vdevno = 77,
  size = 943104
go
disk init
  name = 'order_line4',
  physname =
'/usr/sybase/dbf/ROCKY/dev/order_line4',
  vdevno = 78,
  size = 943104
go
disk init
  name = 'order_line5',
  physname =
'/usr/sybase/dbf/ROCKY/dev/order_line5',
  vdevno = 79,
  size = 943104
go
disk init
  name = 'order_line6',
  physname =
'/usr/sybase/dbf/ROCKY/dev/order_line6',
  vdevno = 80,
  size = 1024000
go
disk init
  name = 'w_d_no_i',
  physname = '/usr/sybase/dbf/ROCKY/dev/w_d_no_i',
  vdevno = 81,
  size = 66560
go
disk init
  name = 'history',
  physname = '/usr/sybase/dbf/ROCKY/dev/history',
  vdevno = 82,
  size = 576000
go
create database tpcc
on master = 100, stock01 = 390, stock02 = 390,
stock03 = 390, stock04 = 390, stock05 = 390,
stock06 = 390, stock07 = 390, stock08 = 390,
stock09 = 390, stock10 = 390, stock11 = 390,
stock12 = 390, stock13 = 390, stock14 = 390,
stock15 = 390, stock16 = 390, stock17 = 390,
stock18 = 390, stock19 = 390, stock20 = 390,
stock21 = 390, stock22 = 390, stock23 = 390,
stock24 = 390, stock25 = 390, stock26 = 390,
stock27 = 390, stock28 = 390, stock29 = 390,
stock30 = 390, stock31 = 390, stock32 = 390,
stock33 = 390, stock34 = 390, stock35 = 390,
stock36 = 390, stock37 = 390, stock38 = 390,
stock39 = 390, stock40 = 390, stock41 = 390,
stock42 = 390, stock43 = 390, stock44 = 1000,
customer01 = 515, customer02 = 515, customer03 =
515, customer04 = 515, customer05 = 515, customer06
= 515, customer07 = 515, customer08 = 515,
customer09 = 515, customer10 = 515, customer11 =
515, customer12 = 515, customer13 = 515, customer14
= 515, customer15 = 515, customer16 = 515,
customer17 = 515, customer18 = 515, customer19 =
515, customer20 = 1000, cust_idx1 = 180, cust_idx2
= 180, cust_idx3 = 180, cust_idx4 = 500, orders1 =
109, orders2 = 109, orders3 = 109, orders4 = 500,
order_line1 = 1842, order_line2 = 1842, order_line3
= 1842, order_line4 = 1842, order_line5 = 1842,
order_line6 = 2000, w_d_no_i = 130, history = 1125
log on tpcc_log1 = 2000
go
use tpcc
go
sp_addsegment Scache , tpcc , w_d_no_i
go
sp_addsegment Scustomer , tpcc , customer01
go
sp_extendsegment Scustomer , tpcc ,  customer02
go
sp_extendsegment Scustomer , tpcc ,  customer03
go
sp_extendsegment Scustomer , tpcc ,  customer04
go
sp_extendsegment Scustomer , tpcc ,  customer05
go
sp_extendsegment Scustomer , tpcc ,  customer06
go
sp_extendsegment Scustomer , tpcc ,  customer07
go
sp_extendsegment Scustomer , tpcc ,  customer08
go
sp_extendsegment Scustomer , tpcc ,  customer09
go
sp_extendsegment Scustomer , tpcc ,  customer10
go
sp_extendsegment Scustomer , tpcc ,  customer11
go
sp_extendsegment Scustomer , tpcc ,  customer12
go
sp_extendsegment Scustomer , tpcc ,  customer13
go
sp_extendsegment Scustomer , tpcc ,  customer14
go
sp_extendsegment Scustomer , tpcc ,  customer15
go
sp_extendsegment Scustomer , tpcc ,  customer16
go
sp_extendsegment Scustomer , tpcc ,  customer17
go
sp_extendsegment Scustomer , tpcc ,  customer18
go
sp_extendsegment Scustomer , tpcc ,  customer19
go
sp_extendsegment Scustomer , tpcc ,  customer20
go
sp_addsegment Scustomer_index , tpcc , cust_idx1
go
sp_extendsegment Scustomer_index , tpcc ,
cust_idx2
go
sp_extendsegment Scustomer_index , tpcc ,
cust_idx3
go
sp_extendsegment Scustomer_index , tpcc ,
cust_idx4
go
sp_addsegment Shistory , tpcc , history
go
sp_addsegment Sorder_line , tpcc , order_line1
go
sp_extendsegment Sorder_line , tpcc ,  order_line2
go
sp_extendsegment Sorder_line , tpcc ,  order_line3
go
sp_extendsegment Sorder_line , tpcc ,  order_line4
go
sp_extendsegment Sorder_line , tpcc ,  order_line5
go
sp_extendsegment Sorder_line , tpcc ,  order_line6
go
sp_addsegment Sorders , tpcc , orders1
go
sp_extendsegment Sorders , tpcc ,  orders2
go
sp_extendsegment Sorders , tpcc ,  orders3
go
sp_extendsegment Sorders , tpcc ,  orders4
go
sp_addsegment Sstock , tpcc , stock01
go
sp_extendsegment Sstock , tpcc ,  stock02
```

```
go
sp_extendsegment Sstock , tpcc ,  stock03
go
sp_extendsegment Sstock , tpcc ,  stock04
go
sp_extendsegment Sstock , tpcc ,  stock05
go
sp_extendsegment Sstock , tpcc ,  stock06
go
sp_extendsegment Sstock , tpcc ,  stock07
go
sp_extendsegment Sstock , tpcc ,  stock08
go
sp_extendsegment Sstock , tpcc ,  stock09
go
sp_extendsegment Sstock , tpcc ,  stock10
go
sp_extendsegment Sstock , tpcc ,  stock11
go
sp_extendsegment Sstock , tpcc ,  stock12
go
sp_extendsegment Sstock , tpcc ,  stock13
go
sp_extendsegment Sstock , tpcc ,  stock14
go
sp_extendsegment Sstock , tpcc ,  stock15
go
sp_extendsegment Sstock , tpcc ,  stock16
go
sp_extendsegment Sstock , tpcc ,  stock17
go
sp_extendsegment Sstock , tpcc ,  stock18
go
sp_extendsegment Sstock , tpcc ,  stock19
go
sp_extendsegment Sstock , tpcc ,  stock20
go
sp_extendsegment Sstock , tpcc ,  stock21
go
sp_extendsegment Sstock , tpcc ,  stock22
go
sp_extendsegment Sstock , tpcc ,  stock23
go
sp_extendsegment Sstock , tpcc ,  stock24
go
sp_extendsegment Sstock , tpcc ,  stock25
go
sp_extendsegment Sstock , tpcc ,  stock26
go
sp_extendsegment Sstock , tpcc ,  stock27
go
sp_extendsegment Sstock , tpcc ,  stock28
go
sp_extendsegment Sstock , tpcc ,  stock29
go
sp_extendsegment Sstock , tpcc ,  stock30
go
sp_extendsegment Sstock , tpcc ,  stock31
go
sp_extendsegment Sstock , tpcc ,  stock32
go
sp_extendsegment Sstock , tpcc ,  stock33
go
sp_extendsegment Sstock , tpcc ,  stock34
go
sp_extendsegment Sstock , tpcc ,  stock35
go
sp_extendsegment Sstock , tpcc ,  stock36
go
sp_extendsegment Sstock , tpcc ,  stock37
go
sp_extendsegment Sstock , tpcc ,  stock38
go
sp_extendsegment Sstock , tpcc ,  stock39
go
sp_extendsegment Sstock , tpcc ,  stock40
go
sp_extendsegment Sstock , tpcc ,  stock41
go
sp_extendsegment Sstock , tpcc ,  stock42
```

```
go
sp_extendsegment Sstock , tpcc ,  stock43
go
sp_extendsegment Sstock , tpcc ,  stock44
go
sp_dropsegment 'default', tpcc , cust_idx1
go
sp_dropsegment 'system', tpcc , cust_idx1
go
sp_dropsegment 'default', tpcc , cust_idx2
go
sp_dropsegment 'system', tpcc , cust_idx2
go
sp_dropsegment 'default', tpcc , cust_idx3
go
sp_dropsegment 'system', tpcc , cust_idx3
go
sp_dropsegment 'default', tpcc , cust_idx4
go
sp_dropsegment 'system', tpcc , cust_idx4
go
sp_dropsegment 'default', tpcc , customer01
go
sp_dropsegment 'system', tpcc , customer01
go
sp_dropsegment 'default', tpcc , customer02
go
sp_dropsegment 'system', tpcc , customer02
go
sp_dropsegment 'default', tpcc , customer03
go
sp_dropsegment 'system', tpcc , customer03
go
sp_dropsegment 'default', tpcc , customer04
go
sp_dropsegment 'system', tpcc , customer04
go
sp_dropsegment 'default', tpcc , customer05
go
sp_dropsegment 'system', tpcc , customer05
go
sp_dropsegment 'default', tpcc , customer06
go
sp_dropsegment 'system', tpcc , customer06
go
sp_dropsegment 'default', tpcc , customer07
go
sp_dropsegment 'system', tpcc , customer07
go
sp_dropsegment 'default', tpcc , customer08
go
sp_dropsegment 'system', tpcc , customer08
go
sp_dropsegment 'default', tpcc , customer09
go
sp_dropsegment 'system', tpcc , customer09
go
sp_dropsegment 'default', tpcc , customer10
go
sp_dropsegment 'system', tpcc , customer10
go
sp_dropsegment 'default', tpcc , customer11
go
sp_dropsegment 'system', tpcc , customer11
go
sp_dropsegment 'default', tpcc , customer12
go
sp_dropsegment 'system', tpcc , customer12
go
sp_dropsegment 'default', tpcc , customer13
go
sp_dropsegment 'system', tpcc , customer13
go
sp_dropsegment 'default', tpcc , customer14
go
sp_dropsegment 'system', tpcc , customer14
go
sp_dropsegment 'default', tpcc , customer15
go
sp_dropsegment 'system', tpcc , customer15
```

```
go
sp_dropsegment 'default', tpcc , customer16
go
sp_dropsegment 'system', tpcc , customer16
go
sp_dropsegment 'default', tpcc , customer17
go
sp_dropsegment 'system', tpcc , customer17
go
sp_dropsegment 'default', tpcc , customer18
go
sp_dropsegment 'system', tpcc , customer18
go
sp_dropsegment 'default', tpcc , customer19
go
sp_dropsegment 'system', tpcc , customer19
go
sp_dropsegment 'default', tpcc , customer20
go
sp_dropsegment 'system', tpcc , customer20
go
sp_dropsegment 'default', tpcc , history
go
sp_dropsegment 'system', tpcc , history
go
sp_dropsegment 'default', tpcc , order_line1
go
sp_dropsegment 'system', tpcc , order_line1
go
sp_dropsegment 'default', tpcc , order_line2
go
sp_dropsegment 'system', tpcc , order_line2
go
sp_dropsegment 'default', tpcc , order_line3
go
sp_dropsegment 'system', tpcc , order_line3
go
sp_dropsegment 'default', tpcc , order_line4
go
sp_dropsegment 'system', tpcc , order_line4
go
sp_dropsegment 'default', tpcc , order_line5
go
sp_dropsegment 'system', tpcc , order_line5
go
sp_dropsegment 'default', tpcc , order_line6
go
sp_dropsegment 'system', tpcc , order_line6
go
sp_dropsegment 'default', tpcc , orders1
go
sp_dropsegment 'system', tpcc , orders1
go
sp_dropsegment 'default', tpcc , orders2
go
sp_dropsegment 'system', tpcc , orders2
go
sp_dropsegment 'default', tpcc , orders3
go
sp_dropsegment 'system', tpcc , orders3
go
sp_dropsegment 'default', tpcc , orders4
go
sp_dropsegment 'system', tpcc , orders4
go
sp_dropsegment 'default', tpcc , stock01
go
sp_dropsegment 'system', tpcc , stock01
go
sp_dropsegment 'default', tpcc , stock02
go
sp_dropsegment 'system', tpcc , stock02
go
sp_dropsegment 'default', tpcc , stock03
go
sp_dropsegment 'system', tpcc , stock03
go
sp_dropsegment 'default', tpcc , stock04
go
sp_dropsegment 'system', tpcc , stock04
```

```
go
sp_dropsegment 'default', tpcc , stock05
go
sp_dropsegment 'system', tpcc , stock05
go
sp_dropsegment 'default', tpcc , stock06
go
sp_dropsegment 'system', tpcc , stock06
go
sp_dropsegment 'default', tpcc , stock07
go
sp_dropsegment 'system', tpcc , stock07
go
sp_dropsegment 'default', tpcc , stock08
go
sp_dropsegment 'system', tpcc , stock08
go
sp_dropsegment 'default', tpcc , stock09
go
sp_dropsegment 'system', tpcc , stock09
go
sp_dropsegment 'default', tpcc , stock10
go
sp_dropsegment 'system', tpcc , stock10
go
sp_dropsegment 'default', tpcc , stock11
go
sp_dropsegment 'system', tpcc , stock11
go
sp_dropsegment 'default', tpcc , stock12
go
sp_dropsegment 'system', tpcc , stock12
go
sp_dropsegment 'default', tpcc , stock13
go
sp_dropsegment 'system', tpcc , stock13
go
sp_dropsegment 'default', tpcc , stock14
go
sp_dropsegment 'system', tpcc , stock14
go
sp_dropsegment 'default', tpcc , stock15
go
sp_dropsegment 'system', tpcc , stock15
go
sp_dropsegment 'default', tpcc , stock16
go
sp_dropsegment 'system', tpcc , stock16
go
sp_dropsegment 'default', tpcc , stock17
go
sp_dropsegment 'system', tpcc , stock17
go
sp_dropsegment 'default', tpcc , stock18
go
sp_dropsegment 'system', tpcc , stock18
go
sp_dropsegment 'default', tpcc , stock19
go
sp_dropsegment 'system', tpcc , stock19
go
sp_dropsegment 'default', tpcc , stock20
go
sp_dropsegment 'system', tpcc , stock20
go
sp_dropsegment 'default', tpcc , stock21
go
sp_dropsegment 'system', tpcc , stock21
go
sp_dropsegment 'default', tpcc , stock22
go
sp_dropsegment 'system', tpcc , stock22
go
sp_dropsegment 'default', tpcc , stock23
go
sp_dropsegment 'system', tpcc , stock23
go
sp_dropsegment 'default', tpcc , stock24
go
sp_dropsegment 'system', tpcc , stock24
```

```
go
sp_dropsegment 'default', tpcc , stock25
go
sp_dropsegment 'system', tpcc , stock25
go
sp_dropsegment 'default', tpcc , stock26
go
sp_dropsegment 'system', tpcc , stock26
go
sp_dropsegment 'default', tpcc , stock27
go
sp_dropsegment 'system', tpcc , stock27
go
sp_dropsegment 'default', tpcc , stock28
go
sp_dropsegment 'system', tpcc , stock28
go
sp_dropsegment 'default', tpcc , stock29
go
sp_dropsegment 'system', tpcc , stock29
go
sp_dropsegment 'default', tpcc , stock30
go
sp_dropsegment 'system', tpcc , stock30
go
sp_dropsegment 'default', tpcc , stock31
go
sp_dropsegment 'system', tpcc , stock31
go
sp_dropsegment 'default', tpcc , stock32
go
sp_dropsegment 'system', tpcc , stock32
go
sp_dropsegment 'default', tpcc , stock33
go
sp_dropsegment 'system', tpcc , stock33
go
sp_dropsegment 'default', tpcc , stock34
go
sp_dropsegment 'system', tpcc , stock34
go
sp_dropsegment 'default', tpcc , stock35
go
sp_dropsegment 'system', tpcc , stock35
go
sp_dropsegment 'default', tpcc , stock36
go
sp_dropsegment 'system', tpcc , stock36
go
sp_dropsegment 'default', tpcc , stock37
go
sp_dropsegment 'system', tpcc , stock37
go
sp_dropsegment 'default', tpcc , stock38
go
sp_dropsegment 'system', tpcc , stock38
go
sp_dropsegment 'default', tpcc , stock39
go
sp_dropsegment 'system', tpcc , stock39
go
sp_dropsegment 'default', tpcc , stock40
go
sp_dropsegment 'system', tpcc , stock40
go
sp_dropsegment 'default', tpcc , stock41
go
sp_dropsegment 'system', tpcc , stock41
go
sp_dropsegment 'default', tpcc , stock42
go
sp_dropsegment 'system', tpcc , stock42
go
sp_dropsegment 'default', tpcc , stock43
go
sp_dropsegment 'system', tpcc , stock43
go
sp_dropsegment 'default', tpcc , stock44
go
sp_dropsegment 'system', tpcc , stock44
```

```
go
use master
go
checkpoint
go




###################
extend_log.sql
###################
use master
go

disk init name="tpcc_log2",
physname="/usr/sybase/dbf/ROCKY/dev/tpcc_log2",
vdevno=83, size=1024000
go
alter database tpcc log on tpcc_log2 = 2000
go
disk init name="tpcc_log3",
physname="/usr/sybase/dbf/ROCKY/dev/tpcc_log3",
vdevno=84, size=1024000
go
alter database tpcc log on tpcc_log3 = 2000
go
disk init name="tpcc_log4",
physname="/usr/sybase/dbf/ROCKY/dev/tpcc_log4",
vdevno=85, size=1024000
go
alter database tpcc log on tpcc_log4 = 2000
go
disk init name="tpcc_log5",
physname="/usr/sybase/dbf/ROCKY/dev/tpcc_log5",
vdevno=86, size=1024000
go
alter database tpcc log on tpcc_log5 = 2000
go
disk init name="tpcc_log6",
physname="/usr/sybase/dbf/ROCKY/dev/tpcc_log6",
vdevno=87, size=1024000
go
alter database tpcc log on tpcc_log6 = 2000
go

disk mirror name="tpcc_log1",
mirror="/usr/sybase/dbf/ROCKY/dev/mirror_tpcc_log1"
, writes=noserial
go
disk mirror name="tpcc_log2",
mirror="/usr/sybase/dbf/ROCKY/dev/mirror_tpcc_log2"
, writes=noserial
go
disk mirror name="tpcc_log3",
mirror="/usr/sybase/dbf/ROCKY/dev/mirror_tpcc_log3"
, writes=noserial
go
disk mirror name="tpcc_log4",
mirror="/usr/sybase/dbf/ROCKY/dev/mirror_tpcc_log4"
, writes=noserial
go
disk mirror name="tpcc_log5",
mirror="/usr/sybase/dbf/ROCKY/dev/mirror_tpcc_log5"
, writes=noserial
go
disk mirror name="tpcc_log6",
mirror="/usr/sybase/dbf/ROCKY/dev/mirror_tpcc_log6"
, writes=noserial
go









#########################
tpcc_tables.sh
#########################
```

```
#!/bin/sh -f

isql -Usa -P$PASSWORD << EOF
/* This script will create all the tables required
for TPC-C benchmark */
/* It will also create some of the indexes. */
sp_dboption tpcc,"select into/bulkcopy",true
go
use tpcc
go
checkpoint
go

if exists ( select name from sysobjects where name
= 'warehouse' )
        drop table warehouse
go
create table warehouse (
        w_id            smallint,
        w_name          char(10),
        w_street_1      char(20),
        w_street_2      char(20),
        w_city          char(20),
        w_state         char(2),
        w_zip           char(9),
        w_tax           real,
        w_ytd           float           /*- Updated by
PID, PNM */
) on Scache
go

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

if exists ( select name from sysobjects where name
= 'customer' )
        drop table customer
go
create table customer (
        c_id            int,
        c_d_id          tinyint,
        c_w_id          smallint,
        c_first         char(16),
        c_middle        char(2),
        c_last          char(16),
        c_street_1      char(20),
        c_street_2      char(20),
        c_city          char(20),
        c_state         char(2),
        c_zip           char(9),
        c_phone         char(16),
        c_since         datetime,
        c_credit        char(2),
        c_credit_lim    numeric(12,0),
        c_discount      real,
        c_delivery_cnt  smallint,
        c_payment_cnt   smallint,       /*- Updated by
PNM, PID */
        c_balance       float,          /*- Updated by
PNM, PID */
        c_ytd_payment   float,          /*- Updated by
PNM, PID */
        c_data1         char(250),      /*- Updated
(?) by PNM, PID */
        c_data2         char(250)       /*- Updated
(?) by PNM, PID */
) on Scustomer
go
create unique clustered index c_clu
        on customer(c_w_id, c_id, c_d_id)
        on Scustomer
go

if exists ( select name from sysobjects where name
= 'history' )
        drop table history
go
create table history (
        h_c_id          int,
        h_c_d_id        tinyint,
        h_c_w_id        smallint,
        h_d_id          tinyint,
        h_w_id          smallint,
        h_date          datetime,
        h_amount        float,
        h_data          char(24)
) on Shistory
go
alter table history partition 16
go

if exists ( select name from sysobjects where name
= 'new_order' )
        drop table new_order
go
create table new_order (
        no_o_id         int,
        no_d_id         tinyint,
        no_w_id         smallint,
) on Scache
go
create unique clustered index no_clu
        on new_order(no_w_id, no_d_id, no_o_id)
        on Scache
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips, 100, new_order)
go

if exists ( select name from sysobjects where name
= 'orders' )
        drop table orders
go
create table orders (
        o_id            int,
        o_c_id          int,
        o_d_id          tinyint,
        o_w_id          smallint,
        o_entry_d       datetime,
        o_carrier_id    smallint,       /*- Updated by
D */
        o_ol_cnt        tinyint,
        o_all_local     tinyint
) on Sorders
go
create unique clustered index o_clu
        on orders(o_w_id, o_d_id, o_id)
        on Sorders
go
dbcc tune(ascinserts, 1, orders)
go
dbcc tune(oamtrips, 100, orders)
go

if exists ( select name from sysobjects where name
= 'order_line' )
        drop table order_line
go
create table order_line (
        ol_o_id         int,
        ol_d_id         tinyint,
```

```
        ol_w_id         smallint,
        ol_number       tinyint,
        ol_i_id         int,
        ol_supply_w_id smallint,
        ol_delivery_d   datetime,       /*- Updated by
D */
        ol_quantity     smallint,
        ol_amount       float,
        ol_dist_info    char(24)
) on Sorder_line
go
create unique clustered index ol_clu
        on order_line(ol_w_id, ol_d_id, ol_o_id,
ol_number)
        on Sorder_line
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go

if exists ( select name from sysobjects where name
= 'item' )
        drop table item
go
create table item (
        i_id            int,
        i_im_id         int,
        i_name          char(24),
        i_price         float,
        i_data          char(50)
) on Scache
go
create unique clustered index i_clu
        on item(i_id)
        on Scache
go

if exists ( select name from sysobjects where name
= 'stock' )
        drop table stock
go
create table stock (
        s_i_id          int,
        s_w_id          smallint,
        s_quantity      smallint,       /*- Updated by
NO */
        s_ytd           int,            /*- Updated by
NO */
        s_order_cnt     smallint,       /*- Updated by
NO */
        s_remote_cnt    smallint,       /*- Updated by
NO */
        s_dist_01       char(24),
        s_dist_02       char(24),
        s_dist_03       char(24),
        s_dist_04       char(24),
        s_dist_05       char(24),
        s_dist_06       char(24),
        s_dist_07       char(24),
        s_dist_08       char(24),
        s_dist_09       char(24),
        s_dist_10       char(24),
        s_data          char(50)
) on Sstock
go
create unique clustered index s_clu
        on stock(s_i_id, s_w_id)
        on Sstock
go
dbcc tune(indextrips, 10, stock)
go

checkpoint
go
EOF



############################
```

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

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

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

create unique clustered index d_clu
        on district(d_w_id, d_id)
        with fillfactor = 1
        on Scache
go
dbcc tune(indextrips, 100, district)
go

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

checkpoint
go
EOF




######################
Loader/load.c
######################
typedef unsigned long  BitVector;
#define WSZ             (sizeof(BitVector)*8)
/* For axposf use WAREBATCH of 144 */
#ifndef WAREBATCH
#ifdef _NTINTEL
#define WAREBATCH       288
#else
#define WAREBATCH       960
#endif
#endif
#define nthbit(map,n)   map[(n)/WSZ] &
(((BitVector)0x1)<< ((n)%WSZ))
#define setbit(map,n)   map[(n)/WSZ] |=
(((BitVector)0x1)<< ((n)%WSZ))

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

int load_item;
int load_warehouse;
int load_district;
int load_history;
int load_orders;
int load_new_order;
int load_order_line;
int load_customer;
int load_stock;
ID  w1, w2;
ID  warehouse;
int batch_size = 1000;
```

```
char   password[10];

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

        /* Setup to use  the dblib version 10 for
numeric datatypes */
         dbsetversion(DBVERSION_100);

        getargs(argn, argv);
        Randomize();

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


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

Warehouse

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


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


int bulk_w;
NTVOID
LoadWarehouse(w1, w2)
        ID w1, w2;
{

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

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

                w_tax = RandomNumber(0, 2000) /
10000.0;
                w_ytd = 300000.00 * 100;

                warehouse_load();

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

NTVOID
begin_warehouse_load()
{
        int     i = 1;

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

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

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

NTVOID
end_warehouse_load()
{
        bulk_close(bulk_w);
}




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

District

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



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

int bulk_d;

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

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

                d_w_id = w_id;
                d_ytd = 30000.00 * 100;
                d_next_o_id = 3001;

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

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

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



NTVOID
begin_district_load()
{
        int     i = 1;

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

        bulk_bind(bulk_d, i++, "d_id", &d_id, ID_T);
        bulk_bind(bulk_d, i++, "d_w_id", &d_w_id,
ID_T);
        bulk_bind(bulk_d, i++, "d_name", d_name,
TEXT_T);
        bulk_bind(bulk_d, i++, "d_street_1",
d_street_1, TEXT_T);
        bulk_bind(bulk_d, i++, "d_street_2",
d_street_2, TEXT_T);
        bulk_bind(bulk_d, i++, "d_city", d_city,
TEXT_T);
        bulk_bind(bulk_d, i++, "d_state", d_state,
TEXT_T);
        bulk_bind(bulk_d, i++, "d_zip", d_zip,
TEXT_T);
        bulk_bind(bulk_d, i++, "d_tax", &d_tax,
FLOAT_T);
        bulk_bind(bulk_d, i++, "d_ytd", &d_ytd,
MONEY_T);
        bulk_bind(bulk_d, i++, "d_next_o_id",
&d_next_o_id, ID_T);
}
NTVOID
district_load()
{
        debug("District %d  w_id=%d\n", d_id,
d_w_id);
        bulk_load(bulk_d);
}
NTVOID
end_district_load()
{
        bulk_close(bulk_d);
}
```

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

Item

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



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

int bulk_i;
NTVOID
LoadItems()
{

        int perm[MAXITEMS+1];
#ifndef _NTINTEL
        int i, r, t;
#endif

        printf("Loading items\n");

        begin_item_load();

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

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

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

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

         item_load();
         }

        end_item_load();
        return;
}



NTVOID
begin_item_load()
{
        int     i = 1;

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

        /* bind the variables to the sybase columns
*/
        bulk_bind(bulk_i, i++, "i_id", &i_id, ID_T);
        bulk_bind(bulk_i, i++, "i_im_id", &i_im_id,
ID_T);
        bulk_bind(bulk_i, i++, "i_name", i_name,
TEXT_T);
        bulk_bind(bulk_i, i++, "i_price", &i_price,
MONEY_T);
```

```
        bulk_bind(bulk_i, i++, "i_data", i_data,
TEXT_T);
}



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

NTVOID
end_item_load()
{
        bulk_close(bulk_i);
}




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

History

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



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

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

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

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

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

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

        h_c_id = c_id;
        h_c_d_id = d_id;
        h_c_w_id = w_id;
        h_d_id = d_id;
        h_w_id = w_id;
        h_amount = 10.0 * 100;
        MakeAlphaString(12, 24, h_data);
        datetime(&h_date);
        history_load();
}


NTVOID
begin_history_load()
{
        int     i = 1;

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

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

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

NTVOID
end_history_load()
{
        bulk_close(bulk_h);
}




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

Customer

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

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

```
TEXT c_credit[2+1] = "?C";
MONEY c_credit_lim = 50000.0 * 100;
FLOAT c_discount;
MONEY c_balance = -10.0 * 100;
MONEY c_ytd_payment = 10.0 * 100;
COUNT c_payment_cnt = 1;
COUNT c_delivery_cnt = 0;
TEXT c_data[500+1];
TEXT c_data1[250+1];
TEXT c_data2[250+1];
ID      len;

int bulk_c;

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

        begin_customer_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {
                Customer(w_id);
                printf("\nLoaded customer for
warehouse %d\n", w_id);
        }
        end_customer_load();
}
NTVOID
Customer(w_id)
        int w_id;
{
        BitVector
badcredit[DIST_PER_WARE][(3000+WSZ-1)/WSZ], * bmp;
        int i, j;
        ID d_id;

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

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

        LastName(i<1000?i:NURandomNumber(255,NURAND_
C,0,999),c_last);
                        MakeAlphaString(8, 16,
c_first);

        MakeAddress(c_street_1,c_street_2,c_city,c_s
tate,c_zip);
                        MakeNumberString(16, 16,
c_phone);
                        MakeAlphaString(300, 500,
c_data);
                        datetime(&c_since);
                        c_credit[0] =
nthbit(badcredit[d_id-1],i) ? 'B' : 'G';
                        c_discount = RandomNumber(0,
5000) / 10000.0;
                        customer_load();
                }
        }
}
NTVOID
begin_customer_load()
{
        int     i = 1;

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

        bulk_bind(bulk_c, i++, "c_id",
&c_id,   ID_T);
        bulk_bind(bulk_c, i++, "c_d_id",
        &c_d_id, ID_T);
        bulk_bind(bulk_c, i++, "c_w_id",
        &c_w_id, ID_T);
        bulk_bind(bulk_c, i++, "c_first",   c_first,
TEXT_T);
        bulk_bind(bulk_c, i++, "c_middle",
c_middle, TEXT_T);
        bulk_bind(bulk_c, i++, "c_last",
        c_last,   TEXT_T);
        bulk_bind(bulk_c, i++, "street_1",
c_street_1 , TEXT_T);
        bulk_bind(bulk_c, i++, "street_2",
c_street_2,  TEXT_T);
        bulk_bind(bulk_c, i++, "c_city",
        c_city,   TEXT_T);
        bulk_bind(bulk_c, i++, "c_state",   c_state,
         TEXT_T);
        bulk_bind(bulk_c, i++, "c_zip",
c_zip,     TEXT_T);
        bulk_bind(bulk_c, i++, "c_phone",   c_phone,
         TEXT_T);
        bulk_bind(bulk_c, i++, "c_since",
&c_since,       DATE_T);
        bulk_bind(bulk_c, i++, "c_credit",
c_credit,      TEXT_T);
        bulk_bind(bulk_c, i++, "c_credit_lim",
&c_credit_lim, MONEY_T);
        bulk_bind(bulk_c, i++, "c_discount",
&c_discount, FLOAT_T);
        bulk_bind(bulk_c, i++, "c_delivery_cnt",
&c_delivery_cnt, COUNT_T);
        bulk_bind(bulk_c, i++,
"c_payment_cnt",&c_payment_cnt, COUNT_T);
        bulk_bind(bulk_c, i++, "c_balance",
&c_balance, MONEY_T);
        bulk_bind(bulk_c, i++, "c_ytd_payment",
&c_ytd_payment, MONEY_T);
        bulk_bind(bulk_c, i++, "c_data_1", c_data1,
TEXT_T);
        bulk_bind(bulk_c, i++, "c_data_2", c_data2,
TEXT_T);
}
NTVOID
customer_load()
{
        debug("c_id=%-5d  d_id=%-5d  w_id=%-5d
c_last=%s\n",
           c_id,       c_d_id,  c_w_id,  c_last);


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

        else
                {
                memcpy(c_data1, c_data, 250+1);
```

```
                strcpy(c_data2,"");
                }

        /* load the data */
        bulk_load(bulk_c);
}
NTVOID
end_customer_load()
{
        bulk_close(bulk_c);
}




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

Order, Order line, New order

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

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

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

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


int  o_bulk;
int ol_bulk;
int no_bulk;


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

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

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

                printf("\nLoaded order + order_line
for warehouse %d\n", w_id);
        }
        end_order_line_load();
        end_order_load();
}
NTVOID
LoadNew(w1, w2)
        ID w1, w2;
{
        ID w_id;
        ID d_id;

        begin_new_order_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {
                for (d_id = 1;  d_id <=
DIST_PER_WARE;  d_id++)
                {
                        no_d_id = d_id;
                        no_w_id = w_id;
                        for (no_o_id=2101;  no_o_id
<= ORD_PER_DIST; no_o_id++)
                                new_order_load();
                }
                printf("\nLoaded new_order for
warehouse %d\n", w_id);
        }
        end_new_order_load();
}
NTVOID
Orders(w_id, d_id)
        ID w_id;
        ID d_id;
{
        int cust[ORD_PER_DIST+1];
        int ol_cnt[ORD_PER_DIST+1], sum;
        ID ol;

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

        RandomPermutation(cust, ORD_PER_DIST);


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

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

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

        for (o_id = 1;  o_id <= ORD_PER_DIST;
o_id++)
        {
        o_c_id = cust[o_id];
        o_d_id = d_id;
        o_w_id = w_id;
        datetime(&o_entry_d);
        if (o_id <= 2100)
        o_carrier_id = RandomNumber(1,10);
        else o_carrier_id = -1;
        o_ol_cnt = ol_cnt[o_id];
        /* o_ol_cnt = RandomNumber(5, 15); */
        o_all_local = 1;
```

```
        order_load();

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

        }
}



NTVOID
OrderLine(ol)
        ID ol;
{

        ol_o_id = o_id;
        ol_d_id = o_d_id;
        ol_w_id = o_w_id;
        ol_number = ol;
        ol_i_id = RandomNumber(1, MAXITEMS);
        ol_supply_w_id = o_w_id;
        if (o_id <= 2100) {
          ol_delivery_d = o_entry_d;
         } else {
          ol_delivery_d.x[0] = 0;
          ol_delivery_d.x[1] = 0;
         }

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

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

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



NTVOID
begin_order_load()
{
        int     i = 1;

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

        bulk_bind(o_bulk, i++, "o_id", &o_id, ID_T);
        bulk_bind(o_bulk, i++, "o_c_id", &o_c_id,
ID_T);
        bulk_bind(o_bulk, i++, "o_d_id", &o_d_id,
ID_T);
        bulk_bind(o_bulk, i++, "o_w_id", &o_w_id,
ID_T);
        bulk_bind(o_bulk, i++, "o_entry_d",
&o_entry_d, DATE_T);
        bulk_bind(o_bulk, i++, "o_carrier_id",
&o_carrier_id, ID_T);
        bulk_bind(o_ol_cnt, i++, "o_ol_cnt",
&o_ol_cnt, COUNT_T);
        bulk_bind(o_all_local, i++, "o_all_local",
&o_all_local, LOGICAL_T);
}
NTVOID
order_load()
{
        debug("o_id=%d o_c_id=%d  count=%d\n", o_id,
o_c_id, o_ol_cnt);
        bulk_load(o_bulk);
}
NTVOID
end_order_load()
{
        bulk_close(o_bulk);
}



NTVOID
begin_order_line_load()
{
        int     i = 1;

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

        bulk_bind(ol_bulk, i++, "ol_o_id", &ol_o_id,
ID_T);
        bulk_bind(ol_bulk, i++, "ol_d_id", &ol_d_id,
ID_T);
        bulk_bind(ol_bulk, i++, "ol_w_id", &ol_w_id,
ID_T);
        bulk_bind(ol_bulk, i++, "ol_number",
&ol_number, ID_T);
        bulk_bind(ol_bulk, i++, "ol_i_id", &ol_i_id,
ID_T);
        bulk_bind(ol_bulk, i++, "ol_supply_w_id",
&ol_supply_w_id, ID_T);
        bulk_bind(ol_bulk, i++, "ol_delivery_d",
&ol_delivery_d, DATE_T);
        bulk_bind(ol_bulk, i++, "ol_quantity",
&ol_quantity, COUNT_T);
        bulk_bind(ol_bulk, i++, "ol_amount",
&ol_amount, MONEY_T);
        bulk_bind(ol_bulk, i++, "ol_dist_info",
ol_dist_info, TEXT_T);
}
NTVOID
order_line_load()
{
        static int ol_count = 0;
        debug("  ol_o_id=%d  ol_number=%d
ol_amount=%g\n",
                ol_o_id,      ol_number,
        ol_amount);
        bulk_load(ol_bulk);
}

NTVOID
end_order_line_load()
{
        bulk_close(ol_bulk);
}
NTVOID
begin_new_order_load()
{
        int     i = 1;

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

        bulk_bind(no_bulk, i++, "no_o_id", &no_o_id,
ID_T);
        bulk_bind(no_bulk, i++, "no_d_id", &no_d_id,
ID_T);
        bulk_bind(no_bulk, i++, "no_w_id", &no_w_id,
ID_T);
}
NTVOID
new_order_load()
{
        debug("  no_o_id=%d  \n", no_o_id);
        bulk_load(no_bulk);
}

NTVOID
end_new_order_load()
{
        bulk_close(no_bulk);
```

```
}


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

Stock

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


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

int bulk_s;

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

#ifdef _NTINTEL
/* On NT stack overflow happens when you define the
following on stack
*/
        BitVector
original[WAREBATCH][((MAXITEMS+(WSZ-1))/WSZ)] ;
#endif
NTVOID
LoadStock(w1, w2)
        ID w1, w2;
{
        ID w_id;

#ifdef _NTINTEL
        BitVector * bmp;
#else
        BitVector
original[WAREBATCH][((MAXITEMS+(WSZ-1))/WSZ)], *
bmp;
#endif

        int w, i, j;

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

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

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

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

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



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

NTVOID
end_stock_load()
{
        bulk_close(bulk_s);
}


NTVOID
test(){}

NTVOID
getargs(argc, argv)

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

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


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

        }

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

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

        /* Check if warehouse is within the range */
        if (w1 <= 0 || w2 > 1000 || w1 > w2)
        {
                printf("Warehouse id is out of
range\n");
                exit(0);
        }
}

#ifndef _NTINTEL
double drand48();
#endif
NTVOID
MakeAddress(str1, str2, city, state, zip)
        TEXT str1[20+1];
        TEXT str2[20+1];
        TEXT city[20+1];
        TEXT state[2+1];
        TEXT zip[9+1];
{
        MakeAlphaString(10,20,str1);
        MakeAlphaString(10,20,str2);
        MakeAlphaString(10,20,city);
        MakeAlphaString(2,2,state);
        MakezipString(0,9999,zip);

        /* Changed for TPCC V 3.0 */
        strcat(zip, "11111");
```

```
}


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

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


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

        length = RandomNumber(min, max);

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

        return length;
}

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

        length = 4;

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

        return length;
}



int MakeAlphaString(min, max, str)
        int min;
        int max;
        TEXT str[];
{
        static char character[] =
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQ
RSTUVWXYZ0123456789";
        int length;
        int i;

        length = RandomNumber(min, max);

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

        return length;
}

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

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

        pos = RandomNumber(0,len-8);

        str[pos+0] = 'O';
        str[pos+1] = 'R';
        str[pos+2] = 'I';
        str[pos+3] = 'G';
        str[pos+4] = 'I';
        str[pos+5] = 'N';
        str[pos+6] = 'A';
        str[pos+7] = 'L';
}



NTVOID
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;
        }
}




#ifdef _NTINTEL
NTVOID Randomize()
{
        srand(time(0)+_getpid());
}
#else
int Randomize()
{
        srand48(time(0)+getpid());
}
#endif



int RandomNumber(min, max)
        int min;
        int max;
{
        int r;
#ifdef _NTINTEL
        r = (int)((float)rand()/(float)(RAND_MAX +
1) * (max - min + 1)) + min;
#else
        r = (int)(drand48() * (max - min + 1)) +
min;
#endif
```

```
        return r;
}


int NURandomNumber(a, c, min, max)
        int a;
        int c;
        int min;
        int max;
{
        int r;

        r = ((RandomNumber(0, a) | RandomNumber(min,
max)) + c)
        % (max - min + 1) + min;

        return r;
}



####################
Loader/bulk_sybase.c
####################
/*************************************************
**************
*************************************************
***************

Sybase Specific Routines

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

#include <stdio.h>
#ifdef _NTINTEL
#include <sys/timeb.h>
#include <io.h>
#else
#include <sys/time.h>
#endif
#include <string.h>
#include "loader.h"
NTVOID
datetime(date)
        DBDATETIME *date;
{
#ifdef _NTINTEL
        time_t time1;
        time(&time1);
        date->dtdays = time1 / (60*60*24)
                        + (1970-1900)*365 + (1970-
1900)/4;
        date->dttime = (time1 % (60*60*24))*300 ;
#else
        struct timeval time;
        gettimeofday(&time, NULL);
        date->dtdays = time.tv_sec / (60*60*24)
                        + (1970-1900)*365 + (1970-
1900)/4;
        date->dttime = (time.tv_sec %
(60*60*24))*300
                        + time.tv_usec*300/1000000;
#endif
}


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

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

#define MAXOPENS 10

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

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

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

        /* Install an error and Message handler */
#ifdef _NTINTEL
        dbmsghandle((MHANDLEFUNC)msg_handler);
        dberrhandle((EHANDLEFUNC)err_handler);
#else
        dbmsghandle(msg_handler);
        dberrhandle(err_handler);
#endif

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

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

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

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

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

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

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

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

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

```
        return db;
}


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

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

NTVOID
bulk_load(db)
        int db;
{
        count[db]++;
        if (bcp_sendrow(dbproc[db]) != SUCCEED)
                printf("bulk_load: Can't load
row\n");
        if (count[db]%batch_size == 0 &&
(bcp_batch(dbproc[db]) == -1))
        printf("bulk_load: Can't post rows\n");
#ifdef _NTINTEL
        if (count[db]%1000 == 0) _write(1,".",1);
        if (count[db]%50000 == 0) _write(1,"\n",1);
#else
        if (count[db]%1000 == 0) write(1,".",1);
        if (count[db]%50000 == 0) write(1,"\n",1);
#endif
}
NTVOID
bulk_close(db)
        int db;
{
        if (bcp_done(dbproc[db]) == -1)
                printf("Problems completing the bulk
copy.\n");
        dbproc[db] = NULL;
        if (count[db] >= 1000) write(1,"\n",1);
}



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

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

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

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

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

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


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

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

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

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

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

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

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

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



##################
loader/loader.h
##################
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED

#include <sybfront.h>
#include <sybdb.h>
#ifdef _NTINTEL
#include <sys/timeb.h>
#include <stdlib.h>
#include <process.h>
#endif
#include <time.h>


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

#define NURAND_C 123

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

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

typedef struct timeval TIME;




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

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

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

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

#endif /* TPCC_INCLUDED */

#ifdef _NTINTEL
#define NTVOID void
#else
#define NTVOID
#endif
#ifdef _NTINTEL
#define drand48() rand()
void LoadWarehouse() ;
void begin_warehouse_load() ;
void warehouse_load() ;
void end_warehouse_load() ;
void LoadDistrict() ;
void begin_district_load() ;
void district_load() ;
void end_district_load() ;
void LoadItems() ;
void begin_item_load() ;
void item_load() ;
void end_item_load() ;
void LoadHist() ;
void LoadCustHist() ;
void begin_history_load() ;
void history_load() ;
void end_history_load() ;
void LoadCustomer() ;
void Customer() ;
void begin_customer_load() ;
void customer_load() ;
void end_customer_load() ;
void LoadOrd() ;
void LoadNew() ;
void Orders() ;
void OrderLine() ;
void NewOrder() ;
void begin_order_load() ;
void order_load() ;
void end_order_load() ;
void begin_order_line_load() ;
void order_line_load() ;
void end_order_line_load() ;
void begin_new_order_load() ;
void new_order_load() ;
void end_new_order_load() ;
void LoadStock() ;
void begin_stock_load() ;
void stock_load() ;
void end_stock_load() ;
void test() ;
void getargs() ;
void MakeAddress() ;
void LastName() ;
int MakeNumberString() ;
int MakezipString() ;
int MakeAlphaString() ;
void Original() ;
void RandomPermutation() ;
void Randomize() ;
int RandomNumber() ;
int NURandomNumber() ;
void datetime() ;
int bulk_open() ;
void bulk_bind() ;
void bulk_null() ;
void bulk_non_null() ;
void bulk_load() ;
void bulk_close() ;
#endif



###################
loader/makefile
###################
# @(#) Makefile 1.3@(#)
.KEEP_STATE:

SQL_RELEASE     =       /usr/sybase/11

INCLUDE         =       $(SQL_RELEASE)/include
LIBDIR          =       $(SQL_RELEASE)/lib
OPT             =       -O -n32
```

```
CC              =       cc

LDFLAGS         =       $(LIBDIR)/libsybdb.a -lm -lc
$(NSL)

CFLAGS          =       $(OPT) $(DEBUG) -I$(INCLUDE)
$(FLAGS)

LOAD_OBJS       =       error.o load.o bulk_sybase.o
L_HEADERS       =       loader.h

all:    load

load:   $(LOAD_OBJS) $(L_HEADERS)
        $(CC) $(DEBUG) $(OPT) -o load $(FLAGS)
$(LOAD_OBJS) $(LDFLAGS)

clean:
        rm -f *.o

panic:
        rm -f *.o load
```

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

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

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

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

        }

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

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

Appendix C: Tunable Parameters

This Appendix contains the IRIX™ configuration parameters and the RDBMS configuration.

Configuration Parameters for the Clients

```
 group: lockd (dynamically changeable)
        max_lockd_procs = 6666 (0x1a0a)
        nlm_callback_retry = 3 (0x3)
        nlm_maxdupreqs = 0 (0x0)
        nlm_granted_timeout = 1 (0x1)
        lock_share_requests = 0 (0x0)
        lockd_grace_period = 45 (0x2d)

 group: pipefs (dynamically changeable)
        svr3pipe = 1 (0x1)

 group: if_ptr (dynamically changeable)
        mtr_s16Mb = 1 (0x1)
        mtr_arb_enabled = 1 (0x1)
        mtr_participant_claim_token = 1 (0x1)
        mtr_batype = 2 (0x2)
        mtr_mtu = 4472 (0x1178)
        mtr_bc_mask = 32768 (0x8000)
        mtr_max_tx_slots = 16 (0x10)
        mtr_max_rx_slots = 16 (0x10)

 group: usync (statically changeable)
        usync_max_objs = 8192 (0x2000)

 group: utrace (dynamically changeable)
        utrace_mask = 1048383 (0xfff3f) ll

 group: ithread (statically changeable)
        default_timeout_pri = 2 (0x2)
        initial_timeout_pri = 255 (0xff)
        default_intr_pri = 200 (0xc8)
        tape_intr_pri = 205 (0xcd)
        disk_intr_pri = 205 (0xcd)
        scsi_intr_pri = 205 (0xcd)
        network_intr_pri = 215 (0xd7)
        parallel_intr_pri = 225 (0xe1)
        serial_intr_pri = 225 (0xe1)
        graphics_intr_pri = 235 (0xeb)
        video_intr_pri = 245 (0xf5)
        audio_intr_pri = 245 (0xf5)

 group: sthread (statically changeable)
        cl_async_pri = 90 (0x5a)
symbol: hbt_pri not found
symbol: sd_pri not found
        xbox_poll_pri = 90 (0x5a)
        dpipe_pri = 90 (0x5a)
        munldd_pri = 90 (0x5a)
symbol: xpc_test_pri not found
        io_init_pri = 90 (0x5a)
        tpisockd_pri = 90 (0x5a)
        recovery_pri = 90 (0x5a)
        cmsd_pri = 90 (0x5a)
        coalesced_pri = 90 (0x5a)
        sched_pri = 90 (0x5a)
        thand_pri = 90 (0x5a)
        async_call_pri = 90 (0x5a)
        unmountd_pri = 90 (0x5a)
        xfsd_pri = 92 (0x5c)
        bdflush_pri = 92 (0x5c)
        vfs_sync_pri = 92 (0x5c)
        xlvd_pri = 94 (0x5e)
        bpsqueue_pri = 94 (0x5e)
        exim_pri = 96 (0x60)
        shaked_pri = 98 (0x62)
        vhand_pri = 98 (0x62)
        rmonqd_pri = 98 (0x62)
        pdflush_pri = 98 (0x62)
        batchd_pri = 98 (0x62)
        reaper_pri = 100 (0x64)
        onesec_pri = 90 (0x5a)
        klogthrd_pri = 98 (0x62)
        du_conpoll_pri = 255 (0xff)
        console_pri = 255 (0xff)

group: io (statically changeable)
        hwgraph_num_dev = 16384 (0x4000)

group: shm (statically changeable)
        sshmseg = 100 (0x64)
        shmmni = 100 (0x64)
        shmmin = 1 (0x1) ll
        shmmax = 1717986919 (0x66666667) ll

group: vme_dma (statically changeable)
        nvme32_dma = 64 (0x40)

group: lpage_watermarks (dynamically changeable)
        coalesced_run_period = 300 (0x12c)
        percent_totalmem_16m_pages = 0 (0x0)
        percent_totalmem_4m_pages = 0 (0x0)
        percent_totalmem_1m_pages = 0 (0x0)
        percent_totalmem_256k_pages = 0 (0x0)
        percent_totalmem_64k_pages = 0 (0x0)
        percent_totalmem_16k_pages = 0 (0x0)

group: large_pages (statically changeable)
        large_pages_enable = 1 (0x1)
        nlpages_16m = 0 (0x0)
        nlpages_4m = 0 (0x0)
        nlpages_1m = 0 (0x0)
        nlpages_256k = 0 (0x0)
        nlpages_64k = 0 (0x0)

group: internal (statically changeable)
        dumplo = 0 (0x0)
        errbuf_cpusz = 2048 (0x800)
        putbuf_cpusz = 2048 (0x800)
        conbuf_cpusz = 2048 (0x800)
        errbuf_maxsz = 16384 (0x4000)
        putbuf_maxsz = 16384 (0x4000)
        conbuf_maxsz = 16384 (0x4000)
        histmax = 0 (0x0)

group: extacct (statically changeable)
        asarrayid = 65535 (0xffff)
        asmachid = 0 (0x0)
        incr_paggid = 1 (0x1) ll
        max_local_paggid = 65535 (0xffff) ll
        min_local_paggid = 1 (0x1) ll
        dflt_paggid = 0 (0x0) ll
        dfltprid = 0 (0x0) ll
        spilen = 64 (0x40)
        sessaf = 1 (0x1)
        do_sessacct = 0 (0x0)
        do_extpacct = 0 (0x0)
        do_procacct = 1 (0x1)

group: mload (statically changeable)
        vfssw_extra = 5 (0x5)
        fmodsw_extra = 20 (0x14)
        cdevsw_extra = 23 (0x17)
        bdevsw_extra = 21 (0x15)

group: bufcache (statically changeable)
        nbuf = 13207 (0x3397)

group: paging (dynamically changeable)
        enable_kaio_fast_path = 1 (0x1)
```

```
        enable_devzero_opt = 1 (0x1)
        cwcluster = 64 (0x40)
        zone_accum_limit = 30 (0x1e)
        shaked_interval = 1 (0x1)
        min_free_pages = 12800 (0x3200)
        min_file_pages = 3971 (0xf83)
        autoup = 10 (0xa)
        dwcluster = 64 (0x40)
        rsshogslop = 0 (0x0)
        rsshogfrac = 99 (0x63)
        tlbdrop = 100 (0x64)
        maxlkmem = 500 (0x1f4)
        minasmem = 50 (0x32)
        minarmem = 50 (0x32)
        vfs_syncr = 30 (0x1e)
        bdflushr = 5 (0x5)
        maxdc = 254 (0xfe)
        maxfc = 254 (0xfe)
        maxsc = 254 (0xfe)
        gpgsmsk = 1 (0x1)
        gpgshi = 3200 (0xc80)
        gpgslo = 2000 (0x7d0)

 group: memsize (statically changeable)
        scache_pool_size = 32 (0x20)
        maxpglst = 254 (0xfe)
        maxdmasz = 257 (0x101)
        syssegsz = 65536 (0x10000)
        maxpmem = 0 (0x0)

 group: timer (statically changeable)
        timetrim = -107106 (0xfffffffffffe5d9e)
        itimer_on_clkcpu = 0 (0x0)
        fasthz = 1250 (0x4e2)

 group: signals (dynamically changeable)
        maxsigq = 128 (0x80)

 group: actions (statically changeable)
        nactions = 2 (0x2)

 group: streams (dynamically changeable)
        strholdtime = 50 (0x32)

 group: streams (statically changeable)
        strpmonmax = 32 (0x20)
        strctlsz = 1024 (0x400)
        strmsgsz = 32768 (0x8000)
        nstrintr = 1024 (0x400)
        nstrpush = 9 (0x9)

 group: numproc (statically changeable)
        ngroups_max = 16 (0x10)
        ncsize = 20200 (0x4ee8)
        ncallout = 5000 (0x1388)
        ndquot = 20200 (0x4ee8)
        nproc = 10000 (0x2710)

 group: resource (statically changeable)
        rlimit_pthread_max = 1024 (0x400) ll
        rlimit_pthread_cur = 1024 (0x400) ll
        rlimit_nofile_max = 2500 (0x9c4) ll
        rlimit_nofile_cur = 200 (0xc8) ll
        rlimit_rss_max = 536870912 (0x20000000) ll
        rlimit_rss_cur = 536870912 (0x20000000) ll
        rlimit_vmem_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_vmem_cur = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_core_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_core_cur = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_stack_max = 536870912 (0x20000000)
ll
        rlimit_stack_cur = 67108864 (0x4000000) ll
        rlimit_data_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_data_cur = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_fsize_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_fsize_cur = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_cpu_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_cpu_cur = 9223372036854775807
(0x7fffffffffffffff) ll

 group: limits (dynamically changeable)
        idbgmaxfuncs = 1200 (0x4b0)
symbol: vnode_free_ratio not found
        maxup = 7000 (0x1b58)
        maxsymlinks = 30 (0x1e)
        nprofile = 100 (0x64)
        maxwatchpoints = 100 (0x64)
        shlbmax = 8 (0x8)
        ncargs = 20480 (0x5000)

 group: misc (dynamically changeable)
        fru_disable_confidence = 90 (0x5a)
        full_fru_analysis = 1 (0x1)
        bridge_rev_b_data_check_disable = 0 (0x0)
        pcibr_wg_enable_rev = 4 (0x4)
        pcibr_prefetch_enable_rev = 3 (0x3)
        gather_craylink_routerstats = 1 (0x1)
        r10k_progress_diff = 200000000 (0xbebc200)
ll
        r10k_check_count = 30000 (0x7530)
        r10k_progress_nmi = 0 (0x0)
        r10k_intervene = 0 (0x0)
        mmap_async_write = 0 (0x0)
        craylnk_bypass_disable = 0 (0x0)
        heartbeat_enable = 0 (0x0)
        dump_level = 4 (0x4)
        module_unld_delay = 5 (0x5)
        ecc_recover_enable = 60 (0x3c)
        sbe_report_cons = 0 (0x0)
        sbe_mfr_override = 0 (0x0)
        sbe_log_errors = 1 (0x1)
        panic_on_sbe = 0 (0x0)
        corepluspid = 0 (0x0)
        r4k_div_patch = 0 (0x0)

 group: misc (statically changeable)
        io4ia_war_enable = 0 (0x0)
        ignore_conveyor_override = 0 (0x0)
        io4ia_userdma_war = 1 (0x1)

 group: switch (dynamically changeable)
        tty_auto_strhold = 0 (0x0)

 group: switch (statically changeable)
        snerrste_pages_per_cube = 1 (0x1)
        xbox_sysctlr_poll_interval = 60 (0x3c)
        clean_shutdown_time = 120 (0x78)
        powerfail_cleanio = 0 (0x0)
        ignore_sysctlr_intr = 0 (0x0)
        ip32_pci_flush_prefetch = 1 (0x1)
        ip32_pci_enable_prefetch = 0 (0x0)
        enable_pci_BIST = 0 (0x0)
        warbits_override = 18446744073709551615
(0xffffffffffffffff)
        spec_udma_war_on = 1 (0x1)
        disable_r10k_log = 0 (0x0)
        prod_assertion_level = 0 (0x0)
        disable_ip25_check = 0 (0x0)
        mload_auto_rtsyms = 1 (0x1)
        xpg4_sticky_dir = 1 (0x1)
        add_kthread_stack = 0 (0x0)
        ip26_allow_ucmem = 0 (0x0)
        reset_limits_on_exec = 1 (0x1)
        restrict_fastprof = 0 (0x0)
        reboot_on_panic = -1 (0xffffffffffffffff)
        use_old_serialnum = 0 (0x0)
        posix_tty_default = 0 (0x0)
        restricted_chown = 0 (0x0)
        nosuidshells = 1 (0x1)

 group: svckudp (statically changeable)
```

```
        svc_maxdupreqs = 0 (0x0)

 group: cachefs (dynamically changeable)
        replacement_timeout = 600 (0x258)
        fileheader_cache_size = 512 (0x200)
        cachefs_max_threads = 5 (0x5)
        cachefs_readahead = 1 (0x1)

 group: snfs (statically changeable)
        nfs_portmon = 0 (0x0)

 group: psema (statically changeable)
        psema_max_cnt = 8192 (0x2000)

 group: numa (statically changeable)
        numa_migr_queue_control_max_page_age = 8
(0x8)
        numa_migr_queue_time_trigger_interval = 1
(0x1)
        numa_migr_queue_time_trigger_enabled = 0
(0x0)
        numa_migr_queue_capacity_trigger_threshold
= 60 (0x3c)
        numa_migr_queue_capacity_trigger_enabled =
0 (0x0)
        numa_migr_queue_traffic_trigger_threshold =
80 (0x50)
        numa_migr_queue_traffic_trigger_enabled = 0
(0x0)
        numa_migr_queue_control_mlimit = 70 (0x46)
        numa_migr_queue_control_interval = 500
(0x1f4)
        numa_migr_queue_control_enabled = 0 (0x0)
        numa_migr_queue_maxlen = 70 (0x46)
        memoryd_enabled = 0 (0x0)
        numa_migr_enqonfail_enabled = 0 (0x0)
        numa_paging_node_freemem_low_threshold = 40
(0x28)
        numa_zone_radius_max = 2 (0x2)
        numa_kernel_replication_ratio = 1 (0x1)
        numa_repl_mem_lowmark = 32 (0x20)
        numa_repl_traffic_highmark_percentage = 80
(0x50)
        numa_repl_control_enabled = 0 (0x0)
        numa_page_replication_enable = 0 (0x0)
        numa_migr_traffic_control_threshold = 80
(0x50)
        numa_migr_traffic_control_interval = 400
(0x190)
        numa_migr_traffic_control_enabled = 0 (0x0)
        numa_migr_unpegging_control_threshold = 90
(0x5a)
        numa_migr_unpegging_control_interval = 0
(0x0)
        numa_migr_unpegging_control_enabled = 1
(0x1)
        mem_tick_base_period = 1 (0x1)
        mem_tick_enabled = 1 (0x1)
        numa_migr_dampening_factor = 90 (0x5a)
        numa_migr_dampening_enabled = 0 (0x0)
        numa_migr_bounce_control_interval = 0 (0x0)
        numa_migr_melt_threshold = 90 (0x5a)
        numa_migr_melt_enabled = 0 (0x0)
        numa_migr_freeze_threshold = 20 (0x14)
        numa_migr_freeze_enabled = 0 (0x0)
        numa_migr_memory_low_threshold = 20 (0x14)
        numa_migr_memory_low_enabled = 1 (0x1)
        numa_migr_min_distance = 1 (0x1)
        numa_refcnt_overflow_threshold = 50 (0x32)
        numa_refcnt_default_mode = 0 (0x0)
        numa_migr_coaldmigr_mech = 0 (0x0)
        numa_migr_user_migr_mech = 0 (0x0)
        numa_migr_auto_migr_mech = 0 (0x0)
        numa_migr_min_maxradius = 2 (0x2)
        numa_migr_vehicle = 1 (0x1)
        numa_migr_threshold_reference = 0 (0x0)
        numa_migr_default_threshold = 50 (0x32)
        numa_migr_default_mode = 2 (0x2)

 group: nfs3 (statically changeable)
        nfs3_default_xfer = 32768 (0x8000)

group: disp (statically changeable)
       slice_size = 10 (0xa)

group: xfs (statically changeable)
       xfs_nfs_io_units = 10 (0xa)

group: uds (statically changeable)
       unpdg_recvspace = 16384 (0x4000)
       unpdg_sendspace = 8192 (0x2000)
       unpst_recvspace = 16384 (0x4000)
       unpst_sendspace = 16384 (0x4000)

group: sem (statically changeable)
       semaem = 16384 (0x4000)
       semvmx = 32767 (0x7fff)
       semopm = 2048 (0x800)
       semmsl = 2048 (0x800)
       semmni = 2048 (0x800)

group: lockmgr (dynamically changeable)
       normal_retry = 1 (0x1)
       first_retry = 1 (0x1)
       GraceWaitTime = 5 (0x5)
       sm_timeout = 5 (0x5)
       portmap_timeout = 5 (0x5)

group: msg (statically changeable)
       msgseg = 376840 (0x5c008)
       msgtql = 16000 (0x3e80)
       msgssz = 128 (0x80)
       msgmni = 10000 (0x2710)
       msgmnb = 307200 (0x4b000)
       msgmax = 32768 (0x8000)

group: efs (dynamically changeable)
       efs_inline = 0 (0x0)

group: net_udp (dynamically changeable)
       udp_recvgrams = 2 (0x2)
       udp_sendspace = 61440 (0xf000)
       udp_ttl = 60 (0x3c)

group: net_tcp (dynamically changeable)
       tcp_recvspace = 61440 (0xf000)
       tcp_sendspace = 61440 (0xf000)
       tcp_2msl = 60 (0x3c)
       tcp_mtudisc = 1 (0x1)
       tcp_maxpersistidle = 7200 (0x1c20)
       tcp_keepintvl = 75 (0x4b)
       tcp_keepidle = 7200 (0x1c20)
       tcp_ttl = 60 (0x3c)

group: net_rsvp (statically changeable)
       ps_num_batch_pkts = 0 (0x0)
       ps_rsvp_bandwidth = 50 (0x32)
       ps_enabled = 1 (0x1)

group: net_mbuf (statically changeable)
       mbmaxpages = 32768 (0x8000)

group: net_ip (dynamically changeable)
       subnetsarelocal = 1 (0x1)
       allow_brdaddr_srcaddr = 0 (0x0)
       ipdirected_broadcast = 0 (0x0)
       ipsendredirects = 1 (0x1)
       ipforwarding = 1 (0x1)
       ipfilterd_inactive_behavior = 1 (0x1)
       icmp_dropredirects = 0 (0x0)

group: network (statically changeable)
       netthread_float = 0 (0x0)
       netthread_pri = 98 (0x62)

group: inpcb (statically changeable)
       udp_hashtablesz = 2048 (0x800)
       tcp_hashtablesz = 16384 (0x4000)
```

Configuration Parameters for the Server

```
group: lockd (dynamically changeable)
       max_lockd_procs = 1400 (0x578)
       nlm_callback_retry = 3 (0x3)
       nlm_maxdupreqs = 0 (0x0)
       nlm_granted_timeout = 1 (0x1)
       lock_share_requests = 0 (0x0)
       lockd_grace_period = 45 (0x2d)

group: pipefs (dynamically changeable)
       svr3pipe = 1 (0x1)

group: if_ptr (dynamically changeable)
       mtr_s16Mb = 1 (0x1)
       mtr_arb_enabled = 1 (0x1)
       mtr_participant_claim_token = 1 (0x1)
       mtr_batype = 2 (0x2)
       mtr_mtu = 4472 (0x1178)
       mtr_bc_mask = 32768 (0x8000)
       mtr_max_tx_slots = 16 (0x10)
       mtr_max_rx_slots = 16 (0x10)

group: usync (statically changeable)
       usync_max_objs = 8192 (0x2000)

group: ithread (statically changeable)
       default_timeout_pri = 2 (0x2)
       initial_timeout_pri = 255 (0xff)
       default_intr_pri = 200 (0xc8)
       tape_intr_pri = 205 (0xcd)
       disk_intr_pri = 205 (0xcd)
       scsi_intr_pri = 205 (0xcd)
       network_intr_pri = 210 (0xd2)
       parallel_intr_pri = 215 (0xd7)
       serial_intr_pri = 215 (0xd7)
       graphics_intr_pri = 225 (0xe1)
       video_intr_pri = 230 (0xe6)
       audio_intr_pri = 235 (0xeb)

group: sthread (statically changeable)
       cl_async_pri = 90 (0x5a)
       xbox_poll_pri = 90 (0x5a)
       dpipe_pri = 90 (0x5a)
       munldd_pri = 90 (0x5a)
       io_init_pri = 90 (0x5a)
       tpisockd_pri = 90 (0x5a)
       recovery_pri = 90 (0x5a)
       cmsd_pri = 90 (0x5a)
       coalesced_pri = 90 (0x5a)
       sched_pri = 90 (0x5a)
       thand_pri = 90 (0x5a)
       async_call_pri = 90 (0x5a)
       unmountd_pri = 90 (0x5a)
       xfsd_pri = 92 (0x5c)
       bdflush_pri = 92 (0x5c)
       vfs_sync_pri = 92 (0x5c)
       xlvd_pri = 94 (0x5e)
       bpsqueue_pri = 94 (0x5e)
       exim_pri = 96 (0x60)
       shaked_pri = 98 (0x62)
       vhand_pri = 98 (0x62)
       rmonqd_pri = 98 (0x62)
       pdflush_pri = 98 (0x62)
       batchd_pri = 98 (0x62)
       reaper_pri = 100 (0x64)
       onesec_pri = 90 (0x5a)
       klogthrd_pri = 98 (0x62)
       du_conpoll_pri = 255 (0xff)
       console_pri = 255 (0xff)

group: io (statically changeable)
       hwgraph_num_dev = 32768 (0x8000)

group: shm (statically changeable)
       sshmseg = 1000 (0x3e8)
       shmmni = 1000 (0x3e8)
       shmmin = 1 (0x1) ll
       shmmax = 8589934592 (0x200000000) ll

group: vme_dma (statically changeable)
       nvme32_dma = 64 (0x40)

group: lpage_watermarks (dynamically changeable)
       coalesced_run_period = 300 (0x12c)
       percent_totalmem_16m_pages = 90 (0x5a)
       percent_totalmem_4m_pages = 0 (0x0)
       percent_totalmem_1m_pages = 0 (0x0)
       percent_totalmem_256k_pages = 2 (0x2)
       percent_totalmem_64k_pages = 1 (0x1)
       percent_totalmem_16k_pages = 0 (0x0)

group: large_pages (statically changeable)
       large_pages_enable = 1 (0x1)
       nlpages_16m = 240 (0xf0)
       nlpages_4m = 0 (0x0)
       nlpages_1m = 0 (0x0)
       nlpages_256k = 32 (0x20)
       nlpages_64k = 32 (0x20)

group: internal (statically changeable)
       dumplo = 0 (0x0)
       errbuf_cpusz = 2048 (0x800)
       putbuf_cpusz = 2048 (0x800)
       conbuf_cpusz = 2048 (0x800)
       errbuf_maxsz = 16384 (0x4000)
       putbuf_maxsz = 16384 (0x4000)
       conbuf_maxsz = 16384 (0x4000)
       histmax = 0 (0x0)

group: extacct (statically changeable)
       asarrayid = 65535 (0xffff)
       asmachid = 0 (0x0)
       incr_paggid = 1 (0x1) ll
       max_local_paggid = 65535 (0xffff) ll
       min_local_paggid = 1 (0x1) ll
       dflt_paggid = 0 (0x0) ll
       dfltprid = 0 (0x0) ll
       spilen = 64 (0x40)
       sessaf = 1 (0x1)
       do_sessacct = 0 (0x0)
       do_extpacct = 0 (0x0)
       do_procacct = 1 (0x1)

group: mload (statically changeable)
       vfssw_extra = 5 (0x5)
       fmodsw_extra = 20 (0x14)
       cdevsw_extra = 23 (0x17)
       bdevsw_extra = 21 (0x15)

group: bufcache (statically changeable)
       nbuf = 13207 (0x3397)

group: paging (dynamically changeable)
       enable_kaio_fast_path = 1 (0x1)
       enable_devzero_opt = 1 (0x1)
       cwcluster = 64 (0x40)
       zone_accum_limit = 30 (0x1e)
       shaked_interval = 1 (0x1)
       min_free_pages = 12800 (0x3200)
       min_file_pages = 3971 (0xf83)
       autoup = 10 (0xa)
       dwcluster = 64 (0x40)
       rsshogslop = 20 (0x14)
       rsshogfrac = 98 (0x62)
       tlbdrop = 100 (0x64)
       maxlkmem = 97442 (0x17ca2)
       minasmem = 50 (0x32)
       minarmem = 50 (0x32)
       vfs_syncr = 30 (0x1e)
       bdflushr = 5 (0x5)
       maxdc = 254 (0xfe)
       maxfc = 254 (0xfe)
       maxsc = 254 (0xfe)
       gpgsmsk = 1 (0x1)
       gpgshi = 3200 (0xc80)
       gpgslo = 1600 (0x640)
```

```
 group: memsize (statically changeable)
        scache_pool_size = 32 (0x20)
        maxpglst = 254 (0xfe)
        maxdmasz = 8192 (0x2000)
        syssegsz = 1200128 (0x125000)
        maxpmem = 0 (0x0)

 group: timer (statically changeable)
        timetrim = -122500 (0xfffffffffffe217c)
        itimer_on_clkcpu = 0 (0x0)
        fasthz = 1250 (0x4e2)

 group: signals (dynamically changeable)
        maxsigq = 128 (0x80)

 group: actions (statically changeable)
        nactions = 2 (0x2)

 group: streams (dynamically changeable)
        strholdtime = 50 (0x32)

 group: streams (statically changeable)
        strpmonmax = 4 (0x4)
        strctlsz = 1024 (0x400)
        strmsgsz = 32768 (0x8000)
        nstrintr = 1024 (0x400)
        nstrpush = 9 (0x9)

 group: numproc (statically changeable)
        ngroups_max = 16 (0x10)
        ncsize = 4400 (0x1130)
        ncallout = 1050 (0x41a)
        ndquot = 4400 (0x1130)
        nproc = 2100 (0x834)

 group: resource (statically changeable)
        rlimit_pthread_max = 1024 (0x400) ll
        rlimit_pthread_cur = 1024 (0x400) ll
        rlimit_nofile_max = 2500 (0x9c4) ll
        rlimit_nofile_cur = 1024 (0x400) ll
        rlimit_rss_max = 1879048192 (0x70000000) ll
        rlimit_rss_cur = 1879048192 (0x70000000) ll
        rlimit_vmem_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_vmem_cur = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_core_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_core_cur = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_stack_max = 536870912 (0x20000000) ll
        rlimit_stack_cur = 67108864 (0x4000000) ll
        rlimit_data_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_data_cur = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_fsize_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_fsize_cur = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_cpu_max = 9223372036854775807
(0x7fffffffffffffff) ll
        rlimit_cpu_cur = 9223372036854775807
(0x7fffffffffffffff) ll

 group: limits (dynamically changeable)
        idbgmaxfuncs = 1200 (0x4b0)
        maxup = 2000 (0x7d0)
        maxsymlinks = 30 (0x1e)
        nprofile = 100 (0x64)
        maxwatchpoints = 100 (0x64)
        shlbmax = 8 (0x8)
        ncargs = 29000 (0x7148)

 group: debug (dynamically changeable)
        utrace_mask = 1048383 (0xfff3f) ll
        kmem_do_poison = 1 (0x1)

 group: debug (statically changeable)
        utrace_bufsize = 0 (0x0)

 group: misc (dynamically changeable)
        fru_disable_confidence = 90 (0x5a)
        full_fru_analysis = 1 (0x1)
        bridge_rev_b_data_check_disable = 0 (0x0)
        pcibr_wg_enable_rev = 4 (0x4)
        pcibr_prefetch_enable_rev = 3 (0x3)
        r4k_corrupt_scache_data = 0 (0x0)
        gather_craylink_routerstats = 1 (0x1)
        r10k_progress_diff = 200000000 (0xbebc200)
ll
        r10k_check_count = 30000 (0x7530)
        r10k_progress_nmi = 0 (0x0)
        r10k_intervene = 0 (0x0)
        mmap_async_write = 0 (0x0)
        craylnk_bypass_disable = 0 (0x0)
        heartbeat_enable = 0 (0x0)
        dump_level = 4 (0x4)
        module_unld_delay = 5 (0x5)
        ecc_recover_enable = 60 (0x3c)
        sbe_report_cons = 0 (0x0)
        sbe_mfr_override = 0 (0x0)
        sbe_log_errors = 1 (0x1)
        panic_on_sbe = 0 (0x0)
        corepluspid = 0 (0x0)
        r4k_div_patch = 0 (0x0)

 group: misc (statically changeable)
        io4ia_war_enable = 0 (0x0)
        ignore_conveyor_override = 0 (0x0)
        io4ia_userdma_war = 1 (0x1)

 group: switch (dynamically changeable)
        tty_auto_strhold = 0 (0x0)

 group: switch (statically changeable)
        xpc_override = 0 (0x0)
        snerrste_pages_per_cube = 1 (0x1)
        xbox_sysctlr_poll_interval = 60 (0x3c)
        clean_shutdown_time = 120 (0x78)
        powerfail_cleanio = 0 (0x0)
        ignore_sysctlr_intr = 0 (0x0)
        ip32_pci_flush_prefetch = 1 (0x1)
        ip32_pci_enable_prefetch = 0 (0x0)
        enable_pci_BIST = 0 (0x0)
        warbits_override = 8 (0x8)
        spec_udma_war_on = 1 (0x1)
        disable_r10k_log = 0 (0x0)
        prod_assertion_level = 0 (0x0)
        disable_ip25_check = 0 (0x0)
        mload_auto_rtsyms = 1 (0x1)
        xpg4_sticky_dir = 1 (0x1)
        add_kthread_stack = 0 (0x0)
        ip26_allow_ucmem = 0 (0x0)
        reset_limits_on_exec = 1 (0x1)
        restrict_fastprof = 0 (0x0)
        reboot_on_panic = -1 (0xffffffffffffffff)
        use_old_serialnum = 0 (0x0)
        posix_tty_default = 0 (0x0)
        restricted_chown = 0 (0x0)
        nosuidshells = 0 (0x0)

 group: svckudp (statically changeable)
        svc_maxdupreqs = 0 (0x0)

 group: cachefs (dynamically changeable)
        replacement_timeout = 600 (0x258)
        fileheader_cache_size = 512 (0x200)
        cachefs_max_threads = 5 (0x5)
        cachefs_readahead = 1 (0x1)

 group: snfs (statically changeable)
        nfs_portmon = 0 (0x0)

 group: psema (statically changeable)
        psema_max_cnt = 8192 (0x2000)

 group: numa (statically changeable)
        numa_migr_queue_control_max_page_age = 8
(0x8)
```

```
        numa_migr_queue_time_trigger_interval = 1
(0x1)
        numa_migr_queue_time_trigger_enabled = 0
(0x0)
        numa_migr_queue_capacity_trigger_threshold =
60 (0x3c)
        numa_migr_queue_capacity_trigger_enabled = 0
(0x0)
        numa_migr_queue_traffic_trigger_threshold =
80 (0x50)
        numa_migr_queue_traffic_trigger_enabled = 0
(0x0)
        numa_migr_queue_control_mlimit = 70 (0x46)
        numa_migr_queue_control_interval = 500
(0x1f4)
        numa_migr_queue_control_enabled = 0 (0x0)
        numa_migr_queue_maxlen = 70 (0x46)
        memoryd_enabled = 0 (0x0)
        numa_migr_enqonfail_enabled = 0 (0x0)
        numa_paging_node_freemem_low_threshold = 40
(0x28)
        numa_zone_radius_max = 2 (0x2)
        numa_kernel_replication_ratio = 1 (0x1)
        numa_repl_mem_lowmark = 32 (0x20)
        numa_repl_traffic_highmark_percentage = 80
(0x50)
        numa_repl_control_enabled = 0 (0x0)
        numa_page_replication_enable = 0 (0x0)
        numa_migr_traffic_control_threshold = 80
(0x50)
        numa_migr_traffic_control_interval = 400
(0x190)
        numa_migr_traffic_control_enabled = 0 (0x0)
        numa_migr_unpegging_control_threshold = 90
(0x5a)
        numa_migr_unpegging_control_interval = 0
(0x0)
        numa_migr_unpegging_control_enabled = 1
(0x1)
        mem_tick_base_period = 1 (0x1)
        mem_tick_enabled = 1 (0x1)
        numa_migr_dampening_factor = 90 (0x5a)
        numa_migr_dampening_enabled = 0 (0x0)
        numa_migr_bounce_control_interval = 0 (0x0)
        numa_migr_melt_threshold = 90 (0x5a)
        numa_migr_melt_enabled = 0 (0x0)
        numa_migr_freeze_threshold = 20 (0x14)
        numa_migr_freeze_enabled = 0 (0x0)
        numa_migr_memory_low_threshold = 20 (0x14)
        numa_migr_memory_low_enabled = 1 (0x1)
        numa_migr_min_distance = 1 (0x1)
        numa_refcnt_overflow_threshold = 50 (0x32)
        numa_refcnt_default_mode = 0 (0x0)
        numa_migr_coaldmigr_mech = 0 (0x0)
        numa_migr_user_migr_mech = 0 (0x0)
        numa_migr_auto_migr_mech = 0 (0x0)
        numa_migr_min_maxradius = 2 (0x2)
        numa_migr_vehicle = 1 (0x1)
        numa_migr_threshold_reference = 0 (0x0)
        numa_migr_default_threshold = 50 (0x32)
        numa_migr_default_mode = 2 (0x2)

 group: nfs3 (statically changeable)
        nfs3_default_xfer = 32768 (0x8000)

 group: disp (statically changeable)
        memaff_sched = 1 (0x1)
        slice_size = 10 (0xa)

 group: xfs (statically changeable)
        xfs_panic_mask = 0 (0x0) ll
        xfs_nfs_io_units = 10 (0xa)

 group: uds (statically changeable)
        unpdg_recvspace = 16384 (0x4000)
        unpdg_sendspace = 8192 (0x2000)
        unpst_recvspace = 131072 (0x20000)
        unpst_sendspace = 131072 (0x20000)

 group: sem (statically changeable)
        semaem = 16384 (0x4000)
        semvmx = 32767 (0x7fff)
        semopm = 400 (0x190)
        semmsl = 400 (0x190)
        semmni = 2048 (0x800)

 group: lockmgr (dynamically changeable)
        normal_retry = 1 (0x1)
        first_retry = 1 (0x1)
        GraceWaitTime = 5 (0x5)
        sm_timeout = 5 (0x5)
        portmap_timeout = 5 (0x5)

 group: msg (statically changeable)
        pmqomax = 128 (0x80)
        msgseg = 8000 (0x1f40)
        msgtql = 256 (0x100)
        msgssz = 8 (0x8)
        msgmni = 512 (0x200)
        msgmnb = 65536 (0x10000)
        msgmax = 32768 (0x8000)

 group: efs (dynamically changeable)
        efs_inline = 0 (0x0)

 group: net_udp (dynamically changeable)
        udp_recvgrams = 2 (0x2)
        udp_sendspace = 61440 (0xf000)
        udp_ttl = 60 (0x3c)

 group: net_tcp (dynamically changeable)
        tcp_recvspace = 61440 (0xf000)
        tcp_sendspace = 61440 (0xf000)
        tcprexmtthresh = 3 (0x3)
        tcp_2msl = 60 (0x3c)
        tcp_mtudisc = 1 (0x1)
        tcp_maxpersistidle = 7200 (0x1c20)
        tcp_keepintvl = 75 (0x4b)
        tcp_keepidle = 7200 (0x1c20)
        tcp_ttl = 60 (0x3c)

 group: net_rsvp (statically changeable)
        ps_num_batch_pkts = 0 (0x0)
        ps_rsvp_bandwidth = 50 (0x32)
        ps_enabled = 1 (0x1)

 group: net_mbuf (statically changeable)
        mbmaxpages = 32768 (0x8000)

 group: net_ip (dynamically changeable)
        subnetsarelocal = 1 (0x1)
        allow_brdaddr_srcaddr = 0 (0x0)
        ipdirected_broadcast = 0 (0x0)
        ipsendredirects = 1 (0x1)
        ipforwarding = 1 (0x1)
        ipfilterd_inactive_behavior = 1 (0x1)
        icmp_dropredirects = 0 (0x0)

 group: network (statically changeable)
        netthread_float = 0 (0x0)
        netthread_pri = 98 (0x62)

 group: inpcb (statically changeable)
        udp_hashtablesz = 2048 (0x800)
        tcp_hashtablesz = 16384 (0x4000)
```

RDBMS Configuration Values

```
##################################################
############################
#
#               Configuration File for the Sybase
SQL Server
#
#               Please read the System
Administration Guide (SAG)
```

```
#              before changing any of the values in
this file.
#
##################################################
##########################



[Configuration Options]

[General Information]

[Backup/Recovery]
       recovery interval in minutes = 2000
       print recovery information = DEFAULT
       tape retention in days = DEFAULT

[Cache Manager]
       number of oam trips = DEFAULT
       number of index trips = DEFAULT
       procedure cache percent = 1
       memory alignment boundary = DEFAULT
       global async prefetch limit = 0

[Named Cache:c_customer]
       cache size = 5M
       cache status = mixed cache
       cache status = HK ignore cache
       cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
       pool size = 5M
       wash size = 1024 K
       local async prefetch limit = 0

[Named Cache:c_customer_index]
       cache size = 104M
       cache status = mixed cache
       cache status = HK ignore cache
       cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
       pool size = 104M
       wash size = 512 K
       local async prefetch limit = 0

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

[2K I/O Buffer Pool]
       pool size = 1M
       wash size = 512 K
       local async prefetch limit = 0

[4K I/O Buffer Pool]
       pool size = 5M
       wash size = 2048 K
       local async prefetch limit = 0

[Named Cache:c_no_ol]
       cache size = 100M
       cache status = mixed cache
       cache status = HK ignore cache
       cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
       pool size = 100M
       wash size = 512 K
       local async prefetch limit = 0

[Named Cache:c_ol_index]
       cache size = 30M
       cache status = mixed cache
       cache status = HK ignore cache
       cache replacement policy = relaxed LRU
replacement

[2K I/O Buffer Pool]
       pool size = 30M
       wash size = 512 K
       local async prefetch limit = 0

[Named Cache:c_orders]
       cache size = 110M
       cache status = mixed cache
       cache status = HK ignore cache
       cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
       pool size = 100M
       wash size = 4096 K
       local async prefetch limit = 0

[16K I/O Buffer Pool]
       pool size = 10M
       wash size = 1024 K
       local async prefetch limit = 0

[Named Cache:c_stock]
       cache size = 1099M
       cache status = mixed cache
       cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
       pool size = 1099M
       wash size = 4096 K
       local async prefetch limit = 0

[Named Cache:c_stock_index]
       cache size = 155M
       cache status = mixed cache
       cache status = HK ignore cache
       cache replacement policy = relaxed LRU
replacement

[2K I/O Buffer Pool]
       pool size = 155M
       wash size = 2048 K
       local async prefetch limit = 0

[Named Cache:c_tinyhot]
       cache size = 21M
       cache status = mixed cache
       cache status = HK ignore cache
       cache replacement policy = relaxed LRU
replacement

[2K I/O Buffer Pool]
       pool size = 21M
       wash size = 512 K
       local async prefetch limit = 0

[Named Cache:default data cache]
       cache size = 1M
       cache status = default data cache
       cache status = HK ignore cache
       cache replacement policy = relaxed LRU
replacement

[2K I/O Buffer Pool]
       pool size = 1M
       wash size = 256 K
       local async prefetch limit = 0

[Meta-Data Caches]
       number of open databases = DEFAULT
       number of open objects = DEFAULT
       open object spinlock ratio = DEFAULT
       number of open indexes = DEFAULT
       open index hash spinlock ratio = DEFAULT
       open index spinlock ratio = DEFAULT

[Disk I/O]
       allow sql server async i/o = DEFAULT
       disk i/o structures = 4096
       page utilization percent = DEFAULT
       number of devices = 100
       disable character set conversions = DEFAULT
```

```
        enable unicode conversions = DEFAULT
        size of unilib cache = DEFAULT

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

[O/S Resources]
        max async i/os per engine = 512
        max async i/os per server = 512

[Parallel Query]
        number of worker processes = DEFAULT
        memory per worker process = DEFAULT
        max parallel degree = DEFAULT
        max scan parallel degree = DEFAULT

[Physical Resources]

[Physical Memory]
        total memory = 902000
        additional network memory = 1998848
        lock shared memory = 1
        shared memory starting address = 301989888
        max SQL text monitored = DEFAULT

[Processors]
        max online engines = 2
        min online engines = DEFAULT

[SQL Server Administration]
        default database size = DEFAULT
        identity burning set factor = DEFAULT
        allow nested triggers = DEFAULT
        allow updates to system tables = 1
        print deadlock information = DEFAULT
        default fill factor percent = DEFAULT
        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 = 100000
        runnable process search count = DEFAULT
        i/o polling process count = 10
        time slice = DEFAULT
        deadlock retries = DEFAULT
        cpu grace time = DEFAULT
        number of sort buffers = DEFAULT
        number of large i/o buffers = DEFAULT
        size of auto identity column = DEFAULT
        identity grab size = DEFAULT
        lock promotion HWM = DEFAULT
        lock promotion LWM = DEFAULT
        lock promotion PCT = DEFAULT
        housekeeper free write percent = 0
        partition groups = DEFAULT
        partition spinlock ratio = DEFAULT
        allow resource limits = DEFAULT
        number of aux scan descriptors = DEFAULT
        SQL Perfmon Integration = DEFAULT
        allow backward scans = DEFAULT

[User Environment]
        number of user connections = 110
        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 = 10000
        deadlock checking period = 1000
        freelock transfer block size = DEFAULT
        max engine freelocks = 40
        address lock spinlock ratio = 5
        page lock spinlock ratio = 5
        table lock spinlock ratio = 1

[Security Related]
        systemwide password expiration = DEFAULT
        audit queue size = DEFAULT
        curread change w/ open cursors = DEFAULT
        allow procedure grouping = DEFAULT
        select on syscomments.text = DEFAULT
        auditing = DEFAULT
        current audit table = DEFAULT
        suspend audit when device full = DEFAULT
        max roles enabled per user = DEFAULT
        unified login required = DEFAULT
        use security services = DEFAULT
        msg confidentiality reqd = DEFAULT
        msg integrity reqd = DEFAULT
        msg replay detection reqd = DEFAULT
        msg origin checks reqd = DEFAULT
        msg out-of-seq checks reqd = DEFAULT
        secure default login = DEFAULT
        dump on conditions = DEFAULT

[Extended Stored Procedure]
        esp unload dll = DEFAULT
        esp execution priority = DEFAULT
        esp execution stacksize = DEFAULT
        xp_cmdshell context = DEFAULT
        start mail session = DEFAULT

[Error Log]
        event logging = DEFAULT
        log audit logon success = DEFAULT
        log audit logon failure = DEFAULT
        event log computer name = DEFAULT

[Rep Agent Thread Administration]
        enable rep agent threads = DEFAULT
        maximum dump conditions = DEFAULT

[Component Integration Services]
        enable cis = DEFAULT
        cis connect timeout = DEFAULT
        cis bulk insert batch size = DEFAULT
        max cis remote connections = DEFAULT
        max cis remote servers = DEFAULT
        cis packet size = DEFAULT
        cis cursor rows = DEFAULT
        cis rpc handling = DEFAULT
```

Tuxedo Configuration Values

```
*RESOURCES
IPCKEY 40001
MASTER SITE1
UID 4230
GID 241
PERM 0666
MAXACCESSERS 3700
MAXGTT 100
MAXSERVERS 100
MAXSERVICES 100
MAXCONV 1
MODEL SHM
LDBAL Y
CMTRET COMPLETE
MAXBUFTYPE 16
MAXBUFSTYPE 32
SCANUNIT 30
SANITYSCAN 5
```

```
DBBLWAIT 1
BBLQUERY 60
BLOCKTIME 10
NOTIFY DIPIN
SYSTEM_ACCESS FASTPATH
USIGNAL SIGUSR2


*MACHINES
dbclient4       LMID=SITE1
        TUXCONFIG="/usr/people/sybase/tpcc-
kit/workdir/tuxedo/tuxconfig"
        ROOTDIR="/usr/people/sybase/tpcc-
kit/workdir/tuxedo/TUXEDO"
        APPDIR="/usr/people/sybase/tpcc-
kit/workdir/bin"
        ULOGPFX="/usr/people/sybase/tpcc-
kit/workdir/tuxedo/log/ULOG"

*GROUPS
group1  LMID=SITE1      # logical machine
identifier
        GRPNO=1         # server group number
group2  LMID=SITE1      # logical machine
identifier
        GRPNO=2         # server group number
group3  LMID=SITE1      # logical machine
identifier
        GRPNO=3         # server group number
group4  LMID=SITE1      # logical machine
identifier
        GRPNO=4         # server group number
group5  LMID=SITE1      # logical machine
identifier
        GRPNO=5         # server group number


#
#--------------------------------------------#
*SERVERS                                     #
#--------------------------------------------#
#
# new_order servers
newo SRVGRP=group1 CLOPT="-A" SRVID=101
     REPLYQ=Y RQADDR=neworder MIN=18 RESTART=N
# payment servers
paym SRVGRP=group2 CLOPT="-A" SRVID=201
     REPLYQ=Y RQADDR=payment MIN=10 RESTART=N
# order_status servers
ords SRVGRP=group3 CLOPT="-A" SRVID=301
     REPLYQ=Y RQADDR=orderstatus MIN=4 RESTART=N
# delivery servers
dely SRVGRP=group4 CLOPT="-A -- 1" SRVID=401
     REPLYQ=Y RQADDR=delivery RESTART=N
dely SRVGRP=group4 CLOPT="-A -- 2" SRVID=402
     REPLYQ=Y RQADDR=delivery RESTART=N
dely SRVGRP=group4 CLOPT="-A -- 3" SRVID=403
     REPLYQ=Y RQADDR=delivery RESTART=N
dely SRVGRP=group4 CLOPT="-A -- 4" SRVID=404
     REPLYQ=Y RQADDR=delivery RESTART=N
dely SRVGRP=group4 CLOPT="-A -- 5" SRVID=405
     REPLYQ=Y RQADDR=delivery RESTART=N
#dely SRVGRP=group4 CLOPT="-A -- 6" SRVID=406
#     REPLYQ=Y RQADDR=delivery RESTART=N
#dely SRVGRP=group4 CLOPT="-A -- 7" SRVID=407
#     REPLYQ=Y RQADDR=delivery RESTART=N
#dely SRVGRP=group4 CLOPT="-A -- 8" SRVID=408
#     REPLYQ=Y RQADDR=delivery RESTART=N
#dely SRVGRP=group4 CLOPT="-A -- 9" SRVID=409
#     REPLYQ=Y RQADDR=delivery RESTART=N
#dely SRVGRP=group4 CLOPT="-A -- 10" SRVID=410
#     REPLYQ=Y RQADDR=delivery RESTART=N
# stocklevel servers
stockl SRVGRP=group5 CLOPT="-A" SRVID=501
     REPLYQ=Y RQADDR=stocklevel MIN=8 RESTART=N
#
#--------------------------------------------#
*SERVICES                                    #
#--------------------------------------------#
#
"DEL"
             LOAD=50 PRIO=50
BUFTYPE="ALL"
             AUTOTRAN=N
             TRANTIME=45

"NEWO"
             LOAD=50 PRIO=50
BUFTYPE="ALL"
             AUTOTRAN=N
             TRANTIME=45

"ORDS"
             LOAD=50 PRIO=50
BUFTYPE="ALL"
             AUTOTRAN=N
             TRANTIME=45

"PAYM"
             LOAD=50 PRIO=50
BUFTYPE="ALL"
             AUTOTRAN=N
             TRANTIME=45

"STOCK"
             LOAD=50 PRIO=50
BUFTYPE="ALL"
             AUTOTRAN=N
             TRANTIME=45


#
#--------------------------------------------#
*ROUTING                                     #
#--------------------------------------------#
#
```

tpcc_init.sh

```
#!/bin/sh -f

isql -Usa -P << EOF
dbcc tune(deviochar, -1, "40")
go
dbcc tune(maxwritedes, 30)
go
dbcc tune ("doneinproc", 0)
go
dbcc tune("cleanup", 0)
go
dbcc traceoff(-1)
go
EOF

# Set io sizes for tpcc tables

isql -Usa -P << EOF
use tpcc
go
dbcc iosize(tpcc, warehouse, 16)
go
dbcc iosize(tpcc, district, 16)
go
dbcc iosize(tpcc, history, 16)
go
dbcc iosize(tpcc, item, 16)
go
dbcc iosize(tpcc, orders, 16)
go

exec sp_logiosize "4"
go

EOF
```

tpcc_cache_bind.sh

```
#!/bin/sh -f

isql -Usa -P$PASSWORD << EOF

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

use tpcc
go
checkpoint
go

/*
** Cache c_log
*/

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

/*
** Cache c_system
*/

/* sp_bindcache "c_system", "tpcc", "sysindexes"
 * go
 * sp_bindcache "c_system", "tpcc", "sysindexes",
"sysindexes"
 * go
 */

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

use tpcc
go
checkpoint
go

/*
** Cache c_tinyhot
*/

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

/*
** Cache c_no_ol
*/

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

/*
** Cache c_ol_index
*/

sp_bindcache "c_ol_index", "tpcc", "order_line",
"ol_clu"
go

/*
** Cache c_orders
*/

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

/*
** Cache c_stock_index
*/

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

/*
** Cache c_stock
*/

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

/*
** Cache c_customer
*/

sp_bindcache "c_customer", "tpcc", "customer"
go

/*
** Cache c_customer_index
*/

sp_bindcache "c_customer_index", "tpcc",
"customer", "c_clu"
go
sp_bindcache "c_customer_index", "tpcc",
"customer", "c_non1"
go

/*
** Default cache
*/

sp_bindcache "default data cache", "tpcc",
"history"
go
EOF
```

Appendix D: Disk Storage

This Appendix contains the calculations used to determine the storage requirements for the 8 hour logical log and the 180-day space requirements.

	505			**Warehouses**	
TpmC	6040.33				

Table	**Rows**	**Data 1K pages**	**Index 1K pages**	**Extra 5%**	**Total with 5%**
Warehouse	505	1,010	8	51	1,069
District	5,050	10,100	46	507	10,653
Item	100,000	9,524	86	481	10,091
Customer	15,150,000	10,100,000	844,886	547,244	11,492,130
New_Order	4,545,000	49,674	598	2,514	52,786
Stock	50,500,000	16,833,334	187,036	851,019	17,871,389
History (Dynamic)	15,150,000	848,464	0		848,464
Orders (Dynamic)	15,150,000	409,460	4,934		414,394
Order_line (Dynamic)	151,500,000	9,181,820	120,814		9,302,634
Total MB	252,100,555	37,443,386	1,158,408	1,401,815	40,003,609

Loaded	38,601,794
With 5%	40,003,609
8 Hours	

Segment Name	**# of Segs**	**Total Allocated**	**Tables**			
Master	1	46				
				Loaded + 5%+8hrs	**Unused**	**8 Hour Growth**
Cust_idx	4	1,040	c			
Customer	20	10,785	c	11,223	602	
History	1	1,125	h	985	140	156.75
Order Line	6	11,210	ol	10,920	290	1,834.91
Order	4	827	o	531	296	125.97
Stock	44	17,770	s	17,453	317	
w_d_no_l	1	130	w,d,l,no	73	57	
tmpdb	1	50				
Total Allocated		42,983		41,184	1,703	2,118

	MB (2^20 byte)	
Dynamic Space	10,195	Sum of Data for Order, Order_Line and History
Static Space	28,871	Sum of all data, index + 5% - above dynamic space
Free Space	3,916	Total space allocated to DBMS - dynamic and static
Daily growth	1,951.10	(Dynamic space / (W * 62.5)) * tpmC
Daily spread	990	Free space - 1.5 * Daily growth (zero if negative)
	990	Max of Daily Spread and Zero.
180 day space (MB)	558,238	Static space + 180 * (daily growth + daily spread)
180 day space (GB)	**545.15**	(GB=2^30=MB/1024)
8 hr log space (GB)	**9.95**	(excludes RAID-1)

	Space Needed	**Disk size (GB)**	**Disks Needed**	**GB Priced**
180 day space	545.15	8.48	65	551.18
Logical Logs (w/mirrors)	9.95	8.48	4	33.92
OS, file sys, Swap	2.00	8.48	1	8.48
Total	557.11		70	**593.58**

Appendix E: RTE Configuration

This Appendix contains the RTE input parameters and code fragments used to generate each transaction input field.

RTE Parameters:

```
=============================================
                    RTE_TPCC.H
=============================================
#define WAREHOUSES 505

#define NEWORD          1
#define PAYMENT         2
#define ORDSTAT         3
#define DELIVERY        4
#define STOCKLEV        5

#define def_C_C_ID          319
#define def_C_C_LAST        240
#define def_C_OL_I_ID       3849

unsigned int C_C_ID;
unsigned int C_C_LAST;
unsigned int C_OL_I_ID;

#define A_C_ID          1023
#define A_C_LAST        255
#define A_OL_I_ID       8191

#define X_C_ID          A_C_ID
#define X_C_LAST        A_C_LAST
#define X_OL_I_ID       A_OL_I_ID

#define DISTRICTS_PER_WAREHOUSE      10
       /* value = 10 */
#define CUSTOMERS_PER_DISTRICT       3000    /*
value = 3,000 */
#define ITEMS                       100000    /*
value = 100,000 */
#define RandVal         lrand48
#define RandSeed        srand48

#define STATE_BEGIN    0
#define STATE_RAMPUP   1
#define STATE_HOT      2
#define STATE_STOP     6
#ifdef ACID
FILE *acidfp;
#endif

/* New Order structure */
struct  rte_neworder {
       short    s_W_ID;
       short    s_D_ID;
       long     s_C_ID;
       short    s_O_OL_CNT;
       short    s_all_local;
       short    s_temp_cnt;
       struct rte_items {
              short    s_OL_SUPPLY_W_ID;
              long     s_OL_I_ID;
              short    s_OL_QUANTITY;
       } items[15];
};

/* Paymentstructure */
struct rte_payment {
       short    s_W_ID;
       short    s_D_ID;
       short    s_C_ID;
       int     s_C_W_ID;
       int     s_C_D_ID;
       short    s_all_local;
       int     s_H_AMOUNT;
       char     s_C_LAST[17];
       };

/* Order Status structure */
struct rte_ordsta {
        short    s_W_ID;
        short    s_D_ID;
       short    s_C_ID;
        char     s_C_LAST[17];
       };

/* Stock Level Structure */
struct rte_stocklev {
       short    s_W_ID;
       short    s_D_ID;
       short    s_C_ID;
       int s_threshold;
       };

/* Delivery Structure */
struct  rte_delivery {
       short    s_W_ID;
       int s_O_CARRIER_ID;
       };
```

Main Loop Of The Rte

```
/** Each signal to user transitions state in
sequence:
    ** STATE_RAMPUP -> STATE_HOT -> STATE_STOP
    ** Last signal sent to user should transition
to STATE_STOP
    **/
   Signal(set_state);

   /** Initial State is STATE_RAMPUP w/i Full
Logging **/
   while ( tpcc_state !=  STATE_STOP) {  /* do
until state is stop */

      trans_type= Range(1,100000);
      if (trans_type < 4060) option=STOCKLEV;
/* stocklev 4.06% */
      else if (trans_type < 8120)
option=DELIVERY;      /* delivery 4.06% */
      else if (trans_type < 12180)
option=ORDSTAT;      /* ordstat  4.06% */
      else if (trans_type < 55280)
option=PAYMENT;      /* payment 43.1% */

      else option=NEWORD;
/* neword 40+% */


      switch (option) {
          case NEWORD:
             Thinktne(0,12.1,120);
             rte_neword(warehouse_id);
             break;
          case PAYMENT:
             Thinktne(0,12,120);
             rte_payment(warehouse_id);
             break;
          case ORDSTAT:
             Thinktne(0,10,100);
             rte_ordstat(warehouse_id);
             break;
          case DELIVERY:
             Thinktne(0,5,50);
             rte_delivery(warehouse_id);
             break;
          case STOCKLEV:
             Thinktne(0,5,50);
             rte_stocklev(warehouse_id,
stkl_d_id);
```

```
            break;
          default:
            printf("*** ERROR ***, no such
option exists. \n");
            break;
        }

        /** After 25 Txns. set_state=STATE_HOT:
Partial Logging **/
        if ( trans_count == 25 ) {
                set_state();
                trans_count++;
        }
        else trans_count++;
}
```

Code Fragments for generating Transaction inputs:

```
================================================
                        RTE_TRANS.C
================================================

#include "rte_tpcc.h"

int     first=0;
char    buf[1100], OL_S[200], logO[1100];
char    Whoabuf[200];
extern char *last_name_parts[10];

/*************************************************
********
*       Generate a new order transaction
        *
*************************************************
*******/
rte_neword(warehouse_id)
int warehouse_id;
{
struct rte_neworder  neworder;
int     offset, i, length, remote;
long    hours, min;
float   sec;
struct timevalue start, end, KeyTime;

  /* Get New Order Screen */
    sprintf(&buf,"%d", NEWORD );
    Mxmit(&buf,  "");           /* <== send the
buffer  */

 /* Generate transaction */
   remote= 0;
   neworder.s_W_ID = warehouse_id;
   neworder.s_D_ID =
Range(1,DISTRICTS_PER_WAREHOUSE);
   neworder.s_C_ID = NUrand_val(A_C_ID, 1,
CUSTOMERS_PER_DISTRICT, C_C_ID);
   neworder.s_O_OL_CNT = Range(5,15);
   neworder.s_temp_cnt = neworder.s_O_OL_CNT;
   neworder.s_all_local = 1;     /* Let us say
all orders are local */

   /* generate all order lines */
   for (i=0; i < neworder.s_O_OL_CNT; i++) {
       if ( Range(1,10000) > 95 )
           neworder.items[i].s_OL_SUPPLY_W_ID =
neworder.s_W_ID;
       else {
           remote++;
           neworder.s_all_local = 0;  /* Not all
orders are local */
          while
((neworder.items[i].s_OL_SUPPLY_W_ID =
Range(1,WAREHOUSES)) ==
           neworder.s_W_ID);
       }
       neworder.items[i].s_OL_I_ID=
NUrand_val(A_OL_I_ID, 1, ITEMS, C_OL_I_ID);
       neworder.items[i].s_OL_QUANTITY =
Range(1,10);
    }
    sprintf(OL_S," OrderLines
%d:%d",neworder.s_O_OL_CNT,remote);
    Log(&OL_S);
    /* is it a rollback transaction ? */
    i = Range(1,10000);
    if (i < 102) {            /* Yes! */
        neworder.items[neworder.s_O_OL_CNT -
1].s_OL_I_ID = 999999;
        if (neworder.s_all_local)
            Nametransaction("NewOrderRollback");
/* <== transaction type */
        else
            Nametransaction("NewOrderRollRem");
/* <== transaction type */
     }
     else {
     if (neworder.s_all_local)
          Nametransaction("NewOrder");   /* <==
transaction type */
     else
          Nametransaction("NewOrderRemote");   /*
<== transaction type */
     }

    for (i=0; i < neworder.s_O_OL_CNT; i++){

offset+=sprintf(&buf[offset],"^I%d^I%d^I%d",
neworder.items[i].s_OL_SUPPLY_W_ID,
          neworder.items[i].s_OL_I_ID,
neworder.items[i].s_OL_QUANTITY );
      /* offset= offset + length;  Next offset
into empower buffer. */
     }


    /***** Do the neworder transaction *****/
    Mxmit(&buf, "^M", "");       /* <== send the
buffer  */
}

/*-----------------------------------------------
-------*/
/*      Generate a payment transaction
                */
/*-----------------------------------------------
-------*/
rte_payment( warehouse)
int warehouse;
{
static int     i, j, type;
static int     i1, i2, i3;
struct rte_payment payment;
struct timevalue start, end, KeyTime;
long hours, min;
float sec;


if (first == 0 ) {
       init_name_part();
       first++;
}
  /* Get Payment Screen */
    Nametransaction("PayMenue");   /* <==
transaction type */
    sprintf(&buf,"%01d", PAYMENT );
    Mxmit(&buf,  "");           /* <== send the
buffer  */

  /* Generate Payment Transaction */
   payment.s_W_ID = warehouse;
   payment.s_D_ID =
Range(1,DISTRICTS_PER_WAREHOUSE);
   payment.s_H_AMOUNT = Range(100,500000)/100.0;
/* 5000.00 */
   if ( Range(1,1000) <= 850 ) {
       type= 0;
       payment.s_C_W_ID = payment.s_W_ID;
       payment.s_C_D_ID = payment.s_D_ID;
    }
    else {
```

```
       type= 1;
       payment.s_C_D_ID = Range(1,
DISTRICTS_PER_WAREHOUSE);
       while ((payment.s_C_W_ID =
Range(1,WAREHOUSES)) == payment.s_W_ID);
     }

   if (Range(1,1000) < 400) {
      payment.s_C_ID =
NUrand_val(A_C_ID,1,CUSTOMERS_PER_DISTRICT,
C_C_ID);
      payment.s_C_LAST[0] = (char)NULL;
      sprintf(buf,"%d^I%d^I%d^I%d^I^I%d",
payment.s_D_ID,
       payment.s_C_ID, payment.s_C_W_ID,
payment.s_C_D_ID, payment.s_H_AMOUNT);
   }
   else
   {
      int iall;
      if ( type == 0 )
          type= 2;
      else type= 3;
      payment.s_C_ID = 0;
      iall =  NUrand_val(A_C_LAST,1, X_C_LAST,
C_C_LAST);
      i3 = iall%10;
      i2 = (iall/10)%10;
      i1 = (iall/100)%10;
      strcpy(payment.s_C_LAST,
last_name_parts[i1]);
      strcat(payment.s_C_LAST,
last_name_parts[i2]);
      strcat(payment.s_C_LAST,
last_name_parts[i3]);
      payment.s_all_local = 1;
      sprintf(buf,"%d^I^I%d^I%d^I%s^I%d",
        payment.s_D_ID,  payment.s_C_W_ID,
        payment.s_C_D_ID, payment.s_C_LAST,
payment.s_H_AMOUNT);

    sprintf(logO," PAYMENT  %d:%s:%d:%s:%d:%s",
i1, last_name_parts[i1],
       i2, last_name_parts[i2], i3,
last_name_parts[i3]);
    Log(&logO);
    sprintf(logO," PAYMENT  s_D_ID= %d s_C_ID= %d
\
        s_C_W_ID= %04d s_C_D_ID= %02d s_H_AMOUNT=
%d s_C_LAST= %s",
        payment.s_D_ID, payment.s_C_ID,
payment.s_C_W_ID,
       payment.s_C_D_ID, payment.s_H_AMOUNT ,
payment.s_C_LAST);
    Log(&logO);

    }
    if ( type == 0 )
Nametransaction("PaymentHoCid");
    if ( type == 1 )
Nametransaction("PaymentReCid");
    if ( type == 2 )
Nametransaction("PaymentHoLast");
    if ( type == 3 )
Nametransaction("PaymentReLast");

    /***** Do the Payment transaction *****/
    Mxmit(&buf, "^M", "");  /*  send message to
the SUT */
}

/*-------------------------------------------------
-------*/
/*      Generate order status transaction
       */
/*-------------------------------------------------
-------*/
rte_ordstat(warehouse)
int warehouse;
{
static int i, j, iall, i1, i2, i3;
struct rte_ordsta ordsta;
struct timevalue start, end, KeyTime;
long hours, min;
float sec;

if (first == 0 ) {
       init_name_part();
       first++;
}

  /* Get Order Status Screen */
    sprintf(&buf,"%01d", ORDSTAT );
    Mxmit(&buf, "");           /* <== send the
buffer  */

  /* Generate IOrder Status transaction */
   ordsta.s_W_ID = warehouse;
   ordsta.s_D_ID =
Range(1,DISTRICTS_PER_WAREHOUSE);

   if (Range(1,1000) < 400) {
       Nametransaction("OrderStatus ");      /*
transaction name */
        ordsta.s_C_ID =
NUrand_val(A_C_ID,1,CUSTOMERS_PER_DISTRICT,
C_C_ID);
        ordsta.s_C_LAST[0] = (char)NULL;
   }
   else
   {
        Nametransaction("OrderStatusLast");
/* transaction name */
        ordsta.s_C_ID = 0;
       iall = NUrand_val(A_C_LAST,1, X_C_LAST,
C_C_LAST);
       i3 = iall%10;
       i2 = (iall/10)%10;
       i1 = (iall/100)%10;
        strcpy(ordsta.s_C_LAST,
last_name_parts[i1]);
        strcat(ordsta.s_C_LAST,
last_name_parts[i2]);
        strcat(ordsta.s_C_LAST,
last_name_parts[i3]);
    }
    if (ordsta.s_C_ID > 0 )
      sprintf(buf,"%d^I%d", ordsta.s_D_ID,
ordsta.s_C_ID);
    else
      sprintf(buf,"%d^I^I%s",
      ordsta.s_D_ID, ordsta.s_C_LAST);

       /***** Do the Order Status transaction
*****/
       Mxmit(&buf, "^M", "");             /* send
to the SUT */
}

/*-------------------------------------------------
-------*/
/*      Generate stock level transaction
       */
/*-------------------------------------------------
-------*/
rte_stocklev(warehouse, district)
int warehouse, district;
{
struct rte_stocklev stocklev;
struct timevalue start, end, KeyTime;
long hours, min;
float sec;

Beginfunction("Stock Level transactions");

  /* Get Stock Level Screen */
    sprintf(&buf,"%01d", STOCKLEV );
    Mxmit(&buf,  "");           /* <== send the
buffer  */
```

```
 /* Generate the Stock Level transaction */
   stocklev.s_threshold = Range(10,20);
   sprintf(buf,"%d", stocklev.s_threshold);

   /***** Do the Stock Level transaction *****/
   Nametransaction("StockLevel ");    /*
transaction name */
   Mxmit(&buf, "^M", "");             /* send to
SUT */
}

/*---------------------------------------------
-------*/
/*             Generate delivery transaction
       */
/*---------------------------------------------
-------*/
rte_delivery(warehouse)
int warehouse;
{
struct rte_delivery delivery;
struct timevalue start, end, KeyTime;
long hours, min;
float sec;

Beginfunction("Delivery transactions");

  /* Get Delivery Screen */
    sprintf(&buf,"%d", DELIVERY );
    Mxmit(&buf,  "");           /* <== send the
buffer  */

   /* The SUT already has this information */
   delivery.s_O_CARRIER_ID = Range(1,10);
   sprintf(buf,"%d", delivery.s_O_CARRIER_ID);

   /***** Do the Delivery transaction *****/
   Mxmit(&buf, "^M", "");               /* send
to SUT */
}
```

```
=============================================
                        RTE_MISC.C
=============================================
#include "rte_tpcc.h"

char *last_name_parts[1000];

/*---------------------------------------------
---------------*/
/*   NUrand_val
*/
/*---------------------------------------------
---------------*/
/* static int NUrand_val ( A, x, y, C )  */
int NUrand_val ( A, x, y, C )
int A, x, y, C;
{
return((((Range(0,A)|Range(x,y))+C)%(y-x+1))+x);
}


/*---------------------------------------------
---------------*/
/*   init_name_part
*/
/*---------------------------------------------
---------------*/
init_name_part()
{
int     i;

        for (i = 0; i < 100; i++ ) {
                last_name_parts[i*10]= "BAR";
                last_name_parts[i*10+1]= "OUGHT";
                last_name_parts[i*10+2]= "ABLE";
                last_name_parts[i*10+3]=  "PRI";
                last_name_parts[i*10+4]=  "PRES";
                last_name_parts[i*10+5]=  "ESE";
                last_name_parts[i*10+6]=  "ANTI";
                last_name_parts[i*10+7]=  "CALLY";
                last_name_parts[i*10+8]=  "ATION";
                last_name_parts[i*10+9]=  "EING";
        }
}
```

Appendix F: Price Quotes

NTERVISION
STEMS TECHNOLOGIES, INC.
UOTATION

Quotation Number MM042098-21 **Page** 1 **of** 3
Quotation Date April 22, 1998
(Quote is valid for 30 days)

O:
Access Graphics
1426 Pearl street
Boulder, CO 80302

FROM:
InterVision Systems Technologies, Inc.
3218 Scott Blvd.
Santa Clara, CA 95054
Tel: 408-980-8550
Fax: 408-980-8893
Web: www.intervision.com

'e are pleased to quote as follows:
Submitted By: Mark McEnroe Direct: 408-567-4212

eference: | Terms: Net 30 days Credit Terms on approval of InterVision Credit Dept. | FOB: InterVision Santa Clara, CA

tem	Model	Description	Qty.	Unit Net Price	Extended Net Price
1	D1-H6502-4	Origin2000 DS: 2X 250Mhz R10Kwith 4MB cache on 1 Node brd. 9.1gb sys disk CDROM, includes Irix 6.5 and Database Accel	1	$ 46,427	$ $46,427
2	FCEW	Full Care Ext. Warranty, D1-H6502-4	1	17,304	17,304
3	OPN-512-D1	First 512 Mb Memory in one bank	1	6,050	$6,050
4	FCEW	Full Care Ext. Warranty, OPN-512-D1	1	638	$638
5	H4-N512-D1	Additional 512 Mb in One Bank	3	6,050	$18,150
6	FCEW	Full Care Ext. Warranty, H4-N512-D1	3	1920	$1,920
7	FTO-9GB	9.1GB SE USCSI Disk for Orgin2000 and Onyx2 Sys.	1	N/C	N/C
8	XT-FC-2P	2-port Fibre Channel XIO card for Origin2000	2	3,150	$6,300
9	FCEW	Full Care Ext Warranty, XT-FC-2P	2	728	$728
10	SC4-IRIX-6.5	IRIX 6.5 Operating System Software. bundled with hardware	1	N/C	N/C
11	DK-LD1-001	Deskside destination kit for Origin2000 and Onyx2 systems, US/Canada (110VAC)	1	N/C	N/C
12	DK-SL1-001	Destination Kit for Single Tower Origin200, Origin Vault, US/Canada (110VAC)	1	N/C	N/C

Estimated Shipment: _____ **Days After Receipt of Order, Subject to Change.**

PLEASE READ THIS QUOTATION CAREFULLY. THE TERMS AND CONDITIONS SET FORTH ON THE BACK OF THIS FORM CONSTITUTE THE ENTIRE AGREEMENT BETWEEN SELLER AND BUYER. SELLER WILL NOT BE BOUND BY ANY TERMS OF BUYER'S ORDER THAT ARE INCONSISTENT WITH THE TERMS HEREIN.

THIS QUOTATION REFLECTS INTERVISION'S PRICES AND ESTIMATED DELIVERY DATES TO BUYER FOR THE NEXT THIRTY (30) DAYS, UNLESS CONTRARY NOTICE IS GIVEN BY INTERVISION DURING SUCH THIRTY (30) DAY PERIOD. SHOULD BUYER PLACE AN ORDER FOR THE PRODUCTS SPECIFIED ABOVE, THE SALE SHALL BE SUBJECT SOLELY TO THE TERMS AND CONDITIONS CONTAINED ON THE FACE AND REVERSE SIDE HEREOF. TO PLACE AN ORDER, PLEASE SIGN THIS QUOTATION WHERE INDICATED AND RETURN A COPY TO INTERVISION

Accepted by Customer:

Print or Type Name _____ Title _____

Signature _____ Date _____

Accepted by InterVision:

Print or Type Name _____ Title _____

Signature _____ Date _____

952 P02 APR 22 '98 09:36 INTERVISION 408-980-8893

NTERVISION
STEMS TECHNOLOGIES, INC.
UOTATION

Quotation Number MM042098-21 Page 2 of 3
Quotation Date April 22, 1998
(Quote is valid for 30 days)

em	Model	Description	Qty.	Unit Net Price	Extended Net Price
3	MO5-CD	CD-ROM Update Media requirement For Support only	1	$ 1,042	$ $1,042
4	P-F-RACK	Origin FibreRack (empty) for rackmount Fibre Vault, FC RAID and FC RAID Vault	1	5,900	5,900
5	FCEW	Full Care Ext. Warranty, P-F-RACK	1	634	$634
6	P-F-B10X9-R	Origin rackmount Fibre Vault with ten 9.1GB FC disks, one FC loop controller (LCC) one pwr suply	8	22,070	$176,560
7	FCEW	Full Ext. Warranty, P-F-B10X9-R	8	18,684	$18,684
8	X-F-COP-10M	10-meter copper fibre Channel cable	4	100	$400
9	X-F-COP-).3M	0.3-meter copper fibre Channel cable	4	50	$200
20	9360005-10FT	Twisted Pair Jumper Cable 10ft	3	35	$105
		THIS QUOTE IS VALID UNTIL JUNE 23, 1998			
		DISCOUNTED QUOTE TOTAL			**$243,407**

Accepted by Customer:

Print or Type Name _____ Title _____

Signature _____ Date _____

Accepted by InterVision:

Print or Type Name _____ Title _____

Signature _____ Date _____

APR 22 '98 09:37 952 P03 INTERVISION 408-980-8893

NTERVISION
STEMS TECHNOLOGIES, INC.
UOTATION

Quotation Number MM042098-27 Page 1 of 3
Quotation Date April 22, 1998
(Quote is valid for 30 days)

'O:
Access Graphics
1426 Pearl St.
Boulder, Ca 80302

FROM:
InterVision Systems Technologies, Inc.
3218 Scott Blvd.
Santa Clara, CA 95054
Tel: 408-980-8550
Fax: 408-980-8893
Web: www.intervision.com

Ve are pleased to quote as follows:
Submitted By: Mark McEnroe Direct: 408 567-4212

teference:
Terms: Net 30 days
Credit Terms on approval of InterVision Credit Dept.
FOB: InterVision
Santa Clara, CA

tem	Model	Description	Qty.	Unit Net Price	Extended Net Price
01	L1-S6302-1	Origin200 Server 2XR10000 180MHz w/1MB Cache 128MB Memory, 4.5GB System Disk (no skins) incl IRIX 6.5	2	$ 14,158	$ 28,316
02	FCEW	Full Care Extended Warranty, L1-S6302-1	2	12,980	12,980
03	FTO-SI-TMOD	Skins for Deskside version Origin200 Single tower	2	N/C	N/C
04	OPN-512-D1	First 512MB memory in one bank	2	6,052	12,104
05	FCEW	Full Care Extended Warranty, OPN-512-D1	2	638	638
06	H4-N512-D1	Additional 512MB memory in one bank	6	6,052	36,312
07	FCEW	Full Care Extended Warranty, H4-N512-D1	6	3,840	3,840
08	FTO-4GB	4.5GB SE Ultra SCSI System Disk for Origin200 Sys.	2	N/C	N/C
09	PCI-FE-TX-MK	PCI 1-Port Fast Ethernet (100Base TX) Adapter with 1 Mouse/Keyboard Port for Origin200 Only	6	825	4,950
10	FCEW	Full Care Extended Warranty, PCI-FE-TX-MK	6	441	441
11	SC4-IRIX-6.5	IRIX 6.5 Operating System Software, bundled with hardware	2	N/C	N/C
12	DK-SL1-001	Destination Kit for Single Tower Origin200, Origin Vault, US/Canada (110VAC)	2	N/C	N/C

Estimated Shipment: _____ **Days After Receipt of Order, Subject to Change.**

PLEASE READ THIS QUOTATION CAREFULLY. THE TERMS AND CONDITIONS SET FORTH ON THE BACK OF THIS FORM CONSTITUTE THE ENTIRE AGREEMENT BETWEEN SELLER AND BUYER. SELLER WILL NOT BE BOUND BY ANY TERMS OF BUYER'S ORDER THAT ARE INCONSISTENT WITH THE TERMS HEREIN.

THIS QUOTATION REFLECTS INTERVISION'S PRICES AND ESTIMATED DELIVERY DATES TO BUYER FOR THE NEXT THIRTY (30) DAYS, UNLESS CONTRARY NOTICE IS GIVEN BY INTERVISION DURING SUCH THIRTY (30) DAY PERIOD. SHOULD BUYER PLACE AN ORDER FOR THE PRODUCTS SPECIFIED ABOVE, THE SALE SHALL BE SUBJECT SOLELY TO THE TERMS AND CONDITIONS CONTAINED ON THE FACE AND REVERSE SIDE HEREOF. TO PLACE AN ORDER, PLEASE SIGN THIS QUOTATION WHERE INDICATED AND RETURN A COPY TO INTERVISION

Accepted by Customer:

Print or Type Name _____ Title _____

Signature _____ Date _____

Accepted by InterVision:

Print or Type Name _____ Title _____

Signature _____ Date _____

APR 22 '98 09:37 P04 952 408-980-8893 INTERVISION

NTERVISION
TEMS TECHNOLOGIES, INC.
TOTATION

Quotation Number MM042098-27 Page 2 of 3

Quotation Date April 22, 1998 (Quote is valid for 30 days)

m	Model	Description	Qty.	Unit Net Price	Extended Net Price
3	SC4-C-7.2	MIPSpro C Compiler 7.2 Single User Floating or nodelock =2 CPU, for IRIX 6.2 to all 6.4	1	$ 570	$ 570
4	FCEW	Full Care Extended Warranty, SC4-C-7.2	1	441	441
5	MO5-CD	CD-ROM Update Media requirement- For Support Only	1	1,042	1,042
6	MO5-RTU	Right To Use Update Media requirement	1	103	103
		THIS QUOTE VALID UNTIL JUNE 23, 1998			
		DISCOUNTED QUOTE TOTAL			**$101,737**

Accepted by Customer:

Print or Type Name _____ Title _____

Signature _____ Date _____

Accepted by InterVision:

Print or Type Name _____ Title _____

Signature _____ Date _____

APR 22 '98 09:38 952 P05 INTERVISION 408-980-8893

BEA ENTERPRISE MIDDLEWARE SOLUTIONS

23 April, 1998

Mr. Jeff McDonald
System Performance Manager
Silicon Graphics, Inc.
2011 N. Shoreline Blvd.
Mountain View, CA

Dear Mr. McDonald;

Per your request I am enclosing the pricing information regarding TUXEDO 6.x that you requested. This pricing applies to Tuxedo 6.1, 6.2, 6.3 and 6.4. Please note that Tuxedo 6.4 is our most recent version of Tuxedo but that all 6.x releases are generally available. The pricing quotation in this letter is good for 60 days from the date of this letter. Core Functionality Services pricing is appropriate for your activities. As per the table below, server systems are classified in one of 5 tiers based on CPU type and capacity. An Origin 200 system with 2 RISC CPUs is classified as tier 2 system.

Tuxedo Core Functionality Services (CFS) Program Product Pricing and Description

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.x. Prices range from $3,000 for Tier 1 to $250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

BEA Tux/CFS Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 8) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers	Unlimited	$3,000.00	$450.00	$660.00
Tier 2 -- PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations	Unlimited	$12,000.00	$1,800.00	$2,640.00
Tier 3 -- Midrange Multiprocessors, 2 to 8 CPUs per system capacity	Unlimited	$30,000.00	$4,500.00	$6,600.00
Tier 4 -- Large (more than 8, less than 32 CPUs) and Mainframe	Unlimited	$100,000.00	$15,000.00	$22,000.00

1

Systems				
Tier 5 -- Massively Parallel Systems, > 32 processors	Unlimited	$250,000.00	$37,500.00	$55,000.00

Silicon Graphics Tier Classifications

	Tier 1	Tier 1	Tier 2	Tier 3	Tier 3	Tier 4	Tier 5
Platform	**Class 1**	**Class 2**	**Class 3**	**Class 4**	**Class 5**	**Class 6**	**Class 7**
Silicon Graphics			Challenge S Origin 200	Challenge DM,	All Power servers, Origin2000 Deskside,	Challenge L	Origin2000 Rack, Challenge XL

Very Truly Yours,

Lewis D. Brentano,
Director, Market Planning

NETLUX

14180 Live Oak Ave., Unit E
Baldwin Park, Ca. 91760

1-800-789-1780
Phone #818-851-9737
Fax #818-851-9837

April 17, 1998

Silicon Graphics Computer Sys
Jeff McDonald
2011 North Shoreline Blvd.
Mountain View, CA 94043

Quotation

Quantity	*Part No.*	*Description*	*Unit Price*	*Total*
3	NX-DH4	4-port 100Mbps FAST Ethernet Hub	$85.00	$255.00
700	NX-H9+	(8+1) 9-port 10Mbps Ethernet Hub	$40.00	$28,000.00

Terms and Conditions:
FOB Origin
5 Year Warranty
Prices good for 60 Days

Sincerely,
Martin Parry
NETLUX