BULL S.A.

ESCALA
Deskside D404/8

TPC Benchmark™C
Full Disclosure Report

Client/Server Configuration
Using Sybase SQL server 11.0.3

Special notices

The following terms used in this publication are trademarks of the BULL company in the United States and/or other countries:

BULL
Estrella
ESCALA

The following terms used in this publication are trademarks of the IBM company in the United States and/or other countries:

AIX
IBM
RISC System/6000

The following terms used in this publication are trademarks of other companies as follows:

TPC Benchmark C	Trademark of the Transaction Processing Performance Council
Sybase SQL Server	Trademark of Sybase,Inc.
Sybase Open Client	Trademark of Sybase,Inc.
Tuxedo System	Trademark of UNIX System Laboratories, Inc.

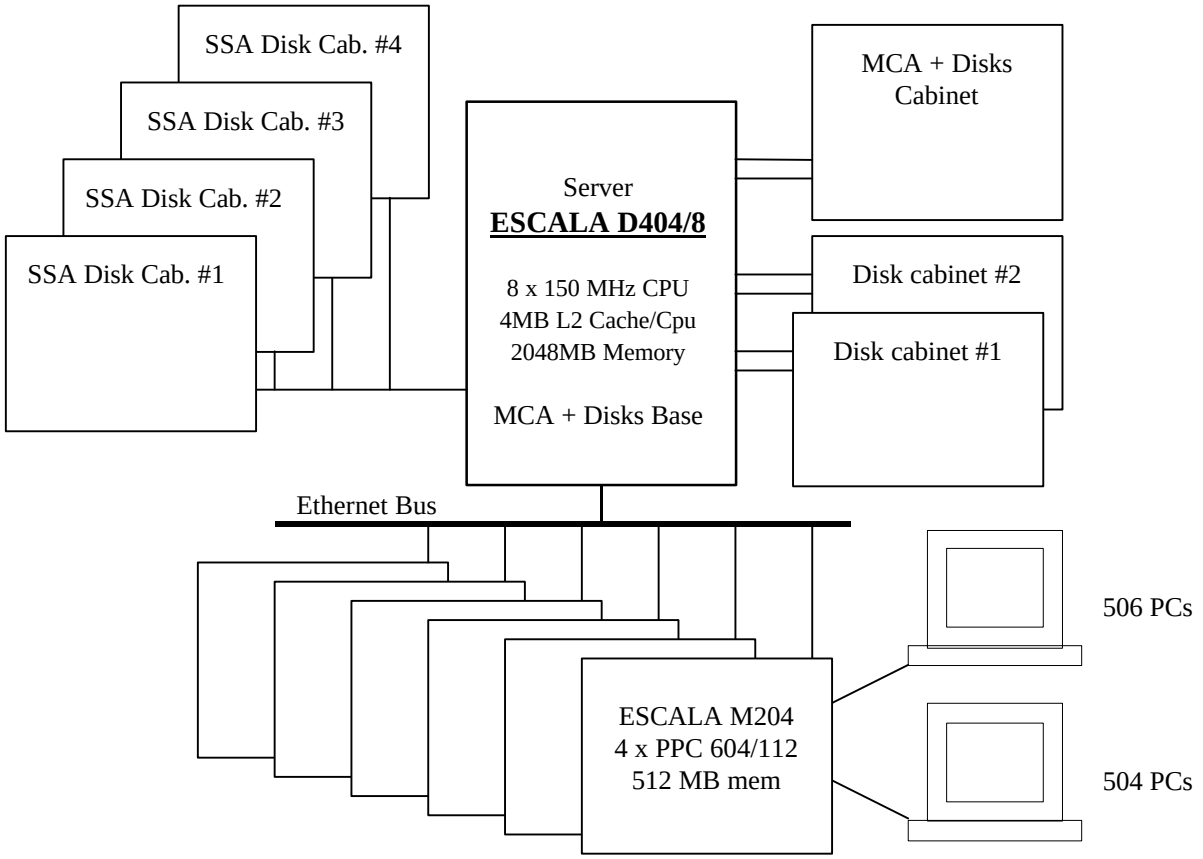
First Edition November 14, 1996

Requests for additional copies of this report should be sent to the following address:

TPC
C/O Shanley Public Relations
777 N. First St., Suite 600
San Jose, CA 95112-6113
USA

BULL S.A.	ESCALA D404/8		TPC - C Rev.3.2 Report Date November 14, 1996
Total System	TPC - C Throughput	Price / Performance	Availability date
$ 1,429,252	**7,303.67 tpmC**	**$195.68 per tpmC**	January 1997

Processors	Database Manager	Operating System	Other Software	Number of users
8 Power PC 604 @ 150MHz	Sybase SQL server 11.0.3	System AIX4.2.0	TCP-IP Tuxedo 4.2.2 V2	6060



6 Clients :
2 thinwire segments per client
Total of 6060 PCs

System Components	Server		Clients	
	Qty	Type	Qty	Type
Processors	8	Power PC 604e @150MHz	6	4* PPC 604 @ 112 MHz
Cache L2	8	4 MB L2 cache per Cpu	6	1 MB L2 cache per Cpu
Memory (MB)	2048 MB	(4 x 512MB)	6	512 MB
Disk Controllers	9	(1 SSA + 8 FW SCSI 2)	18	integrated SCSI adapter
Disk Drives	128	(64 x 4.5GB + 64 x 4.2GB)	12	2.1GB SCSI II
Total Storage (GB)	537 GB			
CDROM	1	CDROM		
Terminals	1	BQ306 console	6	BQ306 console

Bull S.A.	ESCALA D404-8 Client/Server	TPC-C REV 3.2 EXECUTIVE SUMMARY PAGE 2 OF 2 Report Date: Nov. 14, 96

Description	Part Number	Third Party		Unit Price	Qty	Extended Price	5 yr. Maint. Price
Server Hardware		**Brand**	**Pricing**				
ESCALA D404 8WAY PACKAGED SYSTEM (8 *PPC604e, 1GB mem, 12*4.2GB, CDROM, Eth)	CPXG145-U0000			178000	1	178000	28272
3.5M CABLE LOCAL RS232 (D25F/D25M)	CBLG104-1600			100	1	100	0
5M CABLE ETH TO TRANSCEIVER	VCW3630			150	1	150	0
2M SCSI CABLE (SCSI2-DE basic or exp. cab.to other cabinet)	CBLG102-1300			245	6	1470	0
SYSTEM CONSOLE BQ 306 (White display, AZERTY kbd, conn. cable)	CSUG002-1100			620	1	620	48
4.2GB HI SPEED SCSI-2 DISK DRIVE (CRU)	MSUG066-0C00			2970	4	11880	3648
512MB Memory module	CMMG023-0000			31500	2	63000	0
100.8 GB Disk expension cabinet	CABG018-U0000			61960	2	123920	44352
3"1/2 to 5"1/4 form factor adaptor	CKTG047-0000			88	3	264	0
Supply InterFace Board (SIF10)	PSSG003-0000			440	3	1320	0
SCSI-2 F/W DE Ext Disk Adapter	MSCG012-0000			1370	6	8220	0
SSA 4-PORT ADAPTER	MSCG021-0000			1370	1	1370	0
SSA DISK SUBSYSTEM DESKSIDE with 16*4.5GB disks	SSAG001-U000			57606	4	230424	84096
					Subtotal	620738	160416
Server Software							
SQL Server V11, Open Client, 5%discount		Sybase	2	152375	1	152375	128316
					Subtotal	152375	128316
Client Hardware							
ESCALA M204 MINITOWER Packaged SYSTEM (4*PPC604, 512MB,2* DK 2.1GB, CD-ROM, ETH)	CPXG146-U000			52830	6	316980	75456
ETHERNET LAN ADAPTER (LSA)	DCCG074-0000			720	12	8640	0
					Subtotal	325620	75456
Client Software							
BEA TUXEDO R4.2.2		BEA	3	3500	6	21000	15750
					Subtotal	21000	15750
User Connectivity							
24 PORT 10base-T HUB		Netlux	1	251	259	65009	0
24 PORT 10base-T HUB (spare 10%)		Netlux	1	251	26	6526	0
					Subtotal	71535	0
					Other Discounts	-141954	0
					Total	**1049314**	**379938**

Notes:

*** Dollar volume discount 15% on Bull Hardware**

1=Netlux, 2=Sybase, 3=BEA

Audited by Franǫis Raab, Information Paradigm

Five-Year Cost of Ownership	**1429252**
tpmC Rating:	**7303.67**
$ / tpmC:	**195.68**

Numerical Quantities for the BULL ESCALA D404/8

MQTH, Computed Maximum Qualified Throughput **7303.67 tpmC**

Response Times (in seconds)	90th %-ile	Maximum	Average
New-Order	**1.10**	**5.19**	**0.65**
Payment	**1.00**	**12.08**	**0.59**
Order-Status	**1.07**	**3.77**	**0.72**
Delivery (interactive portion)	**0.20**	**1.12**	**0.17**
Delivery (defered portion)	**99.61 % < 80 sec**	**-**	**1.69**
Stock-Level	**15.24**	**40.01**	**7.10**
Menu	**0.21**	**2.55**	**0.15**

Transaction Mix, in percent of Total transactions

New-Order	44.68 %
Payment	43.19 %
Order-Status	4.05 %
Delivery	4.03 %
Stock-Level	4.03 %

Keying / Think Times (in seconds)	Keying Time			Think Time		
	Min.	Avg.	Max.	Min.	Avg.	Max.
New-Order	18.00	18.08	19.15	0.01	12.21	122.04
Payment	3.00	3.04	4.12	0.01	12.24	122.03
Order-Status	2.00	2.04	2.81	0.01	10.61	100.56
Delivery (interactive)	2.00	2.03	2.96	0.01	5.10	51.02
Stock-Level	2.00	2.03	2.96	0.01	5.10	51.14

Test Duration

Ramp-up time	30 minutes
Measurement interval	30 minutes
Transaction during interval	490.384
Ramp-down time	15 minutes
Checkpointing	
Number of checkpoints	1
Checkpoint interval	30 minutes

Reproductibility Run **7298.40 tpmC** -0.07%

Preface

TPC Benchmark™ C Standard Specification was developed by the Transaction Performance Council (TPC) and released on August 13, 1992 and updated with revision 3.0 on February 15, 1995.

This is the full disclosure report for benchmark testing of the BULL ESCALA according to the TPC Benchmark™ C Standard Specification.

TPC Benchmark™ C exercises the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing a mixture of read-only and update intensive transactions. This is a complex OLTP application environment exercising a breadth of system components associated by such environments characterized by:

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

This benchmark defines four on-line transactions and one deferred transaction, intended to emulate functions that are common to many OLTP applications. However, this benchmark does not reflect the entire range of OLTP requirements. 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 any other environment are not recommended.

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

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 a warehouse, and each transaction is subjected to a response time constraint. The performance metric for this benchmark is expressed in transactions-per-minute-C (tpmC). To be compliant with the TPC-C standard, all references to tpmC results must include the tpmC rate, the associated price-per-tpmC, and the availability date of the priced configuration.

1. GENERAL ITEMS 3

1.1 Application Code Disclosure 3
1.2 Benchmark Sponsor 3
1.3 Parameter Settings 3
1.4 Configuration Diagrams 3

2. CLAUSE 1 : LOGICAL DATABASE DESIGN RELATED ITEMS 7

2.1 Data and Space Definition 7
2.2 Physical Organization of Data 7
2.3 Insert and Delete Operations 7
2.4 Database partitioning 7

3. CLAUSE 2 : TRANSACTION AND TERMINAL PROFILES RELATED ITEMS 8

3.1 Random Number Method 8
3.2 Terminal Screen Layouts 8
3.3 Priced Terminal Features 8
3.4 Presentation Manager or intelligent Terminal 8
3.5 Transaction Statistics 9
3.6 Transaction Types Mix 9
3.7 Queuing Mechanism 9

4. CLAUSE 3 : TRANSACTION AND SYSTEM PROPERTIES RELATED ITEMS 10

4.1 ACID Properties 10
4.2 Atomicity Tests 10
4.2.1 Atomicity of Completed Transaction 10
4.2.2 Atomicity of Aborted Transaction 11
4.3 Consistency Tests 11
4.3.1 Consistency Condition 1 11
4.3.2 Consistency Condition 2 11
4.3.3 Consistency Condition 3 11
4.3.4 Consistency Condition 4 11
4.4 Isolation Tests 12
4.4.1 Isolation Test 1 12
4.4.2 Isolation Test 2 12
4.4.3 Isolation Test 3 13
4.4.4 Isolation Test 4 13
4.4.5 Isolation Test 5 14
4.4.6 Isolation Test 6 14
4.4.7 Isolation Test 7 15
4.4.8 Isolation Test 8 15
4.4.9 Isolation Test 9 16
4.5 Durability Tests 16
4.5.1 Durable Media Failure 16
4.5.2 Instantaneous Interruption and Loss of Memory 17

5. CLAUSE 4 : SCALING AND DATABASE POPULATION RELATED ITEMS 18

5.1 Initial Cardinality of Tables 18
5.2 Distribution of Tables and Logs 18
5.3 Data Model and database interface 21
5.4 DBMS partitions and replications 21
5.5 DBMS space requirements 21

6. CLAUSE 5 : PERFORMANCE METRICS AND RESPONSE TIME RELATED ITEMS 22

6.1 Measured Throughput (tpmC) 22
6.2 Response Times 22
6.3 keying and Think Times 22
6.4 Frequency Distribution of Response Times 22
6.5 Performance Curve for Response Time Vs Throughput 25
6.6 Frequency Distribution of Think Times 26

6.7 THROUGHPUT VS ELAPSED TIME : NEW-ORDER TRANSACTION 26
6.8 STEADY STATE DETERMINATION 27
6.9 WORK PERFORMED DURING STEADY STATE 27
6.9.1 Transaction Flow *27*
6.9.2 DataBase Transaction *28*
6.9.3 Checkpoints *28*
6.10 REPRODUCIBILITY METHOD 28
6.11 MEASUREMENT INTERVAL 28

7. CLAUSE 6 : SUT, DRIVER, AND COMMUNICATION DEFINITION. RELATED ITEMS 29

7.1 RTE INPUTS 29
7.2 DRIVER FUNCTIONALITY AND PERFORMANCE 29
7.3 FUNCTIONAL DIAGRAMS AND DETAILS OF DRIVER SYSTEM 29
7.4 NETWORK CONFIGURATIONS AND DRIVER SYSTEM 29
7.5 NETWORK BANDWIDTH 29
7.6 OPERATOR INTERVENTION 29

8. CLAUSE 7 : PRICING RELATED ITEMS 31

8.1 SYSTEM PRICING 31
8.2 SUPPORT 31
8.3 DISCOUNTS 31
8.4 AVAILABILITY STATUS 32
8.5 THROUGHPUT PRICE/PERFORMANCE AND AVAILABILITY 32
8.6 SPACE CALCULATIONS 32

9. CLAUSE 9 : AUDITOR RELATED ITEMS 33

9.1 AUDITOR'S ATTESTATION LETTER 34

10. PRICE QUOTATION 36

1. General Items

1.1 Application Code Disclosure

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

Appendix A contains application code listings for the five transaction types.

1.2 Benchmark Sponsor

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

Bull SA is the sponsor 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*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating systems and application configuration parameters*
- *Compilation and linkage options run-time optimizations used to create / install applications, OS, and / or databases.*

Appendix C contains the system, data base and application parameters changed from their default values used in this bench, it also contains the compilation and linkage options used to create the applications on the server and on the clients.

1.4 Configuration Diagrams

Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *Number and type of processors*
- *Size of allocated memory, and any specific mapping / partitioning of memory unique to the test.*
- *Number and type of disk units (and controllers, if applicable).*
- *Number of channels or bus connections to disk units, including 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 run time execution location of software components (e.g., DBMS, client processes, transaction monitors, software drivers, etc.).*

The benchmark configuration used Remote Terminal Emulator (RTE) programs that executed on Estrella MT604/133MHz driver systems to emulate PCs, and Ethernet connections.
The benchmark configurations are illustrated in Figure 1.

The emulated users on the driver system were directly connected to the client systems under test via local area network (LAN) and communicated using TCP/IP.

The Escala priced configuration is shown in Figure 2 , the RTE is replaced by the appropriate number of PC users connected to the clients through Ethernet.

The clients consisted of 6 Escala systems running the Tuxedo System/T, each emulated terminal ran a Tuxedo client application and made a request to Tuxedo services. The Tuxedo services are processes that send request to the SQL server engine.

The server was Escala D404/8 running SQL server 11.0.3

During the measurement, the Escala D404/8 was configured with 128 disks (64 * 4.5GB + 64 * 4.2GB).
The 128 disks are distributed between:

- The basic cabinet
- The MCA expansion cabinet with up to 4 CRU (customer replacable unit) disks in 1 SCSI bus
- 4 SSA Disk Cabinets (SSAG001-U000) with 64 SSA Disks (16 in each cabinet)
- 2 large size extension cabinets configured with 48 disks in 4 SCSI busses (12 disks on each SCSI, 24 disks in each cabinet)

The clients were 6 Escala M204 4*PPC604/112MHz

The priced Configuration is the same than the Benchmark Configuration.

Note: Disks are declared as 4.5GB or 4.2 GB; this is the unformatted capacity; the formatted capacities are respectively 4.19 and 3.99 GB (1GB = 1024^3 Bytes).

Figure 1 Benchmark configuration Escala D404/8

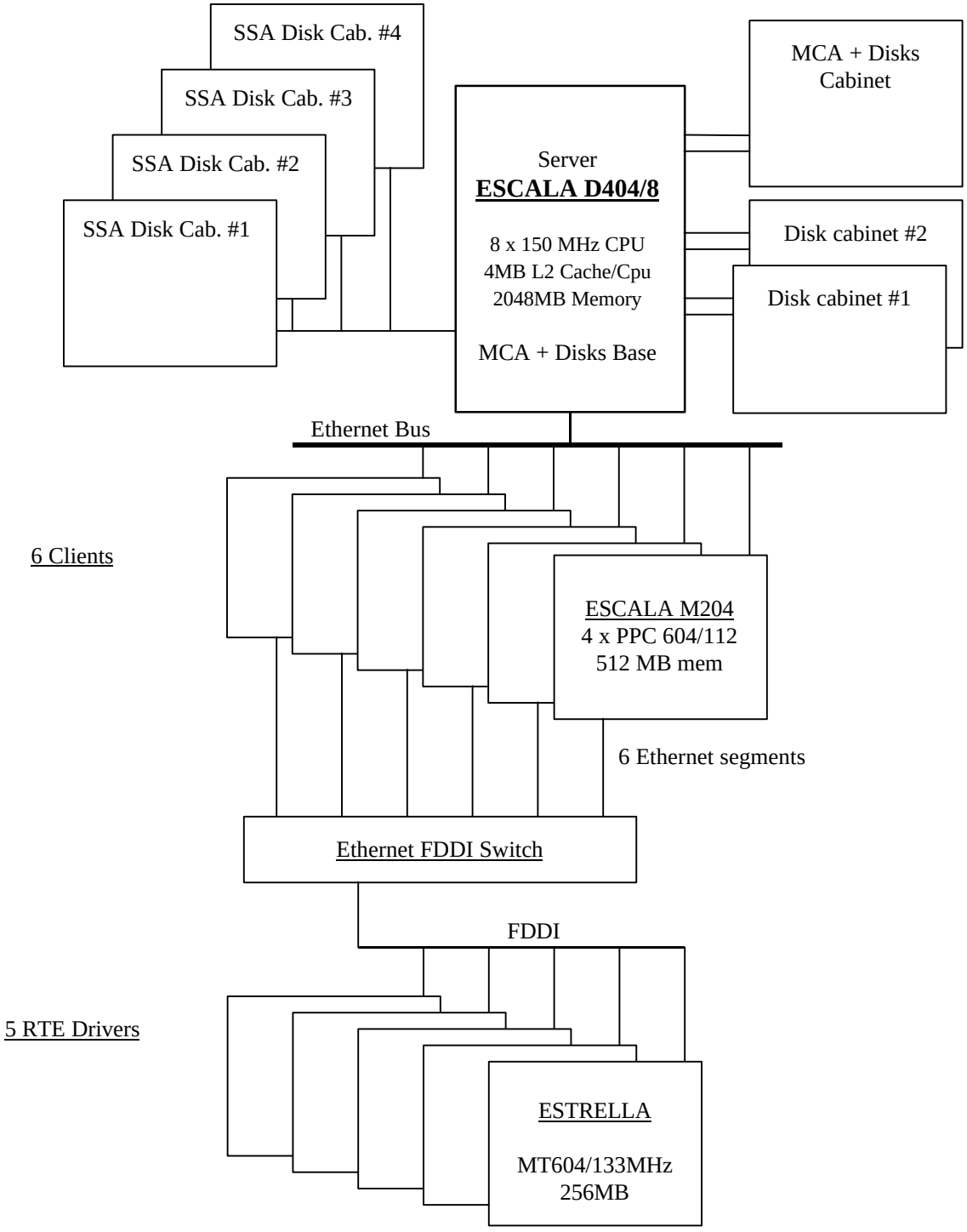
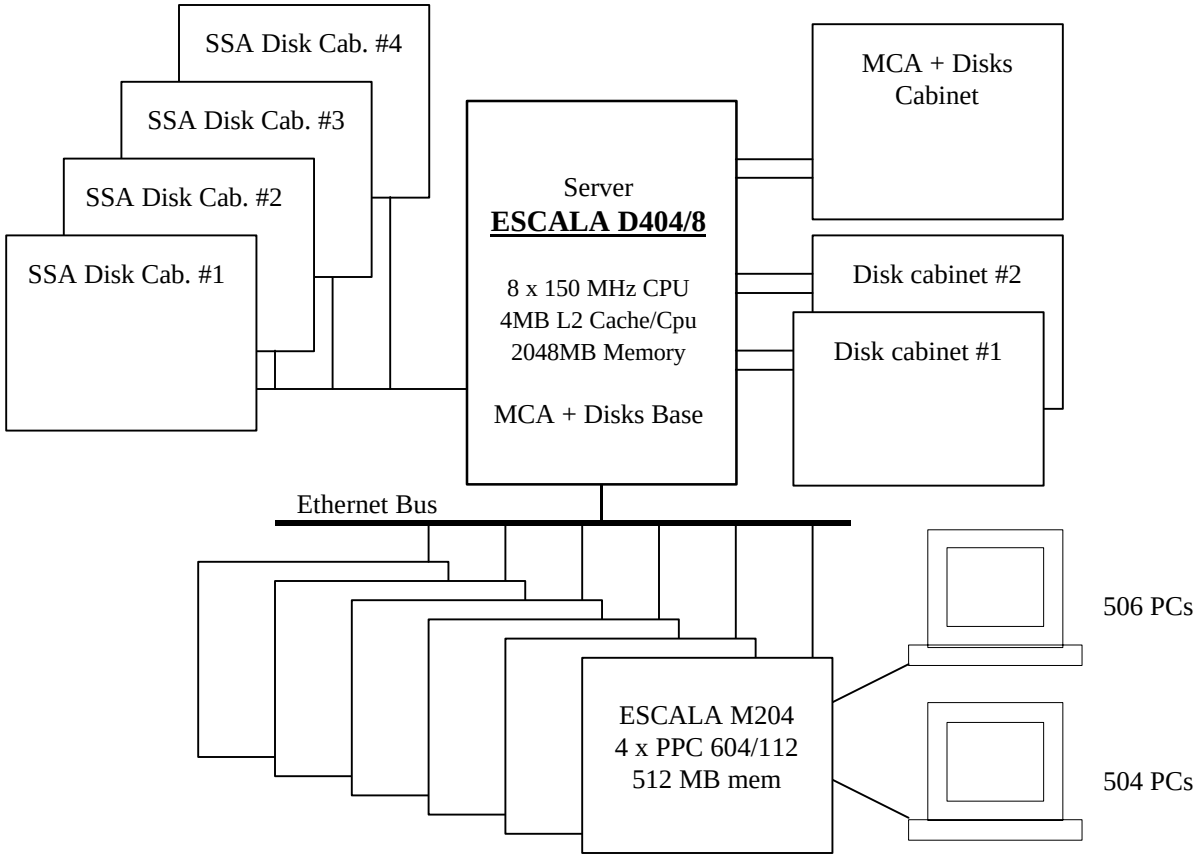


Figure 2 Priced configuration Escala D404/8



6 Clients :
2 thinwire segments per client
Total of 6060 PCs

2. Clause 1 : Logical Database Design Related Items

2.1 Data and Space Definition

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

Appendix B descrives the programs that define, create and populate an SQL server 11 database for TPC - C testing.

2.2 Physical Organization of Data

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

Space was allocated to SQL server 11 on SUT disks according to the details provided in section 5.2. The size of the Database table space on each disk was calculated to provide even distribution of load across the disk subsystem.
AIX Logical Volume Management was used for disk stripping and for log mirroring.

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 preclude 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. The space required for additional 5 % of initial table cardinality was allocated to SQL server 11 and priced as static space.

2.4 Database partitioning

While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark, any such partitioning must be disclosed.

Partitioning was not used in this implementation.

3. Clause 2 : Transaction and Terminal Profiles Related Items

3.1 Random Number Method

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

The srandom(), getpid(), gettimeofday() functions are used to produce unique random seeds for each driver. The drivers use these seeds to seed the srand(), srandom() and srand48() functions. Random numbers are produced using wrappers around the standard system random number generators.
The negative exponential distribution uses the following function to generate the distribution. This function has the property of producing a negative exponential curve with a specified average and a maximum value 4 times the average.

const double RANDOM_4_Z=0.89837799236185
const_double RANDOM_4_K=0.97249842407114

The random functions used by the driver system and the data base generation program were verified. The C_LAST column was queried to verify the generation of random values produced by the database generation program. After a measurement, the HISTORY, ORDER, and ORDER_LINES tables were also queried. The rows were counted and grouped by customer and item numbers. Here is an example of the SQL query we used to verify the random number generation functions:

```
create table TEMP (W_ID int, D_ID, C_LAST char(16), CNTR int);
insert into TEMP select C_W_ID, C_D_ID, C_LAST, COUNT(*) from CUSTOMER
group by C_W_ID, C_D_ID, C_LAST;
select CNTR, COUNT(*) from TEMP group by CNTR order by 1.
```

3.2 Terminal Screen Layouts

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

The screen layouts correspond exactly to the layouts in clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3 and 2.8.3 of the TPC-C specifications.

3.3 Priced Terminal Features

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).

A Bull Zenith Z-station GT (P90) running an ANSI terminal emulator was physically connected to the system under test and the TPC-C application program was run. Each transaction type was executed and all functions described in clause 2.2.2.4 were verified.

3.4 Presentation Manager or intelligent Terminal

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

Application code running on the client implemented the TPC - C user interface. A listing of this code is included in Appendix A. Used capabilities of the terminal beyond basic ASCII entry and display were restricted to cursor positioning.
A presentation manager was not used.

3.5 Transaction Statistics

Clauses 8.1.3.5 through 8.1.3.11 are represented in the table below.

Description	
Percentage of home and remote order-lines in New-Order	**99.01 : 0.99**
New-Orders rolled back as a result of an illegal item	**0.98 %**
Average items per orders entered by New-Order	**10.00**
Percentage of home and remote payments	**85.15 : 14.85**
Percentage of Payment that used C_LAST	**60.01 %**
Percentage of Order-Status that used C_LAST	**59.96 %**
Percentage of Delivery transactions with no NEW-ORDER row	**0 %**

3.6 Transaction Types Mix

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

The following table summarizes the transaction mix:

Transaction mix (percent)

Transaction type	**New-Order**	**Payment**	**Order-Status**	**Delivery**	**Stock-Level**
Min.Requirement	NA	43%	4%	4%	4%
Measured Result	44.68 %	43.19%	4.05%	4.03%	4.03%

3.7 Queuing Mechanism

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

Delivery transactions were submitted using an asynchronous call to a Tuxedo System/T service. Tuxedo System/T returns an immediate response to the calling program and schedules the work to be performed. This allows the Delivery transaction to be submitted, obtain an interactive response and queue the actual database transaction for deferred execution. Please see application code in Appendix A for details.

4. Clause 3 : Transaction and System Properties Related Items

4.1 ACID Properties

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

Clause 3 of the TPC Benchmark C Standard Specification lists specific tests to ensure the Atomicity, Consistency, Isolation and Durability (ACID) properties of the SUT. The following subsections show how the tests required in Clause 3 were performed and the results verified. All mechanisms needed to ensure full ACID properties were enabled during both the measurement and test periods. These tests have been executed during a pre-audit phase done with the Auditor, then on the final configuration. A fully sized warehouse database was used for the power failure Durability tests

Case D was followed for the execution of Isolation Test 7, as described in the TPC - C Standard Specification, Clause 3.4.2
To the best of the test sponsor's knowledge , a fully-loaded and fully-scaled system would pass the data file durability test.

4.2 Atomicity Tests

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

4.2.1 Atomicity of Completed Transaction

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2-5.12) and verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have been changed appropriately.

The following test was performed, and verified the atomicity of completed transactions:

1. Select a random warehouse, district, customer, and payment amount

2. For the selected warehouse, district, and customer, record their contents.

3. Perform a PAYMENT transaction, and COMMIT the transaction.

4. For the selected warehouse, district, and customer, verify the following:

   A. W_YTD in the warehouse is increased by the payment amount

   B . D_YTD in the district is increased by the payment amount.

   C. C_YTD_PAYMENT in the customer is increased by the payment amount, C_BALANCE is decreased by the payment amount, and C_PAYMENT_CNT is incremented by 1.

4.2.2 Atomicity of 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 following test was performed, and verified the atomicity of aborted transactions:

1 . Select a random warehouse, district, customer, and payment amount

2. For the selected warehouse, district, and customer, record their contents.

3. Perform a PAYMENT transaction and substitute a ROLLBACK for the COMMIT.

4. For the selected warehouse, district, and customer, verify that records have not been changed.

4.3 Consistency Tests

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

The consistency of the database was checked after the performance measurement run.
The following consistency conditions were run and verified that all four conditions were met:

4.3.1 Consistency Condition 1

Entries in the WAREHOUSE and DISTRICT tables satisfy:
W_Y'TD = sum(D_YTD) for each warehouse defined by (W_ID = D_W_ID)

4.3.2 Consistency Condition 2

Entries in the DISTRICT, ORDER, and NEW-ORDER tables satisfy:
D_NEXT_O_ID -1= max(O_ID) = max(NO_O_ID) for each district defined by (D_W_ID = O_W_ID = NO_W_ID) and (D_ID = O_D_ID = NO_D_ID)

4.3.3 Consistency Condition 3

Entries in the NEW-ORDER table satisfy:
max(NO_O_ID) - min(NO_O_ID) +1 = [number of rows in NEW-ORDER of this district]
for each district defined by NO-W-ID and NO_D_ID

4.3.4 Consistency Condition 4

Entries in the ORDER and ORDER-LINE tables safisfy:
sum(O_OL_CNT) = [# of rows in ORDER-LINE of this district] for each district defined by (O_W_ID=OL_W_ID) and (O_D_ID = OL_D_ID)

4.4 Isolation Tests

The TPC Benchmark C Standard Revision 3.0 defines ten required tests to be performed to demonstrate that the required levels of transaction isolation are met. All ten required tests were performed successfully.

4.4.1 Isolation Test 1

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

The execution of the above test proceeded as follows:

1. An Ordcr-Status transaction T0 was executed for a randomly selected customcr, and the order returned was noted. T0 was committed.

2. A New-Order transaction T1 was started for the same customer used in T0. T1 was stopped prior to COMMIT.

3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.

4. T1 was allowed to complete and was committed.

5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the order inserted by T1.

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

4.4.2 Isolation Test 2

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

The execution of the above test proceeded as follows:

1. An Order-Status transaction T0 was executed for a randomly selected customer, and the order returned was noted. T0 was committed.

2. A New-Order transaction T1 with an invalid item number was started for the same customer used in T0. T1 was stopped immediately prior to ROLLBACK.

3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.

4. T1 was allowed to ROLLBACK.

5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the same order that T0 had returned.

4.4.3 Isolation Test 3

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

The execution of the above test proceeded as follows:

1. The D-NEXT_O_ID of a randomly selected district was retrieved.

2. A New-Order transaction T1 was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to ROLLBACK.

3. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.

4. T1 was allowed to complete. T2 completed and was committed.

5. Thc order number returned by T1 was the same as the D_NEXT_O_ID retrieved in step 1. The order number returned by T2 was one greater than the order number returned by T1.

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

4.4.4 Isolation Test 4

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

The execution of the above test proceeded as follows:

1. The D_NEXT_O_ID of a randomly selected district was retrieved.

2. A New-Order transaction T1 with an invalid item number was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to ROLLBACK.

3. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.

4. T1 was allowed to roll back, and T2 completed and was committed.

5. The order number returned by T2 was the same as the D_NEXT_O_ID retrieved in step 1.

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

4.4.5 Isolation Test 5

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

The execution of the above test proceeded as follows:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.

2. The C_BALANCE of the customer found in step 1 was retrieved.

3. A Delivery business transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the COMMIT of the database transaction corresponding to the district used in step 1.

4. A Payment transaction T2 was started for the same customer found in step 1. T2 waited.

5. T1 was allowed to complete. T2 completed and was committed.

6. The C_BALANCE of the customer found in step 1 was retrieved again. The C_BALANCE reflected the results of both T1 and T2.

4.4.6 Isolation Test 6

This test demonstrates isolation for write-write conflicts of Pavment and Delivery transactions when the Delivery transaction is rolled back.

The execution of the above test proceeded as follows:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.

2. The C_BALANCE of the customer found in step 1 was retrieved.

3. A Delivery business transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the ROLLBACK of the database transaction corresponding to the district used in step 1.

4. A Payment transaction T2 was started for the same customer found in step 1. T2 waited.

5. T1 was allowed to ROLLBACK. T2 completed and was commined.

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

4.4.7 Isolation Test 7

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

T'he execution of the above test proceeded as follows:

1. The I_PRICE of two randomly selected items X and Y were retrieved.

2. A New-Order transaction T2 with a group of items including items X and Y was started. T2 was stopped immediately, after retrieving the prices of all items. The prices of items X and Y retrieved matched tbose retrieved in step 1.

3. A transaction T3 was started to increase the price of items X and Y by 10%.

4. T3 did not stall and no transaction was rolled back. T3 was committed.

5. T2 was resumed, and the prices of all items were retrieved again within T2. The prices of items X and Y matched those retrieved in step 1.

6. T2 was committed.

7. The prices of items X and Y were retrieved again. The values matched the values set by T3.

4.4.8 Isolation Test 8

This test demonstrates isolation for phantom protection between New-Order and Order-Status transactions. The execution of the above test proceeded as follows:

1. An Order-Status transaction T1 was started for a randomly selected customer.

2. T1 was stopped immediately after reading the order table for the selected customer. The most recent order for that customer was found.

3. A New-Order transaction T2 was started for the same customer. T2 completed and was committed without being blocked by T1.

4. T1 was resumed and the order table was read again to determine the most recent order for the same customer. The order found was the same as the one found in step 2.

5. T1 completed and was committed.

4.4.9 Isolation Test 9

This test demonstrates isolation for phantom protection between New-Order and Delivery transactions. The execution of the above test proceeded as follows:

1. The NO_D_ID of all new-order rows for a randomly selected warehouse and district was changed to 11. The changes were committed.

2. A Delivery transaction T1 was started for the selected warehouse.

3. T1 was stopped immediately after reading the new-order table for the selected warehouse and district. No qualifying row was found.

4. A New-Order transaction T2 was started for the same warehouse and district T2 completed and was committed without being blocked by T1.

5. T1 was resumed and the new-order table was read again. No qualifying row was found.

6. T1 completed and was committed.

7. The NO_D_ID of all new_order rows for the selected warehouse and district was restored to the original value. The changes were committed.

4.5 Durability Tests

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

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

Tests were conducted for each of the preceding types of failures and successfully demonstrated that the durability properties were met.

4.5.1 Durable Media Failure

Permanent irrecoverable failure of any single durable medium containing TPC-C database tables was demonstrated in the following test on a database scaled for 10 warehouses. The standard driving mechanism was used to generate the transaction load of 100 users. The fully scaled database under full load would also have passed the following test.

1. The scaled database was built aad backed up to extra disks

2. The total number of New Orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving the beginning count.

3. The RTE was started with 100 users.

4. The test was allowed to run for a minimum of 10 minutes.

5. One of the data disks was overwritten using the dd command.

6. When the Sybase dataserver recorded corruption errors the RTE and database were shutdown.

7. The dataserver was restarted and a dump of the transaction log was taken.

8. The database was dropped and recreated as an empty database.

9. The database was restored from the database backup, followed by the restore of the transaction log.

10. Consistency condition #3 was executed and verified

11. Step 2 was repeated and difference between the first and the second counts was noted.

12. A RTE report was generated for the entire run time giving the number of NEW_ORDERS successfuly returned to the RTE.

13. The counts in step 12 and 13 were compared and the results verified.

14. The difference between the counts was less than the number of server connections.

15. Samples were taken from the RTE and used to query the database to demonstrate successful transactions had corresponding rows in the ORDER table.

4.5.2 Instantaneous Interruption and Loss of Memory

Instantaneous interruption, loss of memory tests and loss of transaction log data were combined in the following test on a fully scaled database under the full load of 6060 terminals. The following steps were executed.

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

2. A test was executed with 6060 terminals. On the driver system, completed / rolled-back New_Order transactions were recorded as such in a « success » file.

3. After four minutes at full load a log disk from the operating system mirroring was removed and the system continued to run with no failures for another two minutes.

4. After six minutes at full load the server system was unpowered by unplugging the power cord.

5. The test was aborted on the driver RTE.

6. The server system was restarted.

7. The database was restarted and a recovery performed using the transaction log.

8. The contents of the « success » file on the driver and the orders table were spot-compared to verify that records in the « success » file for completed New_Order transactions had corresponding records in the orders table and no entries appeared for rolled-back transactions.

9. Step 1 was repeated to determine the current total number of orders (count2). Count2-count1 was not less than the number of records in the « success » file. *This difference would be due only to transactions witch were committed on the system under test but for which the data was not displayed on the (emulated) input / output screen before the failure.*

5. Clause 4 : Scaling and Database Population Related Items

5.1 Initial Cardinality of Tables

The cardinality (e.g., the number of rows) of each table, as it existed at the start of the benchmark run (see Clause 4.2), must be disclosed.

The TPC-C database for this test was configured with 620 warehouses. Before the measured runs, WAREHOUSE rows 607 to 620 were deleted.

Figure 3 Table population

table	Cardinary as build	Cardinary as benchmarked
warehouse	**620**	**606**
district	**6,200**	**6,200**
customer	**18,600,000**	**18,600,000**
history	**18,603,539**	**18,603,539**
orders	**18,627,929**	**18,627,929**
new_order	**5,633,856**	**5,633,856**
order_line	**186,030,124**	**186,030,124**
item	**100,000**	**100,000**
stock	**62,000,000**	**62,000,000**

5.2 Distribution of Tables and Logs

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

Figure 4 ESCALA D404/8 data distribution Benchmark Configuration

CONTROLLER	DISK	TABLE NAME	FILENAME	MODE	Used (MB)	Capacity (GB)
lsa1	hdisk0		rootvg + swap	std	2392	4.2
	hdisk1		swap	std	592	4.2
	hdisk2		sybase	std	560	4.2
ascsi1	hdisk4	LOG	log1	std	1800	4.2
	hdisk5	ORDER_LINES	ordlx	strip10	1840	4.2
	hdisk6	ORDER_LINES	ordlx	strip10	1840	4.2
	hdisk7	ORDER_LINES	ordlx	strip10	1840	4.2
	hdisk8	ORDER_LINES	ordlx	strip10	1840	4.2
	hdisk9	ORDER_LINES	ordlx	strip10	1840	4.2
	hdisk10	ORDER_LINES	ordlx	strip10	1840	4.2
	hdisk11	ORDER_LINES	ordlx	strip10	1840	4.2
	hdisk12	ORDER_LINES	ordlx	strip10	1840	4.2
	hdisk13	ORDER_LINES	ordlx	strip10	1840	4.2
	hdisk14	ORDER_LINES	ordlx	strip10	1840	4.2
asci2	hdisk16	LOG	log2	std	1800	4.2
	hdisk17	CAHISTORDER	cahiordx	strip8	384	4.2
	hdisk18	CAHISTORDER	cahiordx	strip8	384	4.2
	hdisk19	CUSTOMER	cust1	std	472	4.2
	hdisk20	CUSTOMER	cust2	std	472	4.2
	hdisk21	CUSTOMER	cust3	std	472	4.2
	hdisk22	CUSTOMER	cust4	std	472	4.2
	hdisk23	STOCKS	stk1	std	368	4.2
	hdisk24	STOCKS	stk2	std	368	4.2
	hdisk25	STOCKS	stk3	std	368	4.2
	hdisk26	STOCKS	stk4	std	368	4.2

CONTROLLER	DISK	TABLE NAME	FILENAME	MODE	Used (MB)	Capacity (GB)
ascsi3	hdisk28	LOG	log3	std	1800	4.2
	hdisk29	CAHISTORDER	cahiordx	strip8	384	4.2
	hdisk30	CAHISTORDER	cahiordx	strip8	384	4.2
	hdisk31	CUSTOMER	cust5	std	472	4.2
	hdisk32	CUSTOMER	cust6	std	472	4.2
	hdisk33	CUSTOMER	cust7	std	472	4.2
	hdisk34	CUSTOMER	cust8	std	472	4.2
	hdisk35	STOCKS	stk5	std	368	4.2
	hdisk36	STOCKS	stk6	std	368	4.2
	hdisk37	STOCKS	stk7	std	368	4.2
	hdisk38	STOCKS	stk8	std	368	4.2
ascsi4	hdisk40	LOG	log4	std	1800	4.2
	hdisk41	CUSTOMERIDX	icust1	std	384	4.2
	hdisk42	CUSTOMER	cust9	std	472	4.2
	hdisk43	CUSTOMER	cust10	std	472	4.2
	hdisk44	CUSTOMER	cust11	std	472	4.2
	hdisk45	CUSTOMER	cust12	std	472	4.2
	hdisk46	STOCKS	stk9	std	368	4.2
	hdisk47	STOCKS	stk10	std	368	4.2
	hdisk48	STOCKS	stk11	std	368	4.2
	hdisk49	STOCKS	stk12	std	368	4.2
	hdisk50	STOCKS	stk13	std	368	4.2
	hdisk51	STOCKS	stk14	std	368	4.2
lsa0	hdisk52	STOCKS	stk15	std	368	4.2
	hdisk54	STOCKS	stk16	std	368	4.2
	hdisk55	STOCKS	stk17	std	368	4.2
ascsi5	hdisk56	CAHISTORDER	cahiordx	strip8	384	4.2
	hdisk57	CAHISTORDER	cahiordx	strip8	384	4.2
	hdisk58	CUSTOMER	cust13	std	472	4.2
	hdisk59	CUSTOMER	cust14	std	472	4.2
	hdisk60	CUSTOMER	cust15	std	472	4.2
	hdisk61	STOCKS	stk18	std	368	4.2
	hdisk62	CUSTOMERIDX	icust5	std	384	4.2
ascsi6	hdisk63	CAHISTORDER	cahiordx	strip8	384	4.2
	hdisk64	CAHISTORDER	cahiordx	strip8	384	4.2
	hdisk65	CUSTOMER	cust14	std	472	4.2
	hdisk66	CUSTOMER	cust15	std	472	4.2
	hdisk67	CUSTOMER	cust15	std	472	4.2
	hdisk68	CUSTOMERIDX	icust2	std	384	4.2
ssa0	hdisk70	LOG MIRROR	log1m	mirror	1800	4.5
	hdisk71	LOG MIRROR	log2m	mirror	1800	4.5
	hdisk72	LOG MIRROR	log3m	mirror	1800	4.5
	hdisk73	LOG MIRROR	log4m	mirror	1800	4.5
	hdisk74	CUSTOMER	cust19	std	472	4.5
	hdisk75	CUSTOMER	cust20	std	472	4.5
	hdisk76	CUSTOMER	cust21	std	472	4.5
	hdisk77	CUSTOMER	cust22	std	472	4.5
	hdisk78	STOCKS	stk19	std	368	4.5
	hdisk79	STOCKS	stk20	std	368	4.5
	hdisk80	STOCKS	stk21	std	368	4.5
	hdisk81	STOCKS	stk22	std	368	4.5
	hdisk82	STOCKS	stk23	std	368	4.5
	hdisk83	STOCKS	stk24	std	368	4.5

CONTROLLER	DISK	TABLE NAME	FILENAME	MODE	Used (MB)	Capacity (GB)
(ssa0)	hdisk84	STOCKS	stk25	std	368	4.5
	hdisk85	CUSTOMERIDX	icust3	std	384	4.5
	hdisk86	STOCKS	stk26	std	368	4.5
	hdisk87	STOCKS	stk27	std	368	4.5
	hdisk88	STOCKS	stk28	std	368	4.5
	hdisk89	STOCKS	stk29	std	368	4.5
	hdisk90	STOCKS	stk30	std	368	4.5
	hdisk91	STOCKS	stk31	std	368	4.5
	hdisk92	STOCKS	stk32	std	368	4.5
	hdisk93	CUSTOMER	cust23	std	472	4.5
	hdisk94	CUSTOMER	cust24	std	472	4.5
	hdisk95	CUSTOMER	cust25	std	472	4.5
	hdisk96	CUSTOMER	cust26	std	472	4.5
	hdisk97	STOCKS	stk33	std	368	4.5
	hdisk98	STOCKS	stk34	std	368	4.5
	hdisk99	STOCKS	stk35	std	368	4.5
	hdisk100	STOCKS	stk36	std	368	4.5
	hdisk101	STOCKS	stk37	std	368	4.5
	hdisk102	STOCKS	stk38	std	368	4.5
	hdisk103	STOCKS	stk39	std	368	4.5
	hdisk104	STOCKS	stk40	std	368	4.5
	hdisk105	MASTER	madefnox	strip4	376	4.5
	hdisk106	MASTER	madefnox	strip4	376	4.5
	hdisk107	MASTER	madefnox	strip4	376	4.5
	hdisk108	MASTER	madefnox	strip4	376	4.5
	hdisk109	STOCKS	stk41	std	368	4.5
	hdisk110	STOCKS	stk42	std	368	4.5
	hdsik111	STOCKS	stk43	std	368	4.5
	hdisk112	STOCKS	stk44	std	368	4.5
	hdisk113	STOCKS	stk45	std	368	4.5
	hdisk114	STOCKS	stk46	std	368	4.5
	hdisk115	STOCKS	stk47	std	368	4.5
	hdisk116	STOCKS	stk48	std	368	4.5
	hdisk117	CUSTOMER	cust27	std	472	4.5
	hdisk118	CUSTOMER	cust28	std	472	4.5
	hdisk119	CUSTOMER	cust29	std	472	4.5
	hdisk120	CUSTOMER	cust30	std	472	4.5
	hdisk121	STOCKS	stk49	std	368	4.5
	hdisk122	STOCKS	stk50	std	368	4.5
	hdisk123	STOCKS	stk51	std	368	4.5
	hdisk124	STOCKS	stk52	std	368	4.5
	hdisk125	STOCKS	stk53	std	368	4.5
	hdisk126	STOCKS	stk54	std	368	4.5
	hdisk127	STOCKS	stk55	std	368	4.5
	hdisk128	STOCKS	stk56	std	368	4.5
	hdisk129	STOCKS	stk57	std	368	4.5
	hdisk130	STOCKS	stk58	std	368	4.5
	hdisk131	STOCKS	stk59	std	368	4.5
	hdisk132	STOCKS	stk60	std	368	4.5
	hdisk133	CUSTOMERIDX	icust4	std	384	4.5

5.3 Data Model and database interface

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.*

The Database used for this testing was Sybase SQL server 11.0.3 from Sybase Inc. Sybase SQL server 11.0.3 is a relational DBMS.

5.4 DBMS partitions and 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 computations along with proof that the database is configured to sustain 8 hours of growth for the dynamic tables (Order, Order-Line, and History) must be disclosed (see Clause 4.2.3).

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

6. Clause 5 : Performance Metrics and Response Time Related Items

6.1 Measured Throughput (tpmC)

Measured tpmC must be reported

The measured tpmC was **7307.67 tpmC**.

6.2 Response Times

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

Figure 5 Response Time Data : ESCALA D404/8

Transaction	Average	90th Percentile	Maximum
New Order	0.65	1.10	5.19
Payment	0.59	1.00	12.08
Order Status	0.72	1.07	3.77
Delivery (interactive)	0.17	0.20	1.12
Delivery (deferred)	1.64	-	-
Stock Level	7.10	15.24	40.01
menu	0.15	-	2.55

6.3 keying and Think Times

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

Figure 6 Keying times /think times : ESCALA D404/8

Transaction	Min	Average	Max
New Order	18.00/0.01	18.08/12.21	19.15/122.04
Payment	3.00/0.01	3.04/12.24	4.12/122.03
Order Statu	2.00/0.01	2.04/10.61	2.81/100.56
Delivery	2.00/0.01	2.03/5.10	2.96/51.02
Stock Level	2.00/0.01	2.03/5.10	2.96/51.14

6.4 Frequency Distribution of Response Times

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

Figure 7 New Order Response Times distribution ESCALA D404/8

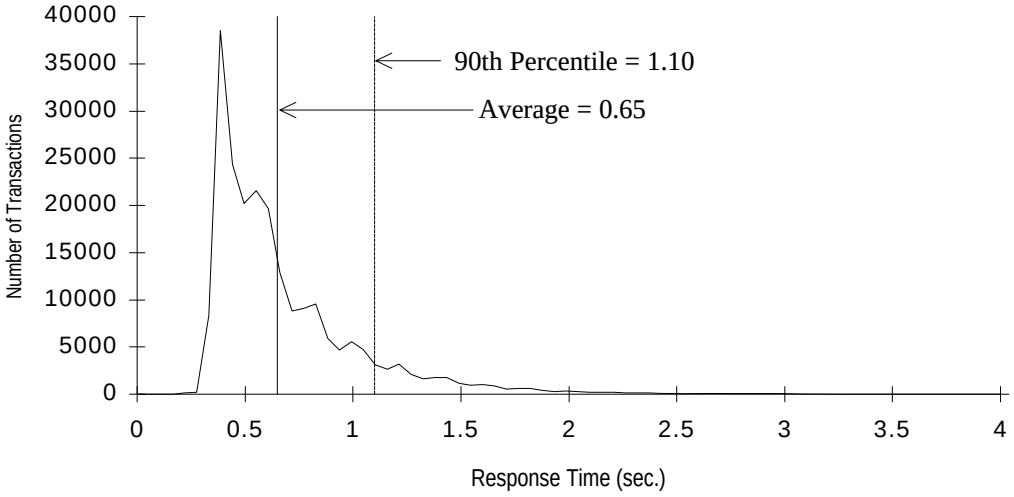


Figure 8 Payment Response Time distribution ESCALA D404/8

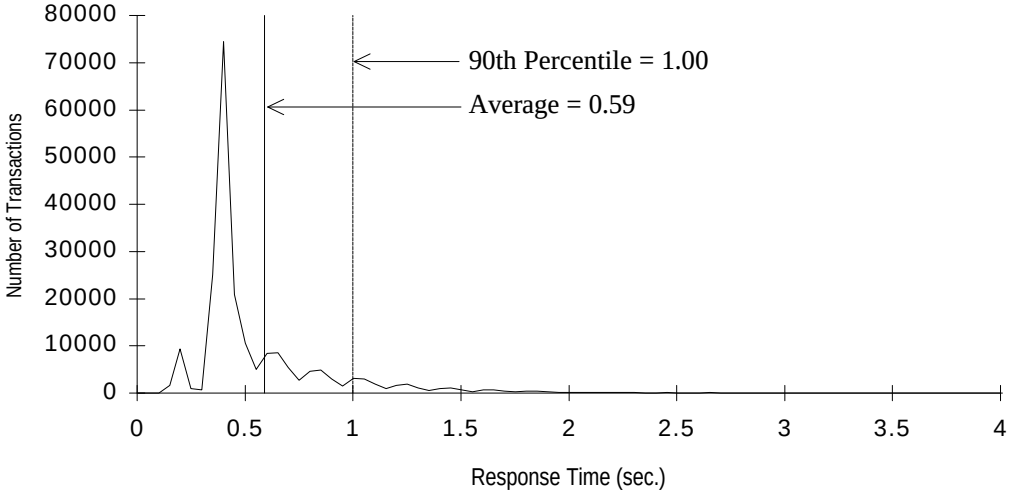


Figure 9 Order Status Response Time distribution ESCALA D404/8

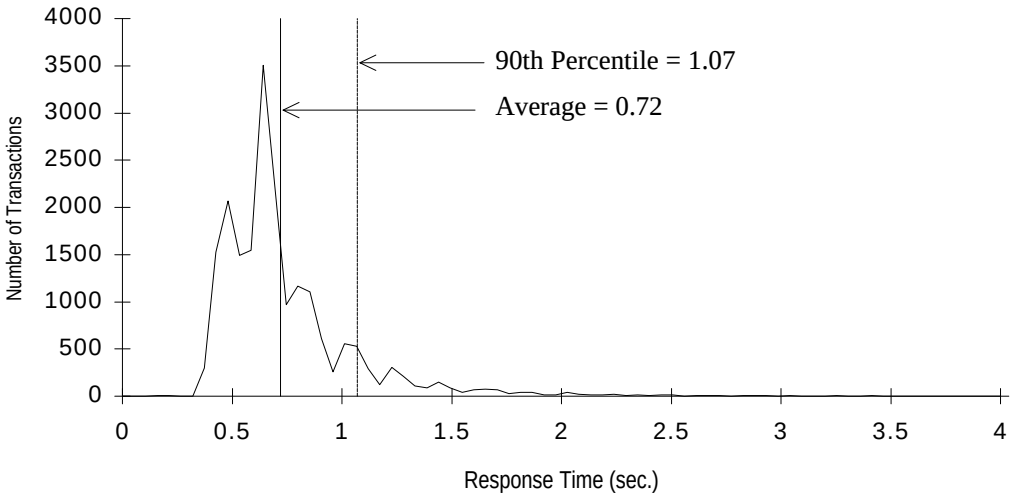


Figure 10 Delivery (interactive) Response Time distribution ESCALA D404/8

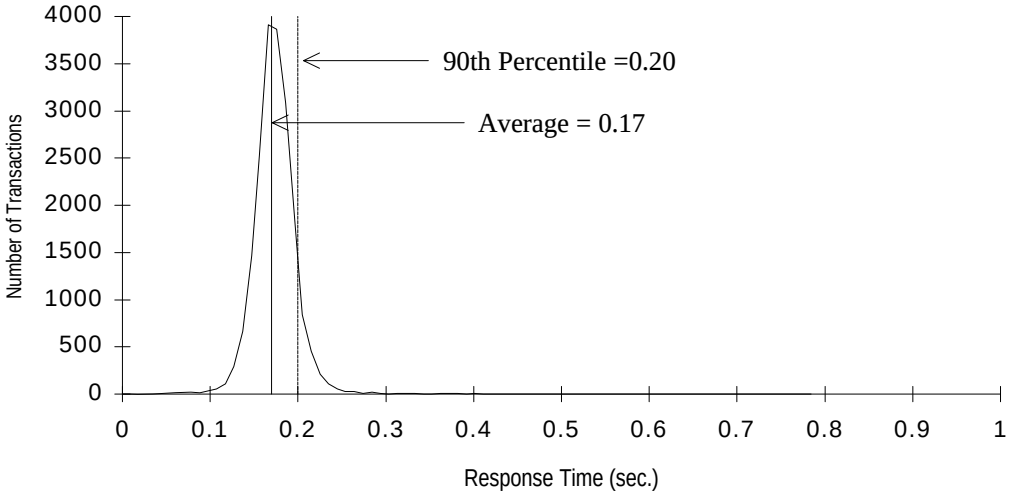
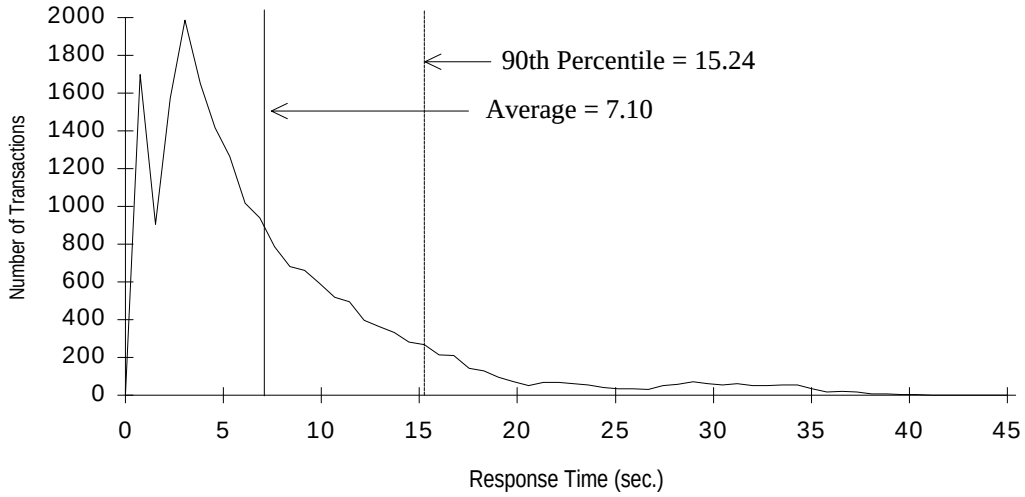


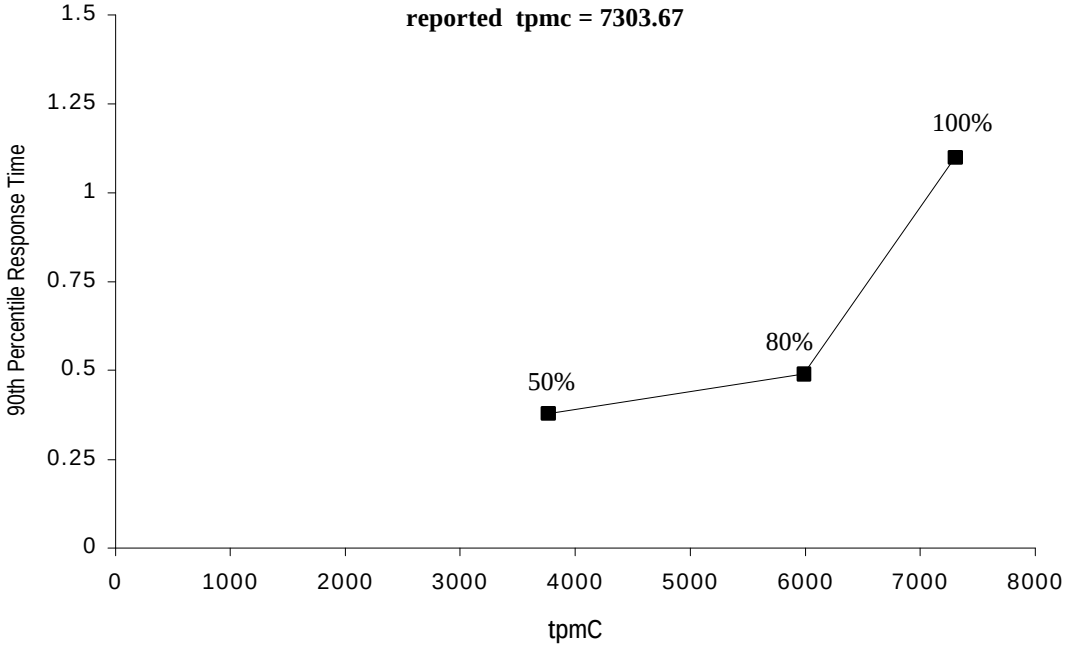
Figure 11 Stock-Level Response Time distribution ESCALA D404/8



6.5 Performance Curve for 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

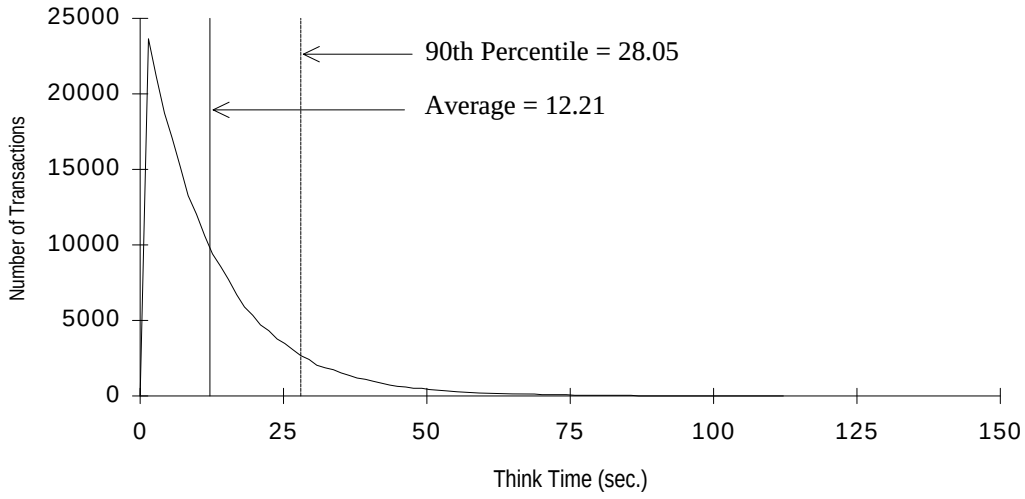
Figure 12 Response Time versus Throughput (tpm-C) ESCALA D404/8



6.6 Frequency Distribution of Think Times

Think time frequency distribution curves (see Clause 5.6.3) must be reported for New Order transaction.

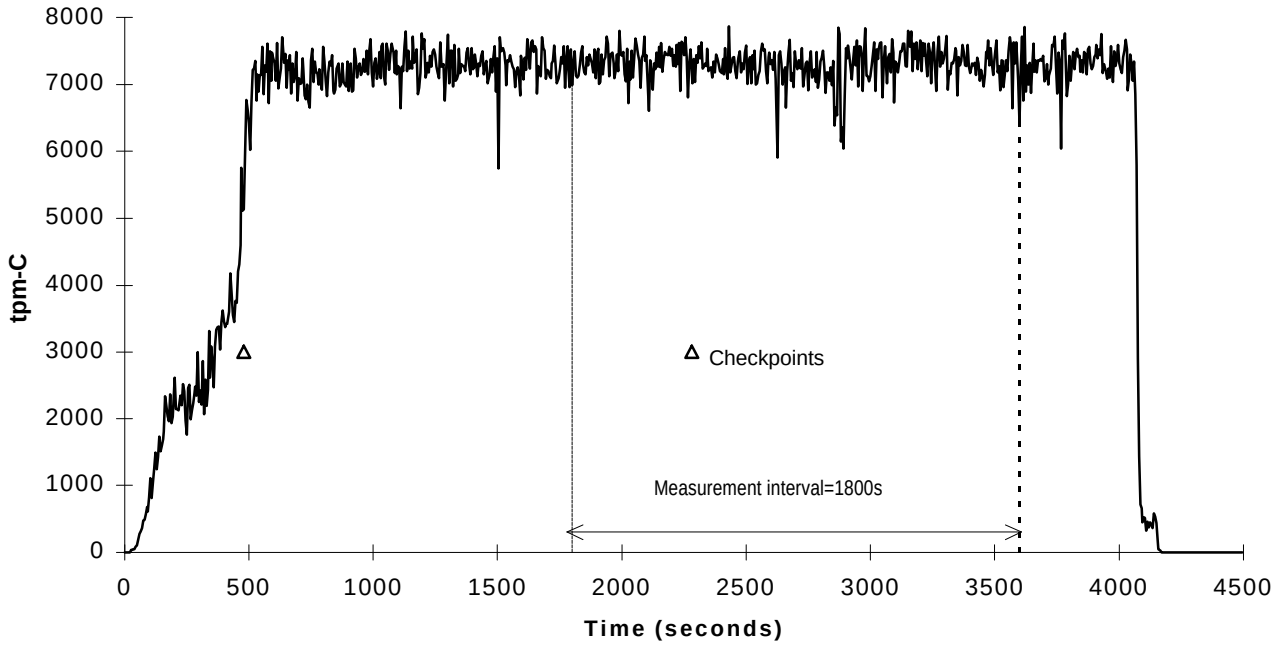
Figure 13 New-Order Think Time distribution ESCALA D404/8



6.7 Throughput Vs Elapsed time : New-Order Transaction

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

Figure 14 Throughput versus Elapsed time ESCALA D404/8



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.

All the emulated users were allowed to logon and start doing transactions. The RTE began actual time-stamping after several minutes of ramp up. Refer to the Numerical Quantities pages for the ramp up time. Figure 14, Throughput versus elapsed time, show that the system was in steady state at the beginning of the measurement interval.

6.9 Work performed during Steady State

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

6.9.1 Transaction Flow

For each of the TPC Benchmark™C transaction types, the following steps are executed:
Tuxedo System/T was used as a transaction manager (TM). Each transaction was divided into two programs. The front end program handled all screen I/O while the back end program handled all database operations. Both the front end and back end programs ran on the client system. The front end program communicates with the back end program through Tuxedo System/T messages. The back end program communicates with the Server system over Ethernet using Sybase Open Client DB-Library/C calls. Besides calling Tuxedo System/T functions for user connection and message communication, all other functions are transparent to the application code. Tuxedo System/T routes the transaction and balances the load according to the options defined in the Tuxedo System/T configuration file listed in appendix B.2. The transaction flow is described below.

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

6.9.2 DataBase Transaction

All database operations are performed by the TPC-C back end program. The process is described below:

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

Sybase SQL Server proceeds to update the database as follws:

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

6.9.3 Checkpoints

A checkpoint is the process of writing all modified data pages to disk. The TPC-C Benchmark was setup to automatically checkpoint every 30 minutes. One checkpoint occurs during the rampup period, with another ocurring during the mesurement interval.

6.10 Reproducibility Method

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

The test was repeated in the same system configuration, and the throughput achieved 7298.40 tpmC.

6.11 Measurement Interval

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

A thirty minute Measurement Interval was used. Further, the measurement interval is a multiple of the checkpoint interval, and the checkpoints fall outside the protected zones of either edge of the measurement interval (as required by Clause 5.5.2.2). this demonstrates that a different measurement interval over the eight hour period would yield similar throughput results.

7. Clause 6 : SUT, Driver, and Communication Definition. Related Items

7.1 RTE Inputs

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

The RTE was the IBM proprietary RTE used in IBM RISC System /6000 C30 and J30 TPC Benchmark™C measurement (Feb 17, 1995).
Appendix D contains all RTE scripts.

7.2 Driver functionality and Performance

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

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

As configured for this test, the driver software emulates the traffic that would be observed from the PCs connected by Ethernet to the Front end clients.

7.3 Functional diagrams and Details of driver system

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

The diagrams in section 1.4 show the tested and priced benchmark configuration.

7.4 Network configurations and driver system

The network configuration of both the tested service and the proposed (target) services which are being represented and a thorough explanation of exactly which parts of the proposed configuration are being replaced with the Driver System must be disclosed. (see Clause 6.6.4).
The diagrams in section 1.4 show the tested and priced network configuration.
Each RTE emulated 1010 PCs connecting over a single Ethernet network 10 megabits per second. In the priced configuration, the PCs are split over two networks, each connecting to a separate Ethernet board in the client machine (506+504).
In the tested configuration, all the drivers are connected by a FDDI network.
A Lanplex 2500 is used to switch the FDDI network into 6 Independant Ethernet networks (1 for each client)
The front-end clients are connected by a single Ethernet 10MBps to the Server.

7.5 Network Bandwidth

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

Ethernet Local Area Network (LAN) with a bandwidth of 10 megabits per seconds are used in the tested/priced configurations.
The FDDI ring between the RTE machines has a bandwith of 100 Mbps.

7.6 Operator Intervention

The need for operator intervention must be disclosed.

The configurations reported do not require any operator intervention to sustain the reported throughput during the eight hour period.

8. Clause 7 : Pricing Related Items

A detailed list of hardware and software used in the priced system must be reported. Each separately orderable 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.
The total 5-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

8.1 System pricing

The detailed list of hardware and programs for the priced configuration is listed in the pricing sheet (refer to the executive summary statement). Prices for all Bull S.A. products are US list prices. Each priced configuration consists of an integrated system package, additional components and third party components.

The Escala D404/8 Packaged system referenced by the MI CPKG145U0000includes :

4 dual CPU model Power PC 604e @150Mhz-4 MB L2 cache/CPU
1024MB of memory
12x4.2 GB disk
2x SCSI-2 F/W DE Ext Disk Adapter with Ethernet port on the board
1 CD ROM
1 floppy disk 1.44MB 3”1/2
1 expansion cabinet
2 MCA bus with 13 free MCA slots (15 total)
AIX 4.2 High End Server Operating system for unlimited users

The Escala M204 4way Packaged system referenced by the MI CPKG146U0000includes :

2 dual CPU model Power PC 604 @112Mhz-1 MB L2 cache/CPU
512 MB of memory
2x2.1 GB disk
1x SCSI-2 F/W SE Ext Disk Adapter with Ethernet port on the board
1 CD ROM
1 floppy disk 1.44MB 3”1/2
1 System console BQ306
AIX 4.2 Mini Tower Server Operating system for unlimited users

8.2 Support

- Bull S.A. support

The five years support pricing for Bull S.A.consists of one year warranty included in the system package price and four years support price.

- SYBASE support

  Sybase standard Technical support includes Product updates, regular technical publication, unlimited telephone service. Annual Support Pricing is 16% of total price.

- TUXEDO support

  Annual Support Pricing is 15% of total price.

- HUB support

Netlux offers five year of warranty included in the priced configuration.
An additional 10% of the required terminal servers and port cards were added in the configuration to provide the required four hour repair for hardware components

8.3 Discounts

- Bull S.A. discount

A Bull S.A. 15% dollar volume discount on Hardware and Software is applicable to Hardware configurations between $500,000 and $1,000,000.

- Sybase Inc. discount

A 5% discount is included in the product and support prices.

8.4 Availability Status

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.

The Escala D404, MI CPKG145U0000 is available on November 30, 1996.
The version of Sybase SQL sever 11 used in measurement will be available in January, 1997 as SQL server 11.0.3.
All other software and hardware components used in the tested and priced systems are available now.

8.5 Throughput Price/Performance and availability

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

System	tpmC	5-year System cost	$/tpmC	Availability Date
Escala D404/8	**7307.67**	**$1,429,252**	**195.68**	**January, 1997**

8.6 Space Calculations

Appendix E contains the 180 day space calculations.

9. Clause 9 : Auditor Related Items

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 on the ESCALA D404/8
was audited by Francois Raab of

Information Paradigm
115 North Wahsatch Ave Suite 107
Colorado Springs CO 80903

9.1 Auditor’s attestation letter

auditors attestation letter (Page 1)

auditors attestation letter (Page 2)

10. Price quotation

SYBASE

BULL SA
1, Rue de Provence
BP 208
38432 ECHIROLLES Cédex

To Mr Jean-François LEMERRE

Lyon, October, 25

Dear Sir,

We are pleased to send you this proposal regarding your application for a SQL Server with unlimited users on a eight CPUs machine.

Deployment Pricing :	$ 150,000
Development Pricing :	$ 9,600
Open Client Pricing :	$ 785
Total Pricing :	$ 160,395
5% Discount	$ 8,020
Total Net Pricing :	**$ 152,375**
Support Pricing (16% * $160,395 * 5 years) :	**$ 128,316**
Total Price :	**$ 280 691**

Sincerely,

Olivier AUDOUZE
Sales Representative
SYBASE Inc.

HUB quotation (Page 1)

BEA quotation (Page 1)

Appendix A: Client / Server Source

This appendix contains the source and makefiles for all client and server programs

A.1 Client Front-End

main.c

```
/*****************************************************************************/
/*      File: main.c                                                         */
/*      program description:                                                 */
/*      TPC-C benchmark of Sybase. this program comprises the                */
/*      TPCC benchmark transactions.                                         */
/*****************************************************************************/
#define AUDIT
#define DEBUG
/*-----------------------------------------------------------*/
/* The makefile will include these modules                   */
/*-----------------------------------------------------------*/
#include <stdio.h>
#include <sys/select.h>
#include <sybfront.h>
#include "../incl/tpcc.h"
#include "../incl/tuxclient.h"
#include "./termio.h"

#define         USER_END_STR    "User End"
#define USER_END_LG     strlen(USER_END_STR)
/*-----------------------------------------------------------*/
/*              main                                         */
/*-----------------------------------------------------------*/
main (argc, argv)
    int  argc;
    char *argv[];
   {
    char          buf[5],
                  pbuf[15];
    int           key,
                  p;
connect_to_TM();
    alloc_buf_for_TM();
    load_scr_bufs(argc,argv);
    set_up_port();
    clear_menu();

    for (;;)
         {
          int rc = 0;

          rc = read(fdi, buf, 2);

        switch (buf[0])
            {
           case '1':
                 clear_neword(neword_ptr);
             key = read_neword(neword_ptr);
               if (key != CNTRL_C_KEY)
                   {
                    key = submit_TM_neword();
                    if (key != -1)
                   write_neword(neword_ptr);
                    else
                    {
                     p      = 0;
                     p      += rowcol(2,1,&pbuf[p]);
                       pbuf[p] = (char)NULL;
                       write(fdo, pbuf, p);
                    }
                  }
          break;

case '2':
   clear_payment(payment_ptr);
   key = read_payment(payment_ptr);
       if (key != CNTRL_C_KEY)
          {
           key = submit_TM_payment();
           if (key != -1)
          write_payment(payment_ptr);
           else
             {
              p      = 0;
              p      += rowcol(2,1,&pbuf[p]);
              pbuf[p] = (char)NULL;
              write(fdo, pbuf, p);
             }
          }
   break;

case '3':
   clear_ordstat(ordstat_ptr);
   key = read_ordstat(ordstat_ptr);
       if (key != CNTRL_C_KEY)
          {
           key = submit_TM_ordstat();
           if (key != -1)
          write_ordstat(ordstat_ptr);
           else
             {
              p      = 0;
                p      += rowcol(2,1,&pbuf[p]);
                pbuf[p] = (char)NULL;
                write(fdo, pbuf, p);
             }
          }
   break;

case '4':
   clear_delivery(delivery_ptr);
   key = read_delivery(delivery_ptr);
       if (key != CNTRL_C_KEY)
          {
           key = submit_TM_delivery();
           if (key != -1)
          write_delivery(delivery_ptr);
           else
             {
              p = 0;
              p += rowcol(2,1,&pbuf[p]);
              pbuf[p] = (char)NULL;
              write(fdo, pbuf, p);
             }
```

```
                }
            break;

        case '5':
            clear_stocklev(stocklev_ptr);
            key = read_stocklev(stocklev_ptr);
                if (key != CNTRL_C_KEY)
                  {
                   key = submit_TM_stocklev();
                   if (key != -1)
                  write_stocklev(stocklev_ptr);
                   else
                     {
                      p       = 0;
                      p       += rowcol(2,1,&pbuf[p]);
                      pbuf[p] = (char)NULL;
                      write(fdo, pbuf, p);
                     }
                  }
            break;

        case '9':
                disconnect_from_TM();
                write(fdo, USER_END_STR, USER_END_LG);
            end_screen();
            exit(0);

        default:
            break;
          }             /* end switch statement */
      }                 /* end for loop */
  }                     /* end of program */
```

tpcc.h

```
#define TPCCH
/*-------------------------------------------------------*/
/*            Global numbers, constants,...              */
/*-------------------------------------------------------*/
short    ware_num;
short    dist_num;

#define INVALID_ITEM            100
#define TRAN_OK                 0
#define REMOTE_WAREHOUSE 17

#define FORM_DATE               1
#define FORM_DATETIME           2

#define I_NAME_LEN              24
#define DATETIME_LEN            19
#define CREDIT_LEN              2
#define LAST_LEN            17
#define MAX_OL                  15
#define DELIVDATA_LEN           20
/*-------------------------------------------------------*/
/*            transaction structures                     */
/*-------------------------------------------------------*/
struct neword_struct
  {
    DBSMALLINT  s_W_ID;
    DBTINYINT   s_D_ID;
    DBINT       s_C_ID;
    char        s_C_LAST[17];
    char        s_C_CREDIT[3];
    DBREAL      s_C_DISCOUNT;
    DBTINYINT   s_O_OL_CNT;
    DBINT       s_O_ID;
    char        s_O_ENTRY_D[20];
    char        s_status_line[25];
    DBFLT8      s_total_amount;
    DBTINYINT   s_all_local;
    short       s_transtatus;
    DBREAL      s_W_TAX;
    DBREAL      s_D_TAX;
    struct items_struct
      {
       DBSMALLINT      s_OL_SUPPLY_W_ID;
       DBINT           s_OL_I_ID;
       char            s_I_NAME[25];
       DBTINYINT       s_OL_QUANTITY;
       DBSMALLINT      s_S_QUANTITY;
       char            s_brand_generic[2];
       DBFLT8          s_I_PRICE;
       DBFLT8          s_OL_AMOUNT;
      } item[15];
  };

struct delivery_struct
  {
    DBSMALLINT  s_W_ID;
    DBSMALLINT  s_O_CARRIER_ID;
    double      s_queued_time;
    char        s_exec_status[50];
  } *delivery;

struct ordstat_struct
  {
    DBSMALLINT  s_W_ID;
    DBTINYINT   s_D_ID;
    DBINT       s_C_ID;
    char        s_C_FIRST[17];
    char        s_C_MIDDLE[3];
    char        s_C_LAST[17];
    DBFLT8      s_C_BALANCE;
    DBINT       s_O_ID;
    char        s_O_ENTRY_D[20];
    DBSMALLINT  s_O_CARRIER_ID;
    short       s_ol_cnt;
    short       s_transtatus;
    struct oitems_struct
      {
       DBSMALLINT      s_OL_SUPPLY_W_ID;
       DBINT           s_OL_I_ID;
```

```
    DBTINYINT       s_OL_QUANTITY;
    DBFLT8          s_OL_AMOUNT;
    char            s_OL_DELIVERY_D[20];
   } item[15];
  };
struct payment_struct
  {
   DBSMALLINT  s_W_ID;
   DBTINYINT   s_D_ID;
   DBINT       s_C_ID;
   DBTINYINT   s_C_D_ID;
   DBSMALLINT  s_C_W_ID;
   short       s_transtatus;
   DBFLT8      s_H_AMOUNT;
   char        s_H_DATE[20];
   char        s_W_STREET_1[21];
   char        s_W_STREET_2[21];
   char        s_W_CITY[21];
   char        s_W_STATE[3];
   char        s_W_ZIP[10];
   char        s_D_STREET_1[21];
   char        s_D_STREET_2[21];
   char        s_D_CITY[21];
   char        s_D_STATE[3];
   char        s_D_ZIP[10];
   char        s_C_FIRST[17];
   char        s_C_MIDDLE[3];
   char        s_C_LAST[17];
   char        s_C_STREET_1[21];
   char        s_C_STREET_2[21];
   char        s_C_CITY[21];
   char        s_C_STATE[3];
   char        s_C_ZIP[10];
   char        s_C_PHONE[17];
   char        s_C_SINCE[20];
   char        s_C_CREDIT[3];
   DBFLT8      s_C_CREDIT_LIM;
   DBREAL      s_C_DISCOUNT;
   DBFLT8      s_C_BALANCE;
   char        s_C_DATA[201];
  };

struct stocklev_struct
  {
   DBSMALLINT  s_W_ID;
   DBTINYINT   s_D_ID;
   DBSMALLINT  s_threshold;
   DBINT       s_low_stock;
   short       s_transtatus;
  };

#define NEWORD_LEN      sizeof(struct neword_struct)
#define PAYMENT_LEN     sizeof(struct payment_struct)
#define ORDSTAT_LEN     sizeof(struct ordstat_struct)
#define DELIVERY_LEN    sizeof(struct delivery_struct)
#define STOCKLEV_LEN    sizeof(struct stocklev_struct)

#define RandVal         lrand48
#define RandSeed        srand48
```

termio.h

```
/*************************************************************************/
/*      File: termio.h                                                   */
/*      program description:                                             */
/*      This module contains all the screen io subroutines for the       */
/*      TPC-C benchmark.                                                 */
/*************************************************************************/
#define DEBUG
#define ACID_TEST
/*-------------------------------------------------*/
/*        terminal types                           */
/*-------------------------------------------------*/
/* mettre HFT pour test sur une vt100 ou un xterm */
/* mettre WYSE50_TERM pour le RTE d'IBM */
/*#define HFT*/
#define WYSE50_TERM
/*-------------------------------------------------*/
/*        include files                            */
/*-------------------------------------------------*/
#ifndef TPCCH
#include "../incl/tpcc.h"
#endif

#include <stdio.h>
#include <termio.h>
#include <termios.h>
#include <sys/str_tty.h>
#include <string.h>
#include <time.h>       /* functions: time() */

#define FORM_DATE       1
#define FORM_DATETIME                   2

#define ERRMSG_ROW              2

#define INVALID_ITEM            100
#define FATAL_SQL_ERROR             -1

/* minimum number of characters to leave screen */
#define SCR_READ_NUM                    100
/* minimum time to leave screen, 10=1sec */
#define SCR_READ_TIMEOUT    2

#define CIRBUF_SIZE             1000
/*-------------------------------------------------------*/
/*              Keyboard Keys                            */
/*-------------------------------------------------------*/
#define ENTER_KEY       10
#define TAB_KEY         9
#define BACK_TAB_KEY    2
#define BACKSPACE_KEY       8
```

```
#define CNTRL_C_KEY     3
/*-------------------------------------------------------*/
/*              Video Atributes                          */
/*-------------------------------------------------------*/
#define SPACE                   ' '
#define UNDERL                  '_'

#ifdef HFT
#define CLEAR_SCREEN_1          " [1H [J"
#define CLEAR_SCREEN_3          " [3H [J"
#define BEEP_SOUND              " "
#endif

/* wyse, televideo, fredom one, ... */
#ifdef ASCII_TERM
#define CLEAR_SCREEN_1          " =   Y"
#define CLEAR_SCREEN_3          " =!  Y"
#define BEEP_SOUND              " "
#endif

#ifdef WYSE50_TERM
#define CLEAR_SCREEN_1          "\033=  \033Y"
#define CLEAR_SCREEN_3          "\033=\" \033Y"
#define BEEP_SOUND              " "
#endif
/*-------------------------------------------------------*/
/*              Terminal Buffers                         */
/*-------------------------------------------------------*/
static char     clear_menu_buf[85];
static char     clear_neword_buf[622];
static char     clear_payment_buf[270];
static char     clear_ordstat_buf[227];
static char     clear_delivery_buf[77];
static char     clear_stocklev_buf[95];
/*-------------------------------------------------------*/
/*              Buffers Size                             */
/*-------------------------------------------------------*/
static int      clear_menu_size;
static int      clear_neword_size;
static int      clear_payment_size;
static int      clear_ordstat_size;
static int      clear_delivery_size;
static int      clear_stocklev_size;
/*-------------------------------------------------------*/
/*              screen routines                          */
/*-------------------------------------------------------*/
static int      clear_menu();

static int      clear_neword();
static int      read_neword();
static int      write_neword();

static int      clear_payment();
static int      read_payment();
static int      write_payment();

static int      clear_ordstat();
static int      read_ordstat();
static int      write_ordstat();

static int      clear_delivery();
static int      read_delivery();
static int      write_delivery();

static int      clear_stocklev();
static int      read_stocklev();
static int      write_stocklev();

double          get_time();
/*-------------------------------------------------------*/
/*              tty definitions                          */
/*-------------------------------------------------------*/
static struct termios in_tty,
                      out_tty,
                      savetty;
static int            fdi = 0,          /* INPUT */
                      fdo = 1;          /* OUTPUT */

set_up_port()
 {
  struct termios fastport;
  int               remote_on = 1;
 /*-----------------------------------------------------------------------*/
  /*       INPUT  fasport configuration.                                   */
  /*-----------------------------------------------------------------------*/
 /* Get the input termios config. */
  ioctl(fdi, TCGETS, &in_tty);

  /* Save the original input termios */
  memcpy(&savetty, &in_tty, sizeof(struct termios));

  /* Copy the original input termios in fastport termios */
  fastport              = in_tty;
  /* c_lflag used by the line discipline to control terminal functions. */
  /* ===> Here in our configuration we want nothing special.           */
  fastport.c_lflag     = 0;
  /* c_oflag described the system treatment of output.                 */
  /* ===> Here in our configuration we want nothing special.           */
  fastport.c_oflag     = 0;
  /* c_iflag described the basic terminal input control.               */
  /* c_cc[NCC] described the special control characters.               */
  /* ===> VMIN used for minimun number of chars to leave screen.       */
  fastport.c_cc[VMIN]  = SCR_READ_NUM;
  /* ===> VTIME used for minimum time to leave screen, 10=1sec.        */
  fastport.c_cc[VTIME] = SCR_READ_TIMEOUT;

  /* Set the new input termios config and flush it.*/
  ioctl(fdi, TCSETSW, &fastport);
  ioctl(fdi, TCFLSH, 2);
  /*-----------------------------------------------------------------------*/
  /*       OUTPUT fasport configuration.                                   */
  /*-----------------------------------------------------------------------*/
 /* Get the output termios config. */
  ioctl(fdo, TCGETS, &out_tty);

  /* Set the new output termios config and flush it.*/
  ioctl(fdo, TCSETSW, &fastport);
  ioctl(fdo, TCFLSH, 2);
```

```
 }
/*---------------------------------------------------------------*/
/*              load screen buffers                              */
/*---------------------------------------------------------------*/
load_scr_bufs (argc,argv)
        int     argc;
        char    *argv[];
{
if (argc < 3) {
        printf("*** error: \n");
        printf("        - warehouse id must be supplied.\n");
        printf("        - district id must be supplied.\n");
        exit(1);
        }
ware_num = atoi(argv[1]);
dist_num = atoi(argv[2]);

strcpy(clear_menu_buf, CLEAR_SCREEN_1);
rowcol(1,1, &clear_menu_buf[strlen(clear_menu_buf)]);
strcat(clear_menu_buf, "(1)New-Order (2)Payment (3)Order-Status (4)");
strcat(clear_menu_buf, "Delivery (5)Stock-Level (9)Exit\n");

clear_menu_size = strlen(clear_menu_buf);

strcpy(clear_neword_buf, CLEAR_SCREEN_3);
rowcol(3,36, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "New Order");
rowcol(4,1, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Warehouse: ");
sprintf(&clear_neword_buf[strlen(clear_neword_buf)], "%d", ware_num);
rowcol(4,19, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "District: __");
rowcol(4,55, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Date:");
rowcol(5,1, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Customer:  ____");
rowcol(5,19, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Name:");
rowcol(5,44, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Credit:");
rowcol(5,57, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Disc:");
rowcol(6,1, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Order Number:");
rowcol(6,25, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Number of Lines:");
rowcol(6,52, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "W_tax:");
rowcol(6,67, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "D_tax:");
rowcol(8,2, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Supp_W  Item_Id  Item Name");
rowcol(8,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Qty  Stock  B/G  Price    Amount");
rowcol(9,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(9,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(9,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(10,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(10,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(10,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(11,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(11,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(11,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(12,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(12,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(12,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(13,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(13,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(13,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(14,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(14,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(14,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(15,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(15,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(15,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(16,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(16,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(16,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(17,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(17,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(17,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(18,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(18,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(18,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(19,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(19,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
```

```
rowcol(19,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(20,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(20,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(20,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(21,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(21,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(21,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(22,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(22,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(22,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(23,3, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "____");
rowcol(23,10, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "______");
rowcol(23,45, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "__");
rowcol(24,1, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Execution Status:");
rowcol(24,62, &clear_neword_buf[strlen(clear_neword_buf)]);
strcat(clear_neword_buf, "Total:");

clear_neword_size = strlen(clear_neword_buf);

strcpy(clear_payment_buf, CLEAR_SCREEN_3);
rowcol(3,38, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Payment");
rowcol(4,1, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Date:");
rowcol(6,1, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Warehouse: ");
sprintf(&clear_payment_buf[strlen(clear_payment_buf)], "%d", ware_num);
rowcol(6,42, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "District: __");
rowcol(11,1, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Customer: ____");
rowcol(11,17, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Cust-Warehouse: ____");
rowcol(11,39, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Cust-District: __");
rowcol(12,1, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Name:");
rowcol(12,29, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "____________________");
rowcol(12,50, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Since:");
rowcol(13,50, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Credit:");
rowcol(14,50, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "%Disc:");
rowcol(15,50, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Phone:");
rowcol(17,1, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Amount Paid:");
rowcol(17,24, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "________");
rowcol(17,38, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "New Cust-Balance");
rowcol(18,1, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Credit Limit:");
rowcol(20,1, &clear_payment_buf[strlen(clear_payment_buf)]);
strcat(clear_payment_buf, "Cust-Data:");

clear_payment_size = strlen(clear_payment_buf);

strcpy(clear_ordstat_buf, CLEAR_SCREEN_3);
rowcol(3,35, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Order-Status");
rowcol(4,1, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Warehouse: ");
sprintf(&clear_ordstat_buf[strlen(clear_ordstat_buf)], "%d", ware_num);
rowcol(4,19, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "District: __");
rowcol(5,1, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Customer: ____   Name:");
rowcol(5,44, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "____________________");
rowcol(6,1, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Cust-Balance:");
rowcol(8,1, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Order-Number:");
rowcol(8,26, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Entry-Date:");
rowcol(8,60, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Carrier-Number:");
rowcol(9,1, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Supply-W:");
rowcol(9,14, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Item-Id:");
rowcol(9,25, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Qty");
rowcol(9,33, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Amount");
rowcol(9,45, &clear_ordstat_buf[strlen(clear_ordstat_buf)]);
strcat(clear_ordstat_buf, "Delivery-Date");

clear_ordstat_size = strlen(clear_ordstat_buf);

strcpy(clear_delivery_buf, CLEAR_SCREEN_3);
rowcol(3,38, &clear_delivery_buf[strlen(clear_delivery_buf)]);
strcat(clear_delivery_buf, "Delivery");
rowcol(4,1, &clear_delivery_buf[strlen(clear_delivery_buf)]);
strcat(clear_delivery_buf, "Warehouse: ");
sprintf(&clear_delivery_buf[strlen(clear_delivery_buf)], "%d", ware_num);
rowcol(6,1, &clear_delivery_buf[strlen(clear_delivery_buf)]);
strcat(clear_delivery_buf, "Carrier Number: __");
rowcol(8,1, &clear_delivery_buf[strlen(clear_delivery_buf)]);
strcat(clear_delivery_buf, "Execution Status:");
```

```
clear_delivery_size = strlen(clear_delivery_buf);

strcpy(clear_stocklev_buf, CLEAR_SCREEN_3);
rowcol(3,35, &clear_stocklev_buf[strlen(clear_stocklev_buf)]);
strcat(clear_stocklev_buf, "Stock-Level");
rowcol(4,1, &clear_stocklev_buf[strlen(clear_stocklev_buf)]);
strcat(clear_stocklev_buf, "Warehouse: ");
sprintf(&clear_stocklev_buf[strlen(clear_stocklev_buf)], "%d", ware_num);
rowcol(4,19, &clear_stocklev_buf[strlen(clear_stocklev_buf)]);
strcat(clear_stocklev_buf, "District: ");
sprintf(&clear_stocklev_buf[strlen(clear_stocklev_buf)], "%d", dist_num);
rowcol(6,1, &clear_stocklev_buf[strlen(clear_stocklev_buf)]);
strcat(clear_stocklev_buf, "Stock Level Threshold: __");
rowcol(8,1, &clear_stocklev_buf[strlen(clear_stocklev_buf)]);
strcat(clear_stocklev_buf, "low stock:");

clear_stocklev_size = strlen(clear_stocklev_buf);
}
/*---------------------------------------------------------------*/
/*              rowcol                                           */
/*---------------------------------------------------------------*/
static int rowcol (r,c,str)
        int     r,c;
        char    *str;
{
static int      size;

#ifdef HFT
size = 4;
if (r < 10) size += 1;
else size += 2;
if (c < 10) size += 1;
else size += 2;
sprintf(str, " [%d;%dH", r, c);
return (size);
#endif

#ifdef ASCII_TERM
str[0] = ' ';
str[1] = '=';
str[2] = ' ' + r - 1;
str[3] = ' ' + c - 1;
str[4] = '\0'
return(4);
#endif

#ifdef WYSE50_TERM
str[0] = '\033';
str[1] = '=';
str[2] = ' ' + r - 1 ;
str[3] = ' ' + c - 1;
str[4] = '\0';
return(4);
#endif
}
/*---------------------------------------------------------------*/
/*              clear menu                                       */
/*---------------------------------------------------------------*/
static int clear_menu ()
{
   write(fdo, clear_menu_buf, clear_menu_size);
}
/*---------------------------------------------------------------*/
/*              clear new order screen                           */
/*---------------------------------------------------------------*/
static int clear_neword (neword)
   struct neword_struct *neword;
  {
   neword->s_W_ID = ware_num;
   write(fdo, clear_neword_buf, clear_neword_size);
  }
/*---------------------------------------------------------------*/
/*              clear payment screen                             */
/*---------------------------------------------------------------*/
static int clear_payment (payment)
   struct payment_struct *payment;
  {
   payment->s_W_ID = ware_num;
   write(fdo, clear_payment_buf, clear_payment_size);
  }
/*---------------------------------------------------------------*/
/*              clear ordstat screen                             */
/*---------------------------------------------------------------*/
static int clear_ordstat (ordstat)
   struct ordstat_struct *ordstat;
  {
   ordstat->s_W_ID = ware_num;
   write(fdo, clear_ordstat_buf, clear_ordstat_size);
  }
/*---------------------------------------------------------------*/
/*              clear delivery screen                            */
/*---------------------------------------------------------------*/
static int clear_delivery (delivery)
   struct delivery_struct *delivery;
  {
   delivery->s_W_ID    = ware_num;
   write(fdo, clear_delivery_buf, clear_delivery_size);
  }
/*---------------------------------------------------------------*/
/*              clear stocklev screen                            */
/*---------------------------------------------------------------*/
static int clear_stocklev (stocklev)
   struct stocklev_struct *stocklev;
  {
   stocklev->s_W_ID = ware_num;
   stocklev->s_D_ID = dist_num;
   write(fdo, clear_stocklev_buf, clear_stocklev_size);
  }
/*---------------------------------------------------------------*/
/*              read new order screen                            */
/*---------------------------------------------------------------*/
static int read_neword (neword)
   struct neword_struct *neword;
  {
   struct items_struct         temp[15];
   static int              temp_cnt, i, row, key, num;

   temp_cnt              = 0;
```

```
    neword->s_C_ID              = 0;
    neword->s_D_ID              = 0;
    neword->s_total_amount = 0;
    for (i = 0; i < 15; i++)
       {
          temp[i].s_OL_SUPPLY_W_ID = 0;
        temp[i].s_OL_I_ID       = 0;
        temp[i].s_OL_QUANTITY    = 0;
          neword->item[i].s_OL_SUPPLY_W_ID = 0;
        neword->item[i].s_OL_I_ID       = 0;
        neword->item[i].s_OL_QUANTITY    = 0;
         }
NOF1:
   key = getintfld(neword->s_D_ID, &num, 2, 4, 29); /* district number */
   neword->s_D_ID = num;
   if (key == CNTRL_C_KEY) goto read_neword_exit;
   if (key == BACK_TAB_KEY) {
         i = 14;
         row = 23;
         goto NOF5;
         }
NOF2:
   key = getintfld(neword->s_C_ID, &num, 4, 5, 12); /* customer number */
   neword->s_C_ID = num;
   if (key == CNTRL_C_KEY) goto read_neword_exit;
   if (key == BACK_TAB_KEY) goto NOF1;

   i = 0;
   row = 9;
NOF3:
       key = getintfld(temp[i].s_OL_SUPPLY_W_ID,&num,4,row,3);
       temp[i].s_OL_SUPPLY_W_ID = num;
       if (key == CNTRL_C_KEY) goto read_neword_exit;
       if (key == BACK_TAB_KEY) {
                  if (i == 0) goto NOF2;
                  else {
                           --i;
                           --row;
                           goto NOF5;
                           }
                  }
NOF4:
       key = getintfld(temp[i].s_OL_I_ID,&num,6,row,10);
       temp[i].s_OL_I_ID = num;
       if (key == CNTRL_C_KEY) goto read_neword_exit;
       if (key == BACK_TAB_KEY) goto NOF3;
NOF5:
       key = getintfld(temp[i].s_OL_QUANTITY,&num,2,row,45);
       temp[i].s_OL_QUANTITY = num;
       if (key == BACK_TAB_KEY) goto NOF4;
         if (key == ENTER_KEY || key == CNTRL_C_KEY) goto
read_neword_exit;

   i++;
   row++;
   if (i == 15) goto NOF1;
   else goto NOF3;

read_neword_exit:
          temp_cnt = i + 1;
          if (neword->s_D_ID == 0) {
                  while (neword->s_D_ID == 0) {
                           errmsg("D_ID must be entered.");
                           key = getintfld(neword->s_D_ID, &num, 2, 4, 29);
                           neword->s_D_ID = num;
                           }
                  }
          if (neword->s_C_ID == 0) {
                  while (neword->s_C_ID == 0) {
                           errmsg("C_ID must be entered.");
                           key = getintfld(neword->s_C_ID, &num, 4, 5, 12);
                           neword->s_C_ID = num;
                           }
                  }
          neword->s_all_local = 1;
          row = 9;
          neword->s_O_OL_CNT = 0;
          for (i = 0; i < temp_cnt; i++) {
            if(!((temp[i].s_OL_SUPPLY_W_ID == 0) &&
                  (temp[i].s_OL_I_ID == 0) &&
                  (temp[i].s_OL_QUANTITY == 0))) {
                  while (temp[i].s_OL_SUPPLY_W_ID == 0) {
                    errmsg("OL_SUPPLY_W_ID must be entered.");
                    key=getintfld(temp[i].s_OL_SUPPLY_W_ID,&num,4,row,3);
                    temp[i].s_OL_SUPPLY_W_ID = num;
                    }
                  while (temp[i].s_OL_I_ID == 0) {
                    errmsg("OL_I_ID must be entered.");
                  key=getintfld(temp[i].s_OL_I_ID,&num,6,row,10);
                  temp[i].s_OL_I_ID = num;
                    }
                  while (temp[i].s_OL_QUANTITY == 0) {
                    errmsg("OL_QUANTITY must be entered.");
                  key=getintfld(temp[i].s_OL_QUANTITY,&num,2,row,45);
                  temp[i].s_OL_QUANTITY = num;
                    }
          if (temp[i].s_OL_SUPPLY_W_ID != neword->s_W_ID)
                    neword->s_all_local = 0;
                  neword->item[neword->s_O_OL_CNT].s_OL_SUPPLY_W_ID =
                  temp[i].s_OL_SUPPLY_W_ID;
                   neword->item[neword->s_O_OL_CNT].s_OL_I_ID =
                  temp[i].s_OL_I_ID;
                   neword->item[neword->s_O_OL_CNT].s_OL_QUANTITY =
                  temp[i].s_OL_QUANTITY;
                   neword->s_O_OL_CNT++;
                  } /* end if */
            row++;
            } /* end for */
          return(key);
}
/*---------------------------------------------------------------*/
/*             read payment screen                               */
/*---------------------------------------------------------------*/
static int read_payment (payment)
    struct payment_struct *payment;
   {
    static int     key, num;
    static double        fnum;
```

```
    payment->s_C_ID      = 0;
    payment->s_D_ID      = 0;
    payment->s_C_D_ID    = 0;
    payment->s_C_W_ID    = 0;
    payment->s_H_AMOUNT  = 0;
    payment->s_C_LAST[0] = '\0';
PF1:
   key = getintfld(payment->s_D_ID, &num, 2, 6, 52); /* district number */
   payment->s_D_ID = num;
   if (key == CNTRL_C_KEY) goto read_payment_exit;
   if (key == BACK_TAB_KEY) goto PF6;
PF2:
   key = getintfld(payment->s_C_ID, &num, 4, 11, 11); /* customer number */
   payment->s_C_ID = num;
   if (key == CNTRL_C_KEY) goto read_payment_exit;
   if (key == BACK_TAB_KEY) goto PF1;
PF3:
   key = getintfld(payment->s_C_W_ID, &num, 4, 11, 33); /* customer
warehouse */
   payment->s_C_W_ID = num;
   if (key == CNTRL_C_KEY) goto read_payment_exit;
   if (key == BACK_TAB_KEY) goto PF2;
PF4:
   key = getintfld(payment->s_C_D_ID, &num, 2, 11, 54); /* customer district */
   payment->s_C_D_ID = num;
   if (key == CNTRL_C_KEY) goto read_payment_exit;
   if (key == BACK_TAB_KEY) goto PF3;

PF5:
   key=getstrfld(payment->s_C_LAST,payment->s_C_LAST,16,12,29); /* last
name */
   if (key == CNTRL_C_KEY) goto read_payment_exit;
   if (key == BACK_TAB_KEY) goto PF4;
PF6:
   key = getmoneyfld(payment->s_H_AMOUNT, &fnum, 8, 17, 24); /* amount */
   if (fnum > 9999.99) {
         errmsg("Amount is too large.");
         goto PF6;
         }
   payment->s_H_AMOUNT = fnum;
   if (key == CNTRL_C_KEY) goto read_payment_exit;
   if (key == TAB_KEY) goto PF1;
   if (key == BACK_TAB_KEY) goto PF5;

read_payment_exit:
         while (payment->s_D_ID == 0) {
                  errmsg("D_ID must be entered.");
                  key = getintfld(payment->s_D_ID, &num, 2, 6, 52);
                  payment->s_D_ID = num;
                  }
         while ((payment->s_C_ID == 0) &&
               (payment->s_C_LAST[0] == '\0')) {
                  errmsg("C_ID or C_LAST must be entered.");
                  key = getintfld(payment->s_C_ID, &num, 4, 11, 11);
                  payment->s_C_ID = num;
                  if (payment->s_C_ID == 0)
key=getstrfld(payment->s_C_LAST,payment->s_C_LAST,16,12,29);
                  }
         while (payment->s_C_W_ID == 0) {
                  errmsg("C_W_ID must be entered.");
         key = getintfld(payment->s_C_W_ID, &num, 4, 11, 33);
                  payment->s_C_W_ID = num;
                  }
         while (payment->s_C_D_ID == 0) {
                  errmsg("C_D_ID must be entered.");
         key = getintfld(payment->s_C_D_ID, &num, 2, 11, 54);
                  payment->s_C_D_ID = num;
                  }
         while (payment->s_H_AMOUNT == 0.0) {
                  errmsg("H_AMOUNT must be entered.");
         key = getmoneyfld(payment->s_H_AMOUNT, &fnum, 8, 17, 24);
                  if (fnum > 9999.99)
                    errmsg("Amount is too large.");
                  else
                    payment->s_H_AMOUNT = fnum;
                  }
         return(key);
}
/*---------------------------------------------------------------*/
/*            read ordstat screen                                */
/*---------------------------------------------------------------*/
static int read_ordstat (ordstat)
    struct ordstat_struct *ordstat;
   {
    static int     key, num;
    static double        fnum;

    ordstat->s_C_ID      = 0;
    ordstat->s_D_ID      = 0;
    ordstat->s_C_LAST[0] = '\0';
OF1:
   key = getintfld(ordstat->s_D_ID, &num, 2, 4, 29); /* district number */
   ordstat->s_D_ID = num;
   if (key == CNTRL_C_KEY) goto read_ordstat_exit;
   if (key == BACK_TAB_KEY) goto OF3;
OF2:
   key = getintfld(ordstat->s_C_ID, &num, 4, 5, 11); /* customer number */
   ordstat->s_C_ID = num;
   if (key == CNTRL_C_KEY) goto read_ordstat_exit;
   if (key == BACK_TAB_KEY) goto OF1;
OF3:
   key = getstrfld(ordstat->s_C_LAST,ordstat->s_C_LAST,16,5,44); /* last
name */
   if (key == CNTRL_C_KEY) goto read_ordstat_exit;
   if (key == TAB_KEY) goto OF1;
   if (key == BACK_TAB_KEY) goto OF2;

read_ordstat_exit:
         while (ordstat->s_D_ID == 0) {
                  errmsg("D_ID must be entered.");
                  key = getintfld(ordstat->s_D_ID, &num, 2, 4, 29);
                  ordstat->s_D_ID = num;
                  }
         while ((ordstat->s_C_ID == 0) &&
               (ordstat->s_C_LAST[0] == '\0')) {
                  errmsg("C_ID or C_LAST must be entered.");
                  key = getintfld(ordstat->s_C_ID, &num, 4, 5, 11);
```

```
                ordstat->s_C_ID = num;
                if (ordstat->s_C_ID == 0)
          key=getstrfld(ordstat->s_C_LAST,ordstat->s_C_LAST,16,5,44);
                }
#ifdef DEBUG
fprintf(stderr, "ordstat->s_C_ID   : %d \n", ordstat->s_C_ID);
fprintf(stderr, "ordstat->s_C_LAST : %s \n", ordstat->s_C_LAST);
#endif
        return(key);
}
/*---------------------------------------------------------------*/
/*            read delivery screen                               */
/*---------------------------------------------------------------*/
static int read_delivery (delivery)
   struct delivery_struct *delivery;
  {
   static int     key, num;

   delivery->s_O_CARRIER_ID = 0;

do {
  key = getintfld(delivery->s_O_CARRIER_ID, &num, 2, 6, 17); /* Carrier Id */
  delivery->s_O_CARRIER_ID = num;
  if (key == CNTRL_C_KEY) goto read_delivery_exit;
  if (num == 0) {
        errmsg("Blank field not allowed.");
        key = TAB_KEY;
        }

  } while (key != ENTER_KEY);

read_delivery_exit:
       delivery->s_queued_time = get_time();
        return(key);
}
/*---------------------------------------------------------------*/
/*            read stocklev screen                               */
/*---------------------------------------------------------------*/
static int read_stocklev (stocklev)
   struct stocklev_struct *stocklev;
  {
   static int     key, num;

stocklev->s_threshold = 0;
do {
  key = getintfld(stocklev->s_threshold, &num, 2, 6, 24); /* threshold */
  stocklev->s_threshold = num;
  if (key == CNTRL_C_KEY) goto read_stocklev_exit;
  if (num == 0) {
        errmsg("Blank field not allowed.");
        key = TAB_KEY;
        }
    }
while (key != ENTER_KEY);
read_stocklev_exit:
        return(key);
 }
/*---------------------------------------------------------------*/
/*            write new order screen                             */
/*---------------------------------------------------------------*/
static write_neword (neword)
   struct neword_struct *neword;
  {
   static char     buf[1500];
   static int      i, pos, digits;

   if (neword->s_transtatus == FATAL_SQL_ERROR)
        return(fatal_error());
  pos = 0;

   if (neword->s_transtatus != INVALID_ITEM)
     {
      pos += rowcol(4,61,&buf[pos]);
      pos += mystrcpy(&buf[pos], neword->s_O_ENTRY_D);
      pos += rowcol(5,25,&buf[pos]);
      pos += mystrcpy(&buf[pos], neword->s_C_LAST);
      pos += rowcol(5,52,&buf[pos]);
      memcpy(&buf[pos], neword->s_C_CREDIT, 2);
      pos += 2;
      pos += rowcol(5,64,&buf[pos]);
      sprintf(&buf[pos], "%1.2f", neword->s_C_DISCOUNT * 100);
      pos += strlen(&buf[pos]);
      pos += rowcol(6,15,&buf[pos]);
      digits = myitoa(neword->s_O_ID, &buf[pos]);
      pos += digits;
      pos += rowcol(6,42,&buf[pos]);
      digits = myitoa(neword->s_O_OL_CNT, &buf[pos]);
      pos += digits;
      pos += rowcol(6,59,&buf[pos]);
      sprintf(&buf[pos], "%1.2f", neword->s_W_TAX * 100);
      pos += strlen(&buf[pos]);
      pos += rowcol(6,74,&buf[pos]);
      sprintf(&buf[pos], "%1.2f", neword->s_D_TAX * 100);
      pos += strlen(&buf[pos]);

      for (i = 0; i < neword->s_O_OL_CNT; i++)
         {
          if ((neword->item[i].s_OL_SUPPLY_W_ID == 0) &&
               (neword->item[i].s_OL_I_ID == 0) &&
               (neword->item[i].s_OL_QUANTITY == 0))
              continue;
          pos += rowcol((9+i),19,&buf[pos]);
          pos += mystrcpy(&buf[pos], neword->item[i].s_I_NAME);
          pos += rowcol((9+i),50,&buf[pos]);
          digits = myitoa(neword->item[i].s_S_QUANTITY, &buf[pos]);
          pos += digits;
          pos += rowcol((9+i),58,&buf[pos]);
          buf[pos] = neword->item[i].s_brand_generic;
          pos++;
          memcpy(&buf[pos], "   ", 3);
          pos += 3;
          myftomoney(neword->item[i].s_I_PRICE, &buf[pos], 7);
          pos += 7;
          pos += rowcol((9+i),71,&buf[pos]);
          myftomoney(neword->item[i].s_OL_AMOUNT, &buf[pos], 8);
          pos += 8;
         }
      pos += rowcol(24,70,&buf[pos]);
```

```
        myftomoney(neword->s_total_amount, &buf[pos], 9);
        pos += 9;
       }
     else
       {
        pos += rowcol(5,25,&buf[pos]);
        pos += mystrcpy(&buf[pos], neword->s_C_LAST);
        pos += rowcol(5,52,&buf[pos]);
        memcpy(&buf[pos], neword->s_C_CREDIT, 2);
        pos += 2;
        pos += rowcol(6,15,&buf[pos]);
        digits = myitoa(neword->s_O_ID, &buf[pos]);
        pos += digits;
        pos += rowcol(24,19,&buf[pos]);
        pos += mystrcpy(&buf[pos], "Item number is not valid");
       }
     pos += rowcol(2,1,&buf[pos]);
     buf[pos] = '\0';
     write(fdo, buf, pos);
     }
/*---------------------------------------------------------------------*/
/*              write payment screen                                   */
/*---------------------------------------------------------------------*/
static int write_payment (payment)
     struct payment_struct *payment;
    {
     static char     buf[800];
     static int      cd_size, i, pos, size;

     if (payment->s_transtatus == FATAL_SQL_ERROR)
          return(fatal_error());
     pos = 0;

     pos += rowcol(4,7,&buf[pos]);
     pos += mystrcpy(&buf[pos],payment->s_H_DATE);
     pos += rowcol(7,1,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_W_STREET_1);
     pos += rowcol(7,42,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_D_STREET_1);
     pos += rowcol(8,1,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_W_STREET_2);
     pos += rowcol(8,42,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_D_STREET_2);
     pos += rowcol(9,1,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_W_CITY);
     pos += rowcol(9,22,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_W_STATE);
     pos += rowcol(9,25,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_W_ZIP);
     pos += rowcol(9,42,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_D_CITY);
     pos += rowcol(9,63,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_D_STATE);
     pos += rowcol(9,66,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_D_ZIP);
     pos += rowcol(11,11,&buf[pos]);
     pos += formatitoa(payment->s_C_ID, &buf[pos], 4);
     pos += rowcol(12,9,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_FIRST);
     pos += rowcol(12,26,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_MIDDLE);
     pos += rowcol(12,29,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_LAST);
     pos += rowcol(12,58,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_SINCE);
     pos += rowcol(13,9,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_STREET_1);
     pos += rowcol(13,58,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_CREDIT);
     pos += rowcol(14,9,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_STREET_2);
     pos += rowcol(14,58,&buf[pos]);
     sprintf(&buf[pos], "%1.2f", payment->s_C_DISCOUNT * 100);
     pos += strlen(&buf[pos]);
     pos += rowcol(15,9,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_CITY);
     pos += rowcol(15,30,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_STATE);
     pos += rowcol(15,33,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_ZIP);
     pos += rowcol(15,58,&buf[pos]);
     pos += mystrcpy(&buf[pos], payment->s_C_PHONE);
     pos += rowcol(17,56,&buf[pos]);
     myftomoney(payment->s_C_BALANCE, &buf[pos], 13);
     pos += 13;
     pos += rowcol(18,18,&buf[pos]);
     myftomoney(payment->s_C_CREDIT_LIM, &buf[pos], 14);
     pos += 14;

     if (payment->s_C_CREDIT[0] == 'B' &&
         payment->s_C_CREDIT[1] == 'C')
        {
         pos += rowcol(20,12,&buf[pos]);
           cd_size = strlen(payment->s_C_DATA);
           for (i = 0; i < 4; i++)
            {
              if (cd_size > 50)
                  {
                      memcpy(&buf[pos], &payment->s_C_DATA[i*50], 50);
                  pos += 50;
                  cd_size -= 50;
                  }
              else
                {
                  strcpy(&buf[pos], &payment->s_C_DATA[i*50]);
                  pos += strlen(&payment->s_C_DATA[i*50]);
                  break;
                  }
              if (i < 3)
                  pos += rowcol((21+i),12,&buf[pos]);
             }
        }
     pos += rowcol(2,1,&buf[pos]);
     buf[pos] = '\0';
     write(fdo, buf, pos);
    }
/*---------------------------------------------------------------------*/
/*              write ordstat screen                                   */
```

```
/*---------------------------------------------------------------*/
static int write_ordstat (ordstat)
    struct ordstat_struct *ordstat;
   {
    static char     buf[1000];
    static int      i, pos;

    if (ordstat->s_transtatus == FATAL_SQL_ERROR)
         return(fatal_error());
    pos = 0;

    pos += rowcol(5,11,&buf[pos]);
    pos += formatitoa(ordstat->s_C_ID, &buf[pos], 4);
    pos += rowcol(5,24,&buf[pos]);
    pos += mystrcpy(&buf[pos], ordstat->s_C_FIRST);
    pos += rowcol(5,41,&buf[pos]);
    pos += mystrcpy(&buf[pos], ordstat->s_C_MIDDLE);
    pos += rowcol(5,44,&buf[pos]);
    pos += mystrcpy(&buf[pos], ordstat->s_C_LAST);
    pos += rowcol(6,16,&buf[pos]);
    myftomoney(ordstat->s_C_BALANCE, &buf[pos], 13);
    pos += 13;
    pos += rowcol(8,15,&buf[pos]);
    pos += myitoa(ordstat->s_O_ID, &buf[pos]);
    pos += rowcol(8,38,&buf[pos]);
    pos += mystrcpy(&buf[pos], ordstat->s_O_ENTRY_D);
    pos += rowcol(8,76,&buf[pos]);
    pos += myitoa(ordstat->s_O_CARRIER_ID, &buf[pos]);

    for (i = 0; i < ordstat->s_ol_cnt; i++)
       {
         buf[pos++] = '\n';
         pos += rowcol((i+10),3,&buf[pos]);
              pos += myitoa(ordstat->item[i].s_OL_SUPPLY_W_ID, &buf[pos]);
         pos += rowcol((i+10),15,&buf[pos]);
         pos += myitoa(ordstat->item[i].s_OL_I_ID, &buf[pos]);
         pos += rowcol((i+10),25,&buf[pos]);
         pos += myitoa(ordstat->item[i].s_OL_QUANTITY, &buf[pos]);
         pos += rowcol((i+10),33,&buf[pos]);
         myftomoney(ordstat->item[i].s_OL_AMOUNT, &buf[pos], 9);
         pos += 9;
         pos += rowcol((i+10),47,&buf[pos]);
              pos += mystrcpy(&buf[pos],ordstat->item[i].s_OL_DELIVERY_D);
         }
    pos += rowcol(2,1,&buf[pos]);
    buf[pos] = '\0';
    write(fdo, buf, pos);
   }
/*---------------------------------------------------------------*/
/*              write delivery screen                            */
/*---------------------------------------------------------------*/
static int write_delivery (delivery)
    struct delivery_struct *delivery;
   {
    static char     buf[100];
    static int      pos;
    pos = 0;

    pos += rowcol(8,19,&buf[pos]);
    pos += mystrcpy(&buf[pos], "Delivery has been queued");
    pos += rowcol(2,1,&buf[pos]);

    buf[pos] = '\0';
    write(fdo, buf, pos);
   }
/*---------------------------------------------------------------*/
/*              write stocklev screen                            */
/*---------------------------------------------------------------*/
static write_stocklev (stocklev)
    struct stocklev_struct *stocklev;
   {
    static char     buf[100];
    static int      pos;

    if (stocklev->s_transtatus == FATAL_SQL_ERROR)
         return(fatal_error());
    pos = 0;
    pos += rowcol(8,12,&buf[pos]);
    pos += myitoa(stocklev->s_low_stock, &buf[pos]);
    pos += rowcol(2,1,&buf[pos]);

    buf[pos] = '\0';
    write(fdo, buf, pos);
   }
/*---------------------------------------------------------------*/
/*              fatal error message                              */
/*---------------------------------------------------------------*/
fatal_error ()
{
char     buf[50];
char    *ptr;
int      pos;

ptr = buf;
ptr += rowcol(24,19,ptr);
ptr += mystrcpy(ptr, "SQL Fatal Error.");
/* For ACID test purposes, the following should be 10,10 */
/* else make it 2,1 */
/* Well, I don't know if this is  supposed to be here but.... CCC */
#ifdef ACID_TEST
#ifdef NO_CCCHACK
   rowcol(10,10,&buf[pos]);
   pos = strlen(buf);
   buf[pos] = '\0';
#else
   ptr += rowcol(10,10,ptr);
   *ptr = '\0';
#endif
#else
#ifdef NO_CCCHACK
   rowcol(2,1,&buf[pos]);
   pos = strlen(buf);
   buf[pos] = '\0';
#else
   ptr += rowcol(2,1,ptr);
   *ptr = '\0';
#endif
#endif
```

```
write(fdo, buf, strlen(buf));
}
/*---------------------------------------------------------------*/
/*              get integer field                                */
/*---------------------------------------------------------------*/
static  int getintfld(inum, onum, size, row, col)
    int             inum, *onum, size, row, col;
{
    static char     str[20], buf[20];
    static int      strlength, status, j, i, key, num;

    for (i = myitoa(inum, buf); i < size; i++)
        buf[i] = UNDERL;
    buf[i] = '\0';

    strlength = rowcol(row, col, str);
    strcpy(&str[strlength], buf);
    write(fdo, str, (strlength + size));

    do {
        key = get_term_value(buf, size, row, col);
        if (key == CNTRL_C_KEY)
            return (CNTRL_C_KEY);

        for (status = 0, i = 0, j = 0; i < size; i++) {
            if ((buf[i] >= '0' && buf[i] <= '9') && status == 0)
                str[j++] = buf[i];
            else if (buf[i] == SPACE || buf[i] == UNDERL) {
                if (j == 0)
                    continue;
                else
                    status = 1;
            } else {
                errmsg("An invalid character was detected.");
                status = -1;
                break;
            }
        }
    } while (status == -1);

    strlength = j;
    str[strlength] = '\0';
    *onum = atoi(str);

    for (i = (size - 1) - strlength; i > -1; --i)
        buf[i] = UNDERL;
    for (j = 0, i = size - strlength; i < size; i++, j++)
        buf[i] = str[j];

    strlength = rowcol(row, col, str);
    strcpy(&str[strlength], buf);
    write(fdo, str, (strlength + size));

    return (key);
}
/*---------------------------------------------------------------*/
/*              get money field                                  */
/*---------------------------------------------------------------*/
static
getmoneyfld(inum, onum, size, row, col)
    double          inum, *onum;
    int             size, row, col;
{
    static char     str[20], buf[20];
    static int      strlength, pointstatus, status, j, i, key, num;
    double          myatof();

    sprintf(buf, "%1.2f", inum);
    i = strlen(buf);

    for (; i < size; i++)
        buf[i] = UNDERL;
    buf[i] = '\0';

    strlength = rowcol(row, col, str);
    strcpy(&str[strlength], buf);
    write(fdo, str, (strlength + size));

    do {
        key = get_term_value(buf, size, row, col);
        if (key == CNTRL_C_KEY)
            return (CNTRL_C_KEY);

        for (pointstatus = 0, status = 0, i = 0, j = 0; i < size; i++) {
            if ((buf[i] >= '0' && buf[i] <= '9') && status == 0)
                str[j++] = buf[i];
            else if (buf[i] == UNDERL || buf[i] == SPACE) {
                if (j == 0)
                    continue;
                else
                    status = 1;
            } else if (buf[i] == '.' && status == 0 && pointstatus == 0) {
                str[j++] = buf[i];
                pointstatus = 1;
            } else if (buf[i] == '$' && j == 0) {
                str[j++] = buf[i];
            } else {
                errmsg("An invalid format was detected.");
                status = -1;
                break;
            }
        }
    } while (status == -1);

    strlength = j;
    str[strlength] = '\0';
    if (str[0] == '$')
        *onum = (double) myatof(&str[1]);
    else
        *onum = (double) myatof(str);

    myftomoney(*onum, buf, size);
    buf[size] = '\0';

    strlength = rowcol(row, col, str);
    strcpy(&str[strlength], buf);
    write(fdo, str, (strlength + size));
```

```
    return (key);
}
/*---------------------------------------------------------------*/
/*              get string field                                 */
/*---------------------------------------------------------------*/
static int getstrfld (istr, ostr, size, row, col)
          char    *istr, *ostr;
        int     size, row, col;
{
static char     str[80];
static int      i, key, strlength, istrlength;

strlength = rowcol(row,col,str);
istrlength = mystrcpy(&str[strlength],istr);
write(fdo,str, (strlength+istrlength));

strcpy(ostr, istr);
for (i=istrlength; i < size; i++) ostr[i] = UNDERL;

key = get_term_value(ostr, size, row, col);

for (i=size-1;; i--) {
        if (i == -1) {
                ostr[0] = '\0';
                break;
                }
        if (ostr[i] != UNDERL && ostr[i] != SPACE) {
                ostr[i+1] = '\0';
                break;
                }
        }
return(key);
}
/*---------------------------------------------------------------*/
/*              get term value                                   */
/*---------------------------------------------------------------*/
static int
get_term_value(buf, size, row, col)
    char          *buf;
    int           size, row, col;
{
    static int      last_cirbuf_pos = 0, cirbuf_pos = 0, readchars = 0;
    static int      buf_pos, curstr_pos, sizeplus;
    static char     cirbuf[CIRBUF_SIZE];
    static char     str[100], in_buf[100], curstr[100];
    static int      col_pos, strlength;
    int             no_chars_yet = 1;

    buf_pos = 0;
    col_pos = col;
    sizeplus = size + 1;

    strlength = rowcol(row, col_pos, str);
    write(fdo, str, strlength);

    for (;;) {
        if (readchars > 0) {
            for(curstr_pos = 0; readchars > 0; readchars--, last_cirbuf_pos++){
                if (cirbuf[last_cirbuf_pos] == TAB_KEY ||
                    cirbuf[last_cirbuf_pos] == ENTER_KEY ||
                    cirbuf[last_cirbuf_pos] == CNTRL_C_KEY ||
                    cirbuf[last_cirbuf_pos] == BACK_TAB_KEY) {
/* ANGELAHACK */
/*                  write(fdo, curstr, curstr_pos); */
                    ++last_cirbuf_pos;
                    --readchars;
                    return ((int) cirbuf[last_cirbuf_pos - 1]);
                } else if (cirbuf[last_cirbuf_pos] == BACKSPACE_KEY) {
                    if (col == col_pos) {
                        write(fdo, BEEP_SOUND, strlen(BEEP_SOUND));
                    } else {
                        --col_pos;
                        --buf_pos;
                        buf[buf_pos] = UNDERL;
                        curstr[curstr_pos] = '\0';
                        strcpy(str, curstr);
                        rowcol(row, col_pos, &str[strlen(str)]);
                        str[strlen(str)] = UNDERL;
                        rowcol(row, col_pos, &str[strlen(str)]);
                        write(fdo, str, strlen(str));
                        curstr_pos = 0;
                    }
                } else if (((cirbuf[last_cirbuf_pos] >= 'A' &&
                        cirbuf[last_cirbuf_pos] <= 'Z') ||
                          (cirbuf[last_cirbuf_pos] >= 'a' &&
                        cirbuf[last_cirbuf_pos] <= 'z') ||
                          (cirbuf[last_cirbuf_pos] >= '0' &&
                        cirbuf[last_cirbuf_pos] <= '9') ||
                          (cirbuf[last_cirbuf_pos] == ' ') ||
                          (cirbuf[last_cirbuf_pos] == '.')) &&
                           (buf_pos < size)) {
/* CCCHACK */
/*                  } else if (buf_pos < size) { */
                    if (no_chars_yet) {
                        int i;
                        no_chars_yet = 0;
                        buf[0] = cirbuf[last_cirbuf_pos];
                        for (i=0; i < size; i++)
                            buf[i] = UNDERL;
                        write(fdo, buf, size);
                        strlength = rowcol(row, col_pos, str);
                        write(fdo, str, strlength);
                        curstr_pos = 0;
                        buf_pos = 0;
                    }
                    curstr[curstr_pos] = cirbuf[last_cirbuf_pos];
                    ++curstr_pos;
                    buf[buf_pos] = cirbuf[last_cirbuf_pos];
                    ++buf_pos;
                    ++col_pos;
                } else
                    write(fdo, BEEP_SOUND, strlen(BEEP_SOUND));
            }
            write(fdo, curstr, curstr_pos);
        }
        /*---------------------------*/
        /* read screen                  */
```

```
        /*---------------------------*/
        readchars = read(fdi, in_buf, SCR_READ_NUM);

        if ((cirbuf_pos + readchars + 1) > CIRBUF_SIZE) {
            cirbuf_pos = 0;
            last_cirbuf_pos = 0;
        }
        memcpy(&cirbuf[cirbuf_pos], in_buf, readchars);
        cirbuf_pos += readchars;
    }
}
/*---------------------------------------------------------------*/
/*              myitoa - integer to ascii                        */
/*---------------------------------------------------------------*/
static int myitoa (num, buf)
        int num;
        char *buf;
{
static int i,j,sign;
static char temp[20];

if (( sign=num) < 0 ) num *= -1;

i = 0;
do      {
        temp[i++] = num % 10 + '0';
        } while ( (num /= 10 ) > 0);

if ( sign < 0 ) temp[i++]= '-';

for (--i, j=0; i >= 0; i--, j++)
        buf[j] = temp[i];

return(j);
}
/*---------------------------------------------------------------*/
/*              formatitoa - format integer to ascii             */
/*---------------------------------------------------------------*/
static formatitoa (num, buf, size)
        int num;
        char *buf;
          int size;
{
static int        i, j, len;
static char     str[20];

len = myitoa(num,buf);

if (len == size) return(size);

i = len;
j = size;
do { buf[--j] = buf[--i];
   } while(i > 0);

do { buf[--j] = UNDERL;
   } while(j > 0);

return(size);
}

/*---------------------------------------------------------------*/
/*              myatof - ascii to float                          */
/*---------------------------------------------------------------*/
static double myatof (buf)
        char *buf;
{
static double  num;
static int        i, j;

for (num=0.0, i=strlen(buf)-1, j=1; i >= 0; i--) {
          if (buf[i] == '.') {
                    num /= j;
                    j = 1;
                    }
          else if (buf[i] == '-') {
                    num *= -1.0;
                    }
          else {
                    num += ((buf[i] - '0') * j);
                    j *= 10;
                    }
          }
return(num);
}
/*---------------------------------------------------------------*/
/*              mymoneytoa - money to ascii                      */
/*---------------------------------------------------------------*/
static int mymoneytoa (num, buf)
        double num;
        char *buf;
{
static int        i, intnum, dec, strlength;
static double  fdec;

i=0;
if (num < 0.0) {
          buf[i++] = '-';
          num *= -1.0;
          }
intnum = (int)num;
fdec = num - (double)intnum;
dec = (int)((fdec*100.0)+0.5);

strlength = myitoa(intnum, &buf[i]);
if (dec > 0) {
          buf[strlength++] = '.';
          strlength += myitoa(dec, &buf[strlength]);
          }
return(strlength);
}
/*---------------------------------------------------------------*/
/*              myftomoney - float to formatted money            */
/*---------------------------------------------------------------*/
static int myftomoney (num, buf, space)
        double num;
        char *buf;
          int space;
```

```
{
static char     str[20];
static int      i;

sprintf(str, "$%1.2f", num);
i = strlen(str);

for (i--,space--; i >= 0 && space >= 0; i--, space--)
        buf[space] = str[i];
for (; space >= 0; space--)
        buf[space] = SPACE;
}
/*-----------------------------------------------------------------*/
/*              mystrcpy - string copy returning str length     */
/*-----------------------------------------------------------------*/
static int mystrcpy (str1, str2)
        char    *str1, *str2;
{
static int      i;

for (i=0; *str2 != '\0'; i++) {
        *str1 = *str2;
        str1++;
        str2++;
        }
return(i);
}
/*-----------------------------------------------------------------*/
/*  error message                                                  */
/*-----------------------------------------------------------------*/
errmsg (msg)
        char    *msg;
{
static char     str[100];

rowcol(ERRMSG_ROW,1,str);
strcat(str,msg);
write(fdo,str,strlen(str));
sleep(3);
rowcol(ERRMSG_ROW,1,str);
#ifdef WYSE50_TERM
strcat(str,"\033T");
#else
strcat(str,"                                ");
#endif
write(fdo,str,strlen(str));
}
/*-----------------------------------------------------------------*/
/*  end screen                                                     */
/*-----------------------------------------------------------------*/
end_screen ()
 {
  /* Set the original output termios config and flush it.*/
  ioctl(fdo, TCSETSW, &savetty);
  ioctl(fdo, TCFLSH, 2);

  /* Set the original input termios config and flush it.*/
  ioctl(fdi, TCSETSW, &savetty);
  ioctl(fdi, TCFLSH, 2);
  }
double get_time()
  {
   struct timeval      current;
   struct timezone     tp;

   gettimeofday(&current, &tp);
   return((double) (current.tv_sec) + (current.tv_usec / 1000000.0));
  }
```

tuxclient.h

```
/*************************************************************************/
/*      File: tuxclient.h                                                */
/*************************************************************************/
/*-----------------------------------------------------------------*/
/*      include files                                              */
/*-----------------------------------------------------------------*/
#include <atmi.h>           /* TUXEDO atmi library */
/*-----------------------------------------------------------------*/
/*      global variables                                           */
/*-----------------------------------------------------------------*/
struct  neword_struct               *neword_ptr;
struct  payment_struct              *payment_ptr;
struct  stocklev_struct     *stocklev_ptr;
struct  delivery_struct     *delivery_ptr;
struct  ordstat_struct              *ordstat_ptr;
int                         transtatus;
long                        TMoutbufsize;
/*-----------------------------------------------------------------*/
/*      connect to TM                                              */
/*-----------------------------------------------------------------*/
void connect_to_TM()
  {
   if (tpinit((TPINIT *)NULL) == -1)
     {
       printf("rtn: tpinit(), tperrno: %d \n", tperrno);
       exit(1);
     }
  }
/*-----------------------------------------------------------------*/
/*      Allocate buffers for TUXEDO servers.            */
/*-----------------------------------------------------------------*/
void alloc_buf_for_TM()
  {
   if ((neword_ptr = (struct neword_struct *)
                        tpalloc("CARRAY",
                                NULL,
                                NEWORD_LEN)) == NULL)
     {
          printf("rtn: tpalloc() failed for NEWORD, tperrno: %d \n", tperrno);
          exit(2);
     }
   if ((payment_ptr = (struct payment_struct *)
                        tpalloc("CARRAY",
                                NULL,
                                PAYMENT_LEN)) == NULL)
     {
          printf("rtn: tpalloc() failed for PAYMENT, tperrno: %d \n", tperrno);
```

```
            exit(2);
        }
    if ((delivery_ptr = (struct delivery_struct *)
                        tpalloc("CARRAY",
                                NULL,
                                DELIVERY_LEN)) == NULL)
        {
            printf("rtn: tpalloc() failed for DELIVERY, tperrno: %d \n", tperrno);
            exit(2);
        }
    if ((stocklev_ptr = (struct stocklev_struct *)
                        tpalloc("CARRAY",
                                NULL,
                                STOCKLEV_LEN)) == NULL)
        {
            printf("rtn: tpalloc() failed for STOCKLEV, tperrno: %d \n", tperrno);
            exit(2);
        }
    if ((ordstat_ptr = (struct ordstat_struct *)
                        tpalloc("CARRAY",
                                NULL,
                                ORDSTAT_LEN)) == NULL)
        {
            printf("rtn: tpalloc() failed for ORDSTAT, tperrno: %d \n", tperrno);
            exit(2);
        }
    }
/*---------------------------------------------------------------*/
/*      disconnect from TM                                       */
/*---------------------------------------------------------------*/
void disconnect_from_TM()
    {
    if (tpterm() == -1)
        {
        printf("rtn: tpterm(), tperrno: %d \n", tperrno);
        exit(3);
        }
    }
/*---------------------------------------------------------------*/
/*      submit neword transaction for NEWORD TUXEDO server.      */
/*---------------------------------------------------------------*/
int submit_TM_neword()
    {
    transtatus = tpcall("NEWORD_DSQL",
                        neword_ptr,
                        NEWORD_LEN,
                        &neword_ptr,
                        &TMoutbufsize,
                        0);
     if (transtatus == -1)
        {
        char    str[60];
     sprintf(str, "rtn: tpcall(NEWORD_DSQL), tperrno: %d, transtatus: %d \n",
                        tperrno, transtatus);
        fprintf(stderr, str);
        return(-1);
        }
    return(0);
    }
/*---------------------------------------------------------------*/
/*      submit payment transaction for PAYMENT TUXEDO server.*/
/*---------------------------------------------------------------*/
int submit_TM_payment()
    {
    transtatus = tpcall("PAYMENT_DSQL",
                        payment_ptr,
                        PAYMENT_LEN,
                        &payment_ptr,
                        &TMoutbufsize,
                        0);
    if (transtatus == -1)
        {
        char    str[60];
    sprintf(str, "rtn: tpcall(PAYMENT_DSQL), tperrno: %d, transtatus: %d \n",
                        tperrno, transtatus);
        fprintf(stderr, str);
        return(-1);
        }
    return(0);
    }
/*---------------------------------------------------------------*/
/*      submit stocklev transaction for STOCKLEV TUXEDO server.*/
/*---------------------------------------------------------------*/
int submit_TM_stocklev()
    {
    transtatus = tpcall("STOCKLEV_DSQL",
                        stocklev_ptr,
                        STOCKLEV_LEN,
                        &stocklev_ptr,
                        &TMoutbufsize,
                        0);
    if (transtatus == -1)
        {
        char    str[60];
        sprintf(str, "rtn: tpcall(STOCKLEV_DSQL), tperrno: %d, transtatus: %d
\n",
                        tperrno, transtatus);
        fprintf(stderr, str);
        return(-1);
        }
    return(0);
    }
/*---------------------------------------------------------------*/
/*      submit ordstat transaction for ORDSTAT TUXEDO server.    */
/*---------------------------------------------------------------*/
int submit_TM_ordstat()
    {
    transtatus = tpcall("ORDSTAT_DSQL",
                        ordstat_ptr,
                        ORDSTAT_LEN,
                        &ordstat_ptr,
                        &TMoutbufsize,
                        0);
    if (transtatus == -1)
        {
        char    str[60];
        sprintf(str, "rtn: tpcall(ORDSTAT_DSQL), tperrno: %d, transtatus: %d
\n",
```

```
                tperrno, transtatus);
        fprintf(stderr, str);
        return(-1);
      }
   return(0);
  }

/*---------------------------------------------------------------*/
/*      submit delivery transaction for DELIVERY TUXEDO server.*/
/*---------------------------------------------------------------*/
int submit_TM_delivery()
  {
   transtatus = tpacall("DELIVERY_DSQL",
                        delivery_ptr,
                       DELIVERY_LEN,
                       TPNOREPLY);

   if (transtatus == -1)
     {
        char    str[60];
        sprintf(str, "rtn: tpcall(DELIVERY_DSQL), tperrno: %d, transtatus: %d
\n",
                tperrno, transtatus);
        fprintf(stderr, str);
        return(-1);
     }
   return(0);
  }
```

msg.h

```
#define INPUT               0
#define OUTPUT              1

#define BASE_KEY 12345
#define NEWORD              0
#define PAYMENT  1
#define ORDSTAT  2
#define DELIVERY 3
#define STOCKLEV            4
#define TRAN                5
#define ACK         6
#define KILL        7

char logdir[] = "/tpcc/tpcs_work/app/SQLERR.client.";
 char logfname[80];
FILE *logfp;
```

A.2 Application Makefile

makefile

```
PRG = tpcc_bc main

BSFLAGS=

CARCHI      =     -qarch=ppc -qtune=604 -qro -qroconst -qmaxmem=-1

#CFLAGS = -O
#CFLAGS = -g
# ROOTDIR is tuxedo directory
ROOTDIR=/usr/tuxedo

INCDIR=$(ROOTDIR)/include
XCBINDIR=$(ROOTDIR)/bin

# add " -DDEBUG" in CGFLAGS for more debugging data
# add " -DUSE_TRACE" in CGFLAGS for trace hooks
CGFLAG=-D_ALL_SOURCE

CFLAGS=-O3 -Ddtp=1 -DNOWHAT=1 $(CARCHI) -I$(INCDIR) $(CGFLAG)

# CM profiling flag
#CMPROFFLAG=-D PROFIL_CM -D PROFIL_CM1
#CMPROFFLAG=-D PROFIL_CM2
CMPROFFLAG=

# CM perf optimization flag
#CMFLAG=-D CM_PERF -g
#CMFLAG=-D CM_PERF -D CM_PERF_TTY
CMFLAG=

#CC=gcc

all: $(PRG)

main: main.c termio.h ../incl/tpcc.h ../incl/tuxclient.h
        $(CC) $(CMPROFFLAG) $(CMFLAG) $(CFLAGS) -c main.c
        buildclient -v -o $@ -f $@.o
clean:
        rm -f *.o $(PRG) misc_fc term_fc
```

A.3 Sybase Stored Procedures

neworder.sql

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

        @i_id  int = 0, @ol_qty       tinyint = 0,
        @i_id2          int = 0, @ol_qty2     tinyint = 0,
        @i_id3          int = 0, @ol_qty3     tinyint = 0,
        @i_id4          int = 0, @ol_qty4     tinyint = 0,
        @i_id5          int = 0, @ol_qty5     tinyint = 0,
        @i_id6          int = 0, @ol_qty6     tinyint = 0,
        @i_id7          int = 0, @ol_qty7     tinyint = 0,
        @i_id8          int = 0, @ol_qty8     tinyint = 0,
        @i_id9          int = 0, @ol_qty9     tinyint = 0,
        @i_id10         int = 0, @ol_qty10    tinyint = 0,
        @i_id11         int = 0, @ol_qty11    tinyint = 0,
```

```
        @i_id12       int = 0, @ol_qty12   tinyint = 0,
        @i_id13       int = 0, @ol_qty13   tinyint = 0,
        @i_id14       int = 0, @ol_qty14   tinyint = 0,
        @i_id15       int = 0, @ol_qty15   tinyint = 0
)
as
declare
        @w_tax              real,    @d_tax              real,
        @c_last             char(16),      @c_credit           char(2),
        @c_discount real,    @commit_flag        tinyint,

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

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

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

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

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

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

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

        begin transaction NO

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

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

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

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

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

```
        if @d_id < @5
         if @d_id < @3
        if @d_id < @2       SELECT @s_dist = @s_dist_01
        else            SELECT @s_dist = @s_dist_02
              else if @d_id < @4    SELECT @s_dist = @s_dist_03
        else            SELECT @s_dist = @s_dist_04
        else if @d_id < @7
        if @d_id < @6       SELECT @s_dist = @s_dist_05
        else            SELECT @s_dist = @s_dist_06
        else if @d_id < @9
        if @d_id < @8       SELECT @s_dist = @s_dist_07
         else           SELECT @s_dist = @s_dist_08
        else if @d_id < @10SELECT @s_dist = @s_dist_09
else                    SELECT @s_dist = @s_dist_10
select @ol_qty_smallint = @ol_qty
if @s_quantity >= @ol_qty_smallint + @ten
        SELECT @s_quantity = @s_quantity - @ol_qty_smallint
else
        SELECT @s_quantity = @s_quantity - @ol_qty_smallint + 91

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

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

        INSERT INTO order_line (
        ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
        ol_supply_w_id, ol_delivery_d, ol_quantity,
        ol_amount, ol_dist_info)
        VALUES (
        @o_id, @d_id, @w_id, @ol_number, @i_id,
        @w_id, "19000101", @ol_qty_smallint,
        @ol_qty * @i_price, @s_dist)
        select          /* Return to client */
                @i_name, @i_price, @s_quantity, @b_g
        CLOSE c_no_is
end


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

        if (@commit_flag = @1)
                commit transaction NO
        else
                rollback transaction NO
        select          /* Return to client */
                @w_tax, @d_tax, @o_id, @c_last,
                @c_discount, @c_credit, @o_entry_d
end
go
if exists ( SELECT name FROM sysobjects WHERE name =
'neworder_remote')
        DROP PROC neworder_remote
go
CREATE PROC neworder_remote (
        @w_id               smallint,
        @d_id       tinyint,
        @c_id       int,
        @o_ol_cnt   tinyint,

@i_id  int = 0, @s_w_id     smallint = 0, @ol_qty       tinyint = 0,
@i_id2      int = 0, @s_w_id2   smallint = 0, @ol_qty2      tinyint = 0,
@i_id3      int = 0, @s_w_id3   smallint = 0, @ol_qty3      tinyint = 0,
@i_id4      int = 0, @s_w_id4   smallint = 0, @ol_qty4      tinyint = 0,
@i_id5      int = 0, @s_w_id5   smallint = 0, @ol_qty5      tinyint = 0,
@i_id6      int = 0, @s_w_id6   smallint = 0, @ol_qty6      tinyint = 0,
@i_id7      int = 0, @s_w_id7   smallint = 0, @ol_qty7      tinyint = 0,
@i_id8      int = 0, @s_w_id8   smallint = 0, @ol_qty8      tinyint = 0,
@i_id9      int = 0, @s_w_id9   smallint = 0, @ol_qty9      tinyint = 0,
@i_id10     int = 0, @s_w_id10  smallint = 0, @ol_qty10     tinyint = 0,
@i_id11     int = 0, @s_w_id11  smallint = 0, @ol_qty11     tinyint = 0,
@i_id12     int = 0, @s_w_id12  smallint = 0, @ol_qty12     tinyint = 0,
@i_id13     int = 0, @s_w_id13  smallint = 0, @ol_qty13     tinyint = 0,
@i_id14     int = 0, @s_w_id14  smallint = 0, @ol_qty14     tinyint = 0,
@i_id15     int = 0, @s_w_id15  smallint = 0, @ol_qty15     tinyint = 0
)
as
declare
@w_tax              real,   @d_tax              real,
@c_last             char(16),       @c_credit     char(2),
@c_discount real,   @commit_flag        tinyint,

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

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

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

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

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

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

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

        begin transaction NO
        OPEN c_no_wdc
        FETCH c_no_wdc INTO
                @w_tax, @d_tax, @o_id,
                @c_last, @c_discount, @c_credit
        UPDATE      district
                SET   d_next_o_id = @o_id + 1
                WHERE CURRENT OF c_no_wdc
        CLOSE c_no_wdc

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

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

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

                if (@@sqlstatus != 0) begin         /* item not found */
                        SELECT @commit_flag = 0
                        select /* Return to client */
                                NULL, NULL, NULL, NULL
                                continue
                end
if @d_id < @5
    if @d_id < @3
        if @d_id < @2       SELECT @s_dist = @s_dist_01
        else            SELECT @s_dist = @s_dist_02
       else if @d_id < @4   SELECT @s_dist = @s_dist_03
        else            SELECT @s_dist = @s_dist_04
else if @d_id < @7
        if @d_id < @6       SELECT @s_dist = @s_dist_05
        else            SELECT @s_dist = @s_dist_06
 else if @d_id < @9
        if @d_id < @8       SELECT @s_dist = @s_dist_07
         else           SELECT @s_dist = @s_dist_08
else if @d_id < @10         SELECT @s_dist = @s_dist_09
else                    SELECT @s_dist = @s_dist_10

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

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

                UPDATE stock set
                s_quantity          = @s_quantity,
                s_ytd       = @s_ytd + @ol_qty,
                s_remote_cnt  = @s_remote_cnt + @remote,
                s_order_cnt     = @s_order_cnt + @one
                        WHERE CURRENT OF c_no_is
                if (patindex("%ORIGINAL%", @i_data) > 0) and
                  (patindex("%ORIGINAL%", @s_data) > 0)
                        SELECT @b_g = "B"
                else
                        SELECT @b_g = "G"

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

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

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

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

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

order_status_byname.sql

```
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 mru) HOLDLOCK
        WHERE   c_w_id  =       @w_id
        AND     c_d_id  =       @d_id
        AND     c_last  =       @c_last
        ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
        FOR READ ONLY
BEGIN TRANSACTION OSNM
        SELECT @n = (count(*)+1)/2
        FROM customer (index c_non1 prefetch 2 mru) HOLDLOCK
        WHERE       c_w_id = @w_id and
                c_d_id = @d_id and
                c_last = @c_last
        OPEN c_find
        while (@n>0) begin
                FETCH c_find INTO @c_id
                SELECT @n = @n-1
        end
        CLOSE c_find
/* Get the latest order made by the customer */
        SELECT      @o_id = o_id, @o_carrier_id = o_carrier_id,
                @o_entry_d = o_entry_d
        FROMorders (index o_clu prefetch 16 mru) HOLDLOCK
        WHERE       o_w_id          = @w_id
        AND   o_d_id= @d_id
        AND   o_c_id= @c_id
        /* ORDER BY o_w_id, o_d_id, o_id */
/* Select order lines for the current order */
        select /* Return multiple rows to client */
                ol_supply_w_id,
                ol_i_id,
                ol_quantity,
                ol_amount,
                ol_delivery_d
        FROMorder_line HOLDLOCK
        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
        FROMcustomer (index c_clu prefetch 2 mru) HOLDLOCK
        WHERE   c_id    = @c_id
        AND   c_d_id  = @d_id
        AND   c_w_id  = @w_id
COMMIT TRANSACTION OSNM
```

```
go
```

order_status _byid.sql

```
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 mru) HOLDLOCK
        WHERE       o_w_id          = @w_id
        AND   o_d_id= @d_id
        AND   o_c_id= @c_id
        /* ORDER BY o_w_id, 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 mru) HOLDLOCK
        WHERE   c_id    = @c_id
        AND   c_d_id  = @d_id
        AND   c_w_id  = @w_id
COMMIT TRANSACTION OSID
go
```

payment_byname.sql

```
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,    @c_ytd_payment float,
        @c_payment_cnt     smallint,       @1      smallint,
        @data1       char(250),     @data2       char(250),
        @c_data_1    char(250),     @c_data_2  char(250)

declare @screen_data       char(200),     @today        datetime

declare c_pay_wd CURSOR FOR
        SELECT      w_street_1, w_street_2, w_city,
                w_state, w_zip, w_name, w_ytd,
                d_street_1, d_street_2, d_city,
                d_state, d_zip, d_name, d_ytd
        FROM 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_pay_c CURSOR FOR
        SELECT      c_first, c_middle, c_last, c_street_1, c_street_2,
                  c_city, c_state, c_zip, c_phone, c_credit, c_credit_lim,
                 c_discount, c_balance, c_ytd_payment, c_payment_cnt,
                c_since, c_data1, c_data2
        FROM customer (index c_clu prefetch 2 mru) HOLDLOCK
        WHERE       c_w_id          = @c_w_id
        AND   c_d_id= @c_d_id
        AND   c_id    = @c_id
        FOR UPDATE OF c_balance, c_payment_cnt, c_ytd_payment,
        c_data1, c_data2
declare c_find CURSOR FOR
        SELECT  c_id
        FROM customer (index c_non1 prefetch 2 mru) HOLDLOCK
        WHERE   c_w_id  =       @c_w_id
        AND     c_d_id  =       @c_d_id
        AND     c_last  =       @c_last
        ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
        FOR READ ONLY

BEGIN TRANSACTION PNM
        SELECT @n = (count(*)+1)/2
```

```
        FROM customer (index c_non1 prefetch 2 mru) HOLDLOCK
        WHERE       c_w_id = @c_w_id and
                c_d_id = @c_d_id and
                c_last = @c_last
        OPEN c_find
        while (@n>0) begin
                FETCH c_find INTO @c_id
                SELECT @n = @n-1
        end
        CLOSE c_find
        select @1 = 1
        OPEN c_pay_wd
        FETCH c_pay_wd INTO
                @w_street_1, @w_street_2, @w_city,
                @w_state, @w_zip, @w_name, @w_ytd,
                @d_street_1, @d_street_2, @d_city,
                @d_state, @d_zip, @d_name, @d_ytd
        UPDATE district
                SET     d_ytd  = @d_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        UPDATE warehouse
                SET w_ytd = @w_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        CLOSE c_pay_wd

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

        SELECT      @c_payment_cnt = @c_payment_cnt + @1,
                @c_balance = @c_balance - @h_amount,
                @c_ytd_payment = @c_ytd_payment + @h_amount

        if (@c_credit = "BC")
        begin
        SELECT  @c_data_2 =
                substring(@data1, 209, 42) +
                substring(@data2, 1, 208)
        SELECT  @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) + substring(@data1, 1, 208)
        SELECT      @screen_data = substring(@c_data_1, 1, 200)

        UPDATE customer SET
                c_payment_cnt = @c_payment_cnt,
                c_ytd_payment = @c_ytd_payment,
                c_balance = @c_balance,
                c_data1 = @c_data_1, c_data2 = @c_data_2
                WHERE CURRENT OF c_pay_c
end
else begin
        SELECT      @screen_data = NULL
        UPDATE      customer SET
                c_payment_cnt       = @c_payment_cnt,
                c_balance     = @c_balance,
                c_ytd_payment       = @c_ytd_payment
                WHERE CURRENT OF c_pay_c
end
CLOSE c_pay_c
/* Create the history record */
   SELECT @today = getdate()
INSERT INTO history (
        h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
        h_date, h_amount, h_data)
VALUES (
        @c_id, @c_d_id, @c_w_id, @d_id, @w_id,
        @today, @h_amount, (@w_name + "    " + @d_name))

COMMIT TRANSACTION PNM

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

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

                @c_first,
                @c_middle,
                @c_street_1,
                @c_street_2,
                @c_city,
                @c_state,
                @c_zip,
                @c_phone,
                @c_since,
                @c_credit,
                @c_credit_lim,
                @c_discount,
                @c_balance,
                @screen_data
go
```

payment_byid.sql

```
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,    @c_ytd_payment     float,
        @c_payment_cnt     smallint,        @1      smallint,
        @data1                  char(250),      @data2          char(250),
        @c_data_1    char(250),      @c_data_2    char(250)

declare @screen_data        char(200),      @today          datetime

declare c_pay_wd CURSOR FOR
        SELECT          w_street_1, w_street_2, w_city,
                w_state, w_zip, w_name, w_ytd,
                d_street_1, d_street_2, d_city,
                d_state, d_zip, d_name, d_ytd
        FROM 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_pay_c CURSOR FOR
SELECT          c_first, c_middle, c_last, c_street_1, c_street_2,
        c_city, c_state, c_zip, c_phone, c_credit, c_credit_lim,
        c_discount, c_balance, c_ytd_payment, c_payment_cnt,
        c_since, c_data1, c_data2
FROM customer (index c_clu prefetch 2 mru) HOLDLOCK
WHERE           c_w_id          = @c_w_id
AND   c_d_id = @c_d_id
AND   c_id    = @c_id
FOR UPDATE OF     c_balance, c_payment_cnt, c_ytd_payment,
        c_data1, c_data2
BEGIN TRANSACTION PID
        select @1 = 1
        OPEN c_pay_wd
        FETCH c_pay_wd INTO
                @w_street_1, @w_street_2, @w_city,
                @w_state, @w_zip, @w_name, @w_ytd,
                @d_street_1, @d_street_2, @d_city,
                @d_state, @d_zip, @d_name, @d_ytd
        UPDATE district
                SET     d_ytd  = @d_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        UPDATE warehouse
                SET w_ytd = @w_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        CLOSE c_pay_wd

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

        SELECT       @c_payment_cnt = @c_payment_cnt + @1,
                @c_balance = @c_balance - @h_amount,
                @c_ytd_payment = @c_ytd_payment + @h_amount

if (@c_credit = "BC")
begin
        SELECT  @c_data_2 =
                substring(@data1, 209, 42) +
                substring(@data2, 1, 208)
        SELECT  @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) + substring(@data1, 1, 208)
        SELECT       @screen_data = substring(@c_data_1, 1, 200)

        UPDATE customer SET
                c_payment_cnt = @c_payment_cnt,
                c_ytd_payment = @c_ytd_payment,
                c_balance = @c_balance,
                c_data1 = @c_data_1, c_data2 = @c_data_2
          WHERE CURRENT OF c_pay_c
end
else begin
        SELECT       @screen_data = NULL
        UPDATE       customer SET
                c_payment_cnt       = @c_payment_cnt,
                c_balance     = @c_balance,
                c_ytd_payment       = @c_ytd_payment
                WHERE CURRENT OF c_pay_c
end
CLOSE c_pay_c
        /* Create the history record */
        SELECT @today = getdate()
        INSERT INTO history (
                h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                h_date, h_amount, h_data)
        VALUES (
                @c_id, @c_d_id, @c_w_id, @d_id, @w_id,
                @today, @h_amount, (@w_name + "    " + @d_name))
COMMIT TRANSACTION PID
        select          /* Return to client */
                @c_id,
```

```
        @c_last,
        @today,
        @w_street_1,
        @w_street_2,
        @w_city,
        @w_state,
        @w_zip,

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

        @c_first,
        @c_middle,
        @c_street_1,
        @c_street_2,
        @c_city,
        @c_state,
        @c_zip,
        @c_phone,
        @c_since,
        @c_credit,
        @c_credit_lim,
        @c_discount,
        @c_balance,
        @screen_data
go
```

delivery.sql

```
if exists (select * from sysobjects where name = 'delivery')
        drop proc delivery
go
CREATE PROC delivery
        @w_id                   smallint,
        @o_carrier_id           smallint,
        @d_id           tinyint = 1
as
declare         @no_o_id        int,            @o_c_id         smallint,
        @ol_total       float,  @ol_amount      float,  @junk_id    smallint,
        @today                  datetime,       @1      smallint,
        @one            tinyint, @ten   tinyint
declare c_del_no CURSOR FOR
        SELECT          no_o_id
        FROM new_order (index no_clu) HOLDLOCK
        WHERE           no_d_id = @d_id
        AND     no_w_id = @w_id
        FOR UPDATE
/*
** The only purpose of the index hint in the above is to ensure
** that the clustered index is used.  As it turns out, our optimizer
** chooses the clustered index anyway -- with or without the hint.
*/
declare c_del_ol CURSOR FOR
        SELECT          ol_amount
        FROM order_line HOLDLOCK
        WHERE           ol_o_id         = @no_o_id
        AND     ol_d_id         = @d_id
        AND     ol_w_id         = @w_id
        FOR UPDATE OF ol_delivery_d
declare c_del_o CURSOR FOR
        SELECT          o_c_id, o_carrier_id
        FROM orders HOLDLOCK
        WHERE           o_id    = @no_o_id
        AND     o_d_id= @d_id
        AND     o_w_id          = @w_id
        FOR UPDATE OF o_carrier_id
begin
   select @1 = 1, @one = 1, @ten = 10
   while (@d_id <= @ten) begin
        BEGIN TRANSACTION DEL
        OPEN c_del_no
        FETCH c_del_no INTO @no_o_id

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

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

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

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

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

                COMMIT TRANSACTION DEL
                select     /* Return to client */
                        @no_o_id
```

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

stock_level.sql

```
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)) /* Return to client */
        FROM district,
                order_line (index ol_clu prefetch 2 mru),
                stock (index s_clu prefetch 2 lru)
        WHERE       d_w_id          =       @w_id
        AND   d_id    =       @d_id
        AND   ol_w_id         =       @w_id
        AND   ol_d_id         =       @d_id
     AND       ol_o_id         between (d_next_o_id - 20)     and  (d_next_o_id -
1)
        AND   s_w_id          =       ol_w_id
        AND   s_i_id =        ol_i_id
        AND     s_quantity < @threshold
go
```

A.4 Tuxedo Server Code

neword.c

```
/*******************************************************************************/
/*      FILE : neword.c                                */
/*******************************************************************************/
/*******************************************************************************/
/*                              INCLUDES FILES          */
/*******************************************************************************/
#include <atmi.h>               /* Tuxedo header file */
#include <sybfront.h>                   /* Sybase include file */
#include <sybdb.h>              /* Sybase include file */
#include <cspublic.h>                   /* Sybase include file */
#include <syberror.h>                   /* Sybase include file */
#include <stdio.h>
#include "syb_tpcc.h"
/*******************************************************************************/
/*                              CONSTANTS                               */
/*******************************************************************************/
#define INVALID_ITEM            100
#define TRAN_OK                 0
#define FATAL_SQL_ERROR                 -1
/*******************************************************************************/
/*                              STRUCTURES                      */
/*******************************************************************************/
struct neword_struct
{
  DBSMALLINT     s_W_ID;
  DBTINYINT      s_D_ID;
  DBINT      s_C_ID;
  char       s_C_LAST[17];
  char       s_C_CREDIT[3];
  DBREAL     s_C_DISCOUNT;
  DBTINYINT      s_O_OL_CNT;
  DBINT      s_O_ID;
  char       s_O_ENTRY_D[20];
  char       s_status_line[25];
  DBFLT8     s_total_amount;
  DBTINYINT      s_all_local;
  short      s_transtatus;
  DBREAL     s_W_TAX;
  DBREAL     s_D_TAX;
  struct items_struct
    {
     DBSMALLINT s_OL_SUPPLY_W_ID;
      DBINT             s_OL_I_ID;
      char          s_I_NAME[25];
      DBTINYINT  s_OL_QUANTITY;
      DBSMALLINT        s_S_QUANTITY;
      char          s_brand_generic[2];
      DBFLT8            s_I_PRICE;
      DBFLT8            s_OL_AMOUNT;
    } item[15];
  } *neword;
/*******************************************************************************/
/*                              GLOBAL VARIABLES                */
/*******************************************************************************/
int             bad_items = 0;
DBFLT8              total_amount,
                tax_n_discount;
/*******************************************************************************/
/* FUNCTION : tpsvrinit()                               */
/* GOAL  : Initialize the sybase environnement with opening database and */
/*         setting transactions characteristics.                */
/*         Moreover it is used first time when TUXEDO server is booted.    */
/*******************************************************************************/

tpsvrinit (argc, argv)
  int   argc;
  char *argv[];
 {
  /* Install the error and message handler */
  dberrhandle(err_handler);
  dbmsghandle(msg_handler);

  login = dblogin();
  DBSETLUSER(login, "sa");
  DBSETLPWD(login, "syBase");
  DBSETLPACKET(login, 4096);
  DBSETLCHARSET(login, "iso_1");
  /* Establish connection to Sybase */
  if ((dbproc = dbopen(login, NULL)) == NULL)
  {
        fprintf(stderr, "Fatal:Could not open connection\n");
        exit(-1);
```

```
  }
  dbloginfree(login); /* release the login record */
  dbuse(dbproc, "tpcc");
 }
/*******************************************************************************/
/* FUNCTION : NEWORD_DSQL()                                    */
/* GOAL     : It the TUXEDO server design for serving the new_order  */
/*            transaction called by the TUXEDO client via a tpcall() call.    */
/*            It will send back the result of the transaction to the TUXEDO   */
/*            client by using the tpreturn() call.                 */
/*******************************************************************************/
NEWORD_DSQL(msg)
  TPSVCINFO *msg;         /* Tuxedo buffer message structure */
 {
  int             try   = 0;
/* Getting the data area from the TUXEDO buffer message. */
  neword = (struct neword_struct *) msg->data;
  /* Repeat until we give up trying */
  for (try = 0; try < MaxTries; try++)
     {
       if (try > 0)
         display_xction("repeating");

       deadlock     = 0;
       total_amount = 0.0;
/* If the transactation succeeds, then done */
      if (neworder_body() == SUCCEED)
        break;
      /* Clean up and try again */
      dbcancel(dbproc);
      sleep_before_retry();
     }
tax_n_discount = (1 - neword->s_C_DISCOUNT) * (1 + neword->s_W_TAX
                                               + neword->s_D_TAX);
  neword->s_total_amount = total_amount * tax_n_discount;

  if (try >= MaxTries)
    {
     display_xction("failed");
     neword->s_transtatus = FATAL_SQL_ERROR;
    }
  else if (neword->s_transtatus != INVALID_ITEM)
    {
     neword->s_transtatus = TRAN_OK;
    }
  tpreturn(TPSUCCESS, 0, neword, sizeof(struct neword_struct), 0);
 }
/*******************************************************************************/
/* FUNCTION : neworder_body()                                  */
/* GOAL     :                                  */
/*       Call via the RPC the necessary procedures to make the new_order */
/*            transaction called by the TUXEDO client via a tpcall() call.    */
/*            Get the results information after the completion of the    .    */
/*            transaction wich will be send back to the client via TUXEDO.    */
/*******************************************************************************/
int neworder_body ()
  {
   int           i;
   DBINT      retcode;
   DBDATETIME     o_entry_d;

   deadlock = 0;
   if (neword->s_all_local)
     {
      dbrpcinit(dbproc, "neworder_local", 0);
     }
   else
     {
      dbrpcinit(dbproc, "neworder_remote", 0);
     }
dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &neword->s_W_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, &neword->s_D_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, &neword->s_C_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, &neword->s_O_OL_CNT);
   /* Send the orderlines (up to 15) */
   for(i = 0; i < (int)neword->s_O_OL_CNT; i++)
      {
       dbrpcparam(dbproc,NULL,0,SYBINT4, -1, -1, &neword-
>item[i].s_OL_I_ID);
       if (!neword->s_all_local)
           {
           dbrpcparam(dbproc,NULL,0,SYBINT2, -1, -1,
                   &neword->item[i].s_OL_SUPPLY_W_ID);
           }
       dbrpcparam(dbproc,NULL,0,SYBINT1,-1,-1,&neword-
>item[i].s_OL_QUANTITY);
      }
   /* Execute the neworder transaction */
   if (dbrpcsend(dbproc) != SUCCEED)
     return FAIL;
   if (dbsqlok(dbproc)   != SUCCEED)
     return FAIL;

   /* Get the results of the overall neworder transaction */
   for (i = 0; i < (int)neword->s_O_OL_CNT; i++)
        {
         /* Control that there is no deadlock or pb */
       if (dbresults(dbproc) != SUCCEED || deadlock)
           {
           return FAIL;
           }
         else
           {
            /* Get some first results */
 dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(neword->item[i].s_I_NAME),
                        neword->item[i].s_I_NAME);
  dbbind(dbproc, 2, FLT8BIND, 0, &neword->item[i].s_I_PRICE);
 dbbind(dbproc, 3, SMALLBIND, 0, &neword->item[i].s_S_QUANTITY),
 dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(neword
>item[i].s_brand_generic),neword->item[i].s_brand_generic);
           if (dbnextrow(dbproc) != REG_ROW)
             return FAIL;
           /* Control that there is no bad items in the transaction */
           if (*neword->item[i].s_I_NAME == '\0')
             {
              bad_items++;
              neword->s_transtatus = INVALID_ITEM;
             }
```

```
            /* Control that there is no deadlock or pb */
            if (dbcanquery(dbproc) != SUCCEED || deadlock)
                return FAIL;
            }
          /* Control that there is no deadlock or pb */
          if (dbhasretstat(dbproc))
            {
             if (retcode = dbretstatus(dbproc) == -3)
               {
                deadlock = 1;
                  display_xction("deadlock detected");
                }
             else if (retcode < 0)
               {
                  display_xction("unknown return code");
                }
             return FAIL;
            }
          /* Set the new Order Line Items amount */
        neword->item[i].s_OL_AMOUNT = neword->item[i].s_I_PRICE *
                        neword->item[i].s_S_QUANTITY;
          /* Calculate the global amount of all Order Line Items */
          total_amount += neword->item[i].s_OL_AMOUNT;
        }
    /* Control that there is no deadlock or pb */
    if (dbresults(dbproc) != SUCCEED || deadlock)
      return FAIL;
    /* Get some others results */
    dbbind(dbproc, 1, REALBIND, 0, &neword->s_W_TAX);
    dbbind(dbproc, 2, REALBIND, 0, &neword->s_D_TAX);
    dbbind(dbproc, 3, INTBIND, 0, &neword->s_O_ID);
    dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(neword->s_C_LAST),
                                neword->s_C_LAST);
    dbbind(dbproc, 5, REALBIND, 0, &neword->s_C_DISCOUNT);
    dbbind(dbproc, 6, NTBSTRINGBIND, sizeof(neword->s_C_CREDIT),
                                 neword->s_C_CREDIT);
    dbbind(dbproc, 7, DATETIMEBIND, 0, &o_entry_d);
    /* Control that there is no deadlock or pb */
    if (dbnextrow(dbproc) != REG_ROW || deadlock)
      return FAIL;
    /* Control that there is no deadlock or pb */
    if (dbcanquery(dbproc) != SUCCEED || deadlock)
      return FAIL;

    put_convert_date_in_neword(&o_entry_d);

    return SUCCEED;
   }
void put_convert_date_in_neword(sybase_date)
    DBDATETIME *sybase_date;
   {
    DBDATEREC      daterec;

    dbdatecrack(NULL, &daterec, sybase_date);

    sprintf(neword->s_O_ENTRY_D,"%02d-%02d-%04d %02d:%02d:%02d",
            daterec.datedmonth, daterec.datemonth+1, daterec.dateyear,
            daterec.datehour, daterec.dateminute, daterec.datesecond);
   }
void sleep_before_retry()
   {
    sleep(rand()%3+1);
   }
int err_handler(dbproc, severity, errno, oserr)
    DBPROCESS      *dbproc;
    int   severity, errno, oserr;
   {
    fprintf(stderr, "DB-LIBRARY Error %d:", errno);
    display_xction(dberrstr(errno));

    if (oserr != DBNOERR)
      {
       fprintf(stderr, "AIX Error :");
       fflush(stderr);
       display_xction(dboserrstr(oserr));
      }
    neword->s_transtatus = FATAL_SQL_ERROR;

    dbcancel(dbproc);
    dbcmd(dbproc, "rollback transaction NO");
    dbsqlexec(dbproc);

    while (dbresults(dbproc) != SUCCEED)
          dbcanquery(dbproc);

    tpreturn(TPSUCCESS, 0, neword, sizeof(struct neword_struct), 0);
   }
int msg_handler(dbproc, msgno, msgstate, severity, msgtext, servername,
procname, line)
     DBPROCESS     *dbproc;
     int  msgno, msgstate, severity;
     char         *msgtext, *servername, *procname;
     int  line;
    {
     /* Changing database messages */
     if (msgno == CONTEXT_SET || msgno == LANGUAGE_SET ||
         msgno == CHARACTER_SET)
       return FAIL;

     fprintf(stderr, "msg no %d", msgno);
     fflush(stderr);
     display_xction(msgtext);

     if (msgno == ABORT_ERROR)
         return FAIL;
     /* Control if there is a deadlock */
     if (msgno == 1205)
       {
         display_xction(msgtext);
         deadlock = 1;
         return FAIL
        }
     /* Rollback transaction and signal fatal error on any other error */
     dbcancel(dbproc);
     dbcmd(dbproc, "rollback transaction NO");
     dbsqlexec(dbproc);

    while (dbresults(dbproc) != SUCCEED)
```

```
        dbcanquery(dbproc);

    tpreturn(TPSUCCESS, 0, neword, sizeof(struct neword_struct), 0);
  }
void display_xction(msg)
    char *msg;
  {
    int i;

    fprintf(stderr, "%s NEWORDER\n", msg);

    fprintf(stderr, "wid = %d, did = %d, cid = %d\n", neword->s_W_ID,
                                neword->s_D_ID,
                                 neword->s_C_ID);
    fprintf(stderr, "%d items : \n[", neword->s_O_OL_CNT);
    for (i = 0; i <= (int) neword->s_O_OL_CNT; i++)
       fprintf(stderr, " %d", neword->item[i].s_OL_I_ID);
    fprintf(stderr, "]\n");
    fflush(stderr);
  }
```

neword.c

```
/*****************************************************************************/
/*       FILE : ordstat.c                                                    */
/*****************************************************************************/
/*****************************************************************************/
/*                          INCLUDES FILES                                   */
/*****************************************************************************/
#include <atmi.h>           /* Tuxedo header file */
#include <sybfront.h>               /* Sybase include file */
#include <sybdb.h>          /* Sybase include file */
#include <cspublic.h>               /* Sybase include file */
#include <syberror.h>               /* Sybase include file */
#include <stdio.h>
#include "syb_tpcc.h"
/*****************************************************************************/
/*                          CONSTANTS                                        */
/*****************************************************************************/
#define TRAN_OK             0
#define FATAL_SQL_ERROR             -1
/*****************************************************************************/
/*                          STRUCTURES                                       */
/*****************************************************************************/
struct ordstat_struct
  {
    DBSMALLINT     s_W_ID;
    DBTINYINT      s_D_ID;
    DBINT      s_C_ID;
    char       s_C_FIRST[17];
    char       s_C_MIDDLE[3];
    char       s_C_LAST[17];
    DBFLT8     s_C_BALANCE;
    DBINT      s_O_ID;
    char       s_O_ENTRY_D[20];
    DBSMALLINT     s_O_CARRIER_ID;
    short      s_ol_cnt;
    short      s_transtatus;
    struct oitems_struct
      {
       DBSMALLINT      s_OL_SUPPLY_W_ID;
       DBINT           s_OL_I_ID;
       DBTINYINT       s_OL_QUANTITY;
       DBFLT8          s_OL_AMOUNT;
         char          s_OL_DELIVERY_D[20];
      } item[15];
  } *ordstat;
/*****************************************************************************/
/* FUNCTION : tpsvrinit()                                                    */
/* GOAL  : Initialize the sybase environnement with opening database and */
/*         setting transactions characteristics.                             */
/*         Moreover it is used first time when TUXEDO server is booted.    */
/*****************************************************************************/
tpsvrinit (argc, argv)
  int   argc;
  char *argv[];
 {
  /* Install the error and message handler */
  dberrhandle(err_handler);
  dbmsghandle(msg_handler);

  login = dblogin();
  DBSETLUSER(login, "sa");
  DBSETLPWD(login, "syBase");
  DBSETLPACKET(login, 4096);
  DBSETLCHARSET(login, "iso_1");

  /* Establish connection to Sybase */
  if ((dbproc = dbopen(login, NULL)) == NULL)
  {
       fprintf(stderr, "Fatal:Could not open connection\n");
       exit(-1);
  }
  dbloginfree(login); /* release the login record */
  dbuse(dbproc, "tpcc");
 }
/*****************************************************************************/
/* FUNCTION : ORDSTAT_DSQL()                                                 */
/* GOAL    : It the TUXEDO server design for serving the ordstat           */
/*         transaction called by the TUXEDO client via a tpcall() call.   */
/*         It will send back the result of the transaction to the TUXEDO   */
/*         client by using the tpreturn() call.                            */
/*****************************************************************************/
  ORDSTAT_DSQL(msg)
  TPSVCINFO *msg;        /* Tuxedo buffer message structure */
 {
  int            try   = 0;
  DBINT      test_id;
  /* Getting the data area from the TUXEDO buffer message. */
  ordstat = (struct ordstat_struct *) msg->data;

  test_id = ordstat->s_C_ID;
  /* Repeat until we give up trying */
  for (try = 0; try < MaxTries; try++)
    {
      if (try > 0)
         display_xction("repeating");
     if (test_id)
```

```
        {
          if (ordstat_byid_begin() != SUCCEED)
            {
             dbcancel(dbproc);
             fprintf(stderr, "Error in ordstat_byid_begin()\n");
             sleep_before_retry();
             continue;
            }
          }
          else
          {
            if (ordstat_byname_begin() != SUCCEED)
              {
               dbcancel(dbproc);
               fprintf(stderr, "Error in ordstat_byid_begin()\n");
               sleep_before_retry();
               continue;
              }
          }
        if (ordstat_end() != SUCCEED)
            {
             dbcancel(dbproc);
             fprintf(stderr, "Error in ordstat_end()\n");
             sleep_before_retry();
             continue;
            }
        break;
       }
     if (try >= MaxTries)
       {
        display_xction("failed");
        ordstat->s_transtatus = FATAL_SQL_ERROR;
       }
     else
       {
        ordstat->s_transtatus = TRAN_OK;
       }
     tpreturn(TPSUCCESS, 0, ordstat, sizeof(struct ordstat_struct), 0);
    }
int ordstat_byid_begin()
    {
     deadlock = 0;
     dbrpcinit(dbproc, "order_status_byid", 0);
     dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &ordstat->s_W_ID);
     dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &ordstat->s_D_ID);
     dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &ordstat->s_C_ID);
     return(dbrpcsend(dbproc) == SUCCEED ? SUCCEED : FAIL);
    }
int ordstat_byname_begin()
    {
     deadlock = 0;
     dbrpcinit(dbproc, "order_status_byname", 0);
     dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &ordstat->s_W_ID);
     dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &ordstat->s_D_ID);
     dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(ordstat->s_C_LAST),
                                  ordstat->s_C_LAST);
     return(dbrpcsend(dbproc) == SUCCEED ? SUCCEED : FAIL);
    }
int ordstat_end()
    {
     int            i = 0,
                    code = 0;
     DBDATETIME   ol_delivery_date,
                  o_entry_date;
     DBSMALLINT   ol_supply_w_id;
     DBINT        ol_i_id;
     DBTINYINT          ol_quantity;
     DBFLT8     ol_amount;

     if (dbsqlok(dbproc) != SUCCEED)
       {
        fprintf(stderr, "dbsqlok() failed for ordstat_end()\n");
        return FAIL;
       }

     if (dbresults(dbproc) != SUCCEED || deadlock)
       {
        fprintf(stderr, "dbresults() failed for ordstat_end()\n");
        return FAIL;
       }
     else
       {
        dbbind(dbproc,1, SMALLBIND, 0, &ol_supply_w_id);
        dbbind(dbproc,2, INTBIND, 0, &ol_i_id);
        dbbind(dbproc,3, TINYBIND, 0, &ol_quantity);
        dbbind(dbproc,4, FLT8BIND, 0, &ol_amount);
        dbbind(dbproc,5, DATETIMEBIND, 0, &ol_delivery_date);
       for (i = 0; (code = dbnextrow(dbproc)) == REG_ROW && !deadlock; i++)

            {
             ordstat->item[i].s_OL_SUPPLY_W_ID = ol_supply_w_id;
             ordstat->item[i].s_OL_I_ID        = ol_i_id;
             ordstat->item[i].s_OL_QUANTITY    = ol_quantity;
             ordstat->item[i].s_OL_AMOUNT      = ol_amount;
             put_convert_date_in_ordstat(&ol_delivery_date, ordstat-
>item[i].s_OL_DELIVERY_D);
            }
       if (code != NO_MORE_ROWS || deadlock)
            return FAIL;
       }
ordstat->s_ol_cnt = i;

     if (dbresults(dbproc) != SUCCEED || deadlock)
       {
        fprintf(stderr, "dbresults() failed for ordstat_end()\n");
        return FAIL;
       }
     else
       {
        /* Used to print the return variables from the stored proc. */
        /* Not using after that the dbbind() cause you won't get results. */
        /* dbprrow(dbproc); */

        dbbind(dbproc,1, INTBIND, 0, &ordstat->s_C_ID);
        dbbind(dbproc,2, NTBSTRINGBIND, sizeof(ordstat->s_C_LAST),
                                  ordstat->s_C_LAST);
        dbbind(dbproc,3, NTBSTRINGBIND, sizeof(ordstat->s_C_FIRST),
```

```
                                ordstat->s_C_FIRST);
        dbbind(dbproc,4, NTBSTRINGBIND, sizeof(ordstat->s_C_MIDDLE),
                                ordstat->s_C_MIDDLE);
        dbbind(dbproc,5, FLT8BIND, 0, &ordstat->s_C_BALANCE);
        dbbind(dbproc,6, INTBIND, 0, &ordstat->s_O_ID);
        dbbind(dbproc,7, DATETIMEBIND, 0, &o_entry_date);
        dbbind(dbproc,8, SMALLBIND, 0, &ordstat->s_O_CARRIER_ID);

        if (dbnextrow(dbproc) != REG_ROW || deadlock)
          return FAIL;

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

        put_convert2_date_in_ordstat(&o_entry_date, ordstat->s_O_ENTRY_D);
       }
     return SUCCEED;
    }
void put_convert_date_in_ordstat(sybase_date, datetime)
     DBDATETIME  *sybase_date;
     char        *datetime;
    {
     DBDATEREC   daterec;

     dbdatecrack(NULL, &daterec, sybase_date);

     sprintf(datetime,"%02d-%02d-%04d",
           daterec.datedmonth, daterec.datemonth+1, daterec.dateyear);
    }
void put_convert2_date_in_ordstat(sybase_date, datetime)
     DBDATETIME  *sybase_date;
     char        *datetime;
    {
     DBDATEREC   daterec;

     dbdatecrack(NULL, &daterec, sybase_date);

     sprintf(datetime,"%02d-%02d-%04d %02d:%02d:%02d",
           daterec.datedmonth, daterec.datemonth+1, daterec.dateyear,
           daterec.datehour, daterec.dateminute, daterec.datesecond);
    }
void sleep_before_retry()
    {
     sleep(rand()%3+1);
    }
int err_handler(dbproc, severity, errno, oserr)
     DBPROCESS   *dbproc;
     int         severity, errno, oserr;
    {
     fprintf(stderr, "DB-LIBRARY Error %d:", errno);
     display_xction(dberrstr(errno));

     if (oserr != DBNOERR)
       {
        fprintf(stderr, "AIX Error :");
        display_xction(dboserrstr(oserr));
       }
     dbcancel(dbproc);
     dbcmd(dbproc, "rollback transaction");
     dbsqlexec(dbproc);

     while (dbresults(dbproc) != SUCCEED)
          dbcanquery(dbproc);

     ordstat->s_transtatus = FATAL_SQL_ERROR;
     tpreturn(TPSUCCESS, 0, ordstat, sizeof(struct ordstat_struct), 0);
    }
int msg_handler(dbproc, msgno, msgstate, severity, msgtext, servername,
procname, line)
     DBPROCESS  *dbproc;
     int        msgno, msgstate, severity;
     char       *msgtext, *servername, *procname;
     int        line;
    {
     TPSVCINFO   *msg;
     /* Changing database messages */
     if (msgno == CONTEXT_SET || msgno == LANGUAGE_SET ||
         msgno == CHARACTER_SET)
       return FAIL;

     fprintf(stderr, "msg no %d", msgno);
     display_xction(msgtext);

     if (msgno == ABORT_ERROR)
       return FAIL;

     if (msgno == 1205)
       {
         display_xction(msgtext);
         deadlock = 1;
         return FAIL;
       }

     if (msg == NULL)
       exit(1);
     else
       {
        dbcancel(dbproc);
        dbcmd(dbproc, "rollback transaction");
        dbsqlexec(dbproc);

        while (dbresults(dbproc) != SUCCEED)
             dbcanquery(dbproc);

        ordstat->s_transtatus = FATAL_SQL_ERROR;
        tpreturn(TPSUCCESS, 0, ordstat, sizeof(struct ordstat_struct), 0);
       }
    }
void display_xction(msg)
     char *msg;
    {
     int i;

     fprintf(stderr, "%s ORDSTAT\n", msg);

     fprintf(stderr, "wid = %d, did = %d, cid = %d\n",
                                ordstat->s_W_ID,
                                 ordstat->s_D_ID,
```

```
                                ordstat->s_C_ID);
  fprintf(stderr, "clast = %s\n", ordstat->s_C_LAST);
 }
```

payment.c

```
/*****************************************************************************/
/*       FILE : payment.c                                                    */
/*****************************************************************************/
/*****************************************************************************/
/*                              INCLUDES FILES                               */
/*****************************************************************************/
#include <atmi.h>             /* Tuxedo header file */
#include <sybfront.h>                 /* Sybase include file */
#include <sybdb.h>            /* Sybase include file */
#include <cspublic.h>                 /* Sybase include file */
#include <syberror.h>                 /* Sybase include file */
#include <stdio.h>
#include "syb_tpcc.h"
/*****************************************************************************/
/*                              CONSTANTS                                    */
/*****************************************************************************/
#define TRAN_OK               0
#define FATAL_SQL_ERROR               -1
/*****************************************************************************/
/*                              STRUCTURES                                   */
/*****************************************************************************/
struct payment_struct
  {
   DBSMALLINT     s_W_ID;
   DBTINYINT        s_D_ID;
   DBINT      s_C_ID;
   DBTINYINT        s_C_D_ID;
   DBSMALLINT     s_C_W_ID;
   short        s_transtatus;
   DBFLT8    s_H_AMOUNT;
   char         s_H_DATE[20];
   char         s_W_STREET_1[21];
   char         s_W_STREET_2[21];
   char         s_W_CITY[21];
   char         s_W_STATE[3];
   char         s_W_ZIP[10];
   char         s_D_STREET_1[21];
   char         s_D_STREET_2[21];
   char         s_D_CITY[21];
   char         s_D_STATE[3];
   char         s_D_ZIP[10];
   char         s_C_FIRST[17];
   char         s_C_MIDDLE[3];
   char         s_C_LAST[17];
   char         s_C_STREET_1[21];
   char         s_C_STREET_2[21];
   char         s_C_CITY[21];
   char         s_C_STATE[3];
   char         s_C_ZIP[10];
   char         s_C_PHONE[17];
   char         s_C_SINCE[20];
   char         s_C_CREDIT[3];
   DBFLT8    s_C_CREDIT_LIM;
   DBREAL    s_C_DISCOUNT;
   DBFLT8    s_C_BALANCE;
   char         s_C_DATA[201];
  } *payment;
/*****************************************************************************/
/* FUNCTION : tpsvrinit()                                                    */
/* GOAL  : Initialize the sybase environnement with opening database and */
/*           setting transactions characteristics.                           */
/*           Moreover it is used first time when TUXEDO server is booted.    */
/*****************************************************************************/
tpsvrinit (argc, argv)
  int   argc;
  char *argv[];
 {
  /* Install the error and message handler */
  dberrhandle(err_handler);
  dbmsghandle(msg_handler);

  login = dblogin();
  DBSETLUSER(login, "sa");
  DBSETLPWD(login, "syBase");
  DBSETLPACKET(login, 4096);
  DBSETLCHARSET(login, "iso_1");
  /* Establish connection to Sybase */
  if ((dbproc = dbopen(login, NULL)) == NULL)
  {
        fprintf(stderr, "Fatal:Could not open connection\n");
        exit(-1);
  }
  dbloginfree(login); /* release the login record */
  dbuse(dbproc, "tpcc");
 }
/*****************************************************************************/
/* FUNCTION : PAYMENT_DSQL()                                                 */
/* GOAL     : It the TUXEDO server design for serving the payement         */
/*           transaction called by the TUXEDO client via a tpcall() call.    */
/*           It will send back the result of the transaction to the TUXEDO   */
/*           client by using the tpreturn() call.                            */
/*****************************************************************************/
PAYMENT_DSQL(msg)
  TPSVCINFO *msg;         /* Tuxedo buffer message structure */
 {
  int             try   = 0;
  DBINT       test_id;
  /* Getting the data area from the TUXEDO buffer message. */
  payment = (struct payment_struct *) msg->data;

  test_id = payment->s_C_ID;
  /* Repeat until we give up trying */
  for (try = 0; try < MaxTries; try++)
     {
       if (try > 0)
          display_xction("repeating");

      if (test_id)
        {
          if (payment_byid_begin() != SUCCEED)
            {
             dbcancel(dbproc);
```

```
            fprintf(stderr, "Error in payment_byid_begin()\n");
            sleep_before_retry();
            continue;
             }
         }
         else
         {
            if (payment_byname_begin() != SUCCEED)
              {
             dbcancel(dbproc);
             fprintf(stderr, "Error in payment_byname_begin()\n");
             sleep_before_retry();
             continue;
              }
         }
        if (payment_end() != SUCCEED)
           {
          dbcancel(dbproc);
          fprintf(stderr, "Error in payment_end()\n");
          sleep_before_retry();
          continue;
          }
        break;
       }
    if (try >= MaxTries)
      {
       display_xction("failed");
       payment->s_transtatus = FATAL_SQL_ERROR;
      }
    else
      {
       payment->s_transtatus = TRAN_OK;
      }
    tpreturn(TPSUCCESS, 0, payment, sizeof(struct payment_struct), 0);
   }
int payment_byid_begin()
  {
   double h_amount_cents;

   deadlock = 0;
   h_amount_cents = payment->s_H_AMOUNT * 100;
   dbrpcinit(dbproc, "payment_byid", 0);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &payment->s_W_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &payment->s_C_W_ID);
   dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &h_amount_cents);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &payment->s_D_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &payment->s_C_D_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &payment->s_C_ID);
   return(dbrpcsend(dbproc) == SUCCEED ? SUCCEED : FAIL);
  }
int payment_byname_begin()
  {
   double h_amount_cents;

   deadlock = 0;
   h_amount_cents = payment->s_H_AMOUNT * 100;
   dbrpcinit(dbproc, "payment_byname", 0);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &payment->s_W_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &payment->s_C_W_ID);
   dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &h_amount_cents);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &payment->s_D_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &payment->s_C_D_ID);
   dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(payment->s_C_LAST),
                                payment->s_C_LAST);
   return(dbrpcsend(dbproc) == SUCCEED ? SUCCEED : FAIL);
  }
int payment_end()
  {
   DBDATETIME   c_since,
                h_date;

   if (dbsqlok(dbproc) != SUCCEED)
      {
       fprintf(stderr, "dbsqlok() failed for payment_end()\n");
       return FAIL;
      }
   if (dbresults(dbproc) != SUCCEED || deadlock)
      {
       fprintf(stderr, "dbresults() failed for payment_end()\n");
       return FAIL;
      }
   else
      {
       dbbind(dbproc,1, INTBIND, 0, &payment->s_C_ID);
       dbbind(dbproc,2, NTBSTRINGBIND, sizeof(payment->s_C_LAST),
                                payment->s_C_LAST);
       dbbind(dbproc,  3, DATETIMEBIND,  0, &h_date);
       dbbind(dbproc,  4, NTBSTRINGBIND, sizeof(payment->s_W_STREET_1),
payment->s_W_STREET_1);
       dbbind(dbproc,  5, NTBSTRINGBIND, sizeof(payment->s_W_STREET_2),
payment->s_W_STREET_2);
       dbbind(dbproc,  6, NTBSTRINGBIND, sizeof(payment->s_W_CITY),
                                payment->s_W_CITY);
       dbbind(dbproc,  7, NTBSTRINGBIND, sizeof(payment->s_W_STATE),
                                payment->s_W_STATE);
       dbbind(dbproc,  8, NTBSTRINGBIND, sizeof(payment->s_W_ZIP),
                                payment->s_W_ZIP);

       dbbind(dbproc,  9, NTBSTRINGBIND, sizeof(payment->s_D_STREET_1),
payment->s_D_STREET_1);
       dbbind(dbproc, 10, NTBSTRINGBIND, sizeof(payment->s_D_STREET_2),
payment->s_D_STREET_2);
       dbbind(dbproc, 11, NTBSTRINGBIND, sizeof(payment->s_D_CITY),
payment->s_D_CITY);
       dbbind(dbproc, 12, NTBSTRINGBIND, sizeof(payment->s_D_STATE),
payment->s_D_STATE);
       dbbind(dbproc, 13, NTBSTRINGBIND, sizeof(payment->s_D_ZIP),
payment->s_D_ZIP);

       dbbind(dbproc, 14, NTBSTRINGBIND, sizeof(payment->s_C_FIRST),
payment->s_C_FIRST);
       dbbind(dbproc, 15, NTBSTRINGBIND, sizeof(payment->s_C_MIDDLE),
payment->s_C_MIDDLE);
       dbbind(dbproc, 16, NTBSTRINGBIND, sizeof(payment->s_C_STREET_1),
payment->s_C_STREET_1);
       dbbind(dbproc, 17, NTBSTRINGBIND, sizeof(payment->s_C_STREET_2),
payment->s_C_STREET_2);
```

```
      dbbind(dbproc, 18, NTBSTRINGBIND, sizeof(payment->s_C_CITY),
payment->s_C_CITY);
      dbbind(dbproc, 19, NTBSTRINGBIND, sizeof(payment->s_C_STATE),
payment->s_C_STATE);
      dbbind(dbproc, 20, NTBSTRINGBIND, sizeof(payment->s_C_ZIP),
payment->s_C_ZIP);
      dbbind(dbproc, 21, NTBSTRINGBIND, sizeof(payment->s_C_PHONE),
payment->s_C_PHONE);
      dbbind(dbproc, 22, DATETIMEBIND, 0, &c_since);
      dbbind(dbproc, 23, NTBSTRINGBIND, sizeof(payment->s_C_CREDIT),
payment->s_C_CREDIT);
      dbbind(dbproc, 24, FLT8BIND,     0, &payment->s_C_CREDIT_LIM);
      dbbind(dbproc, 25, FLT8BIND,    0, &payment->s_C_DISCOUNT);
      dbbind(dbproc, 26, FLT8BIND,     0, &payment->s_C_BALANCE);
      dbbind(dbproc, 27, NTBSTRINGBIND, sizeof(payment->s_C_DATA),
payment->s_C_DATA);

      if (dbnextrow(dbproc) != REG_ROW || deadlock)
        return FAIL;

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

      payment->s_C_CREDIT_LIM /= 100.0;
      payment->s_C_BALANCE /= 100.0;

      put_convert_date_in_payment(&h_date, payment->s_H_DATE);
      put_convert_date_in_payment(&c_since, payment->s_C_SINCE);
     }
   return SUCCEED;
 }
void put_convert_date_in_payment(sybase_date, datetime)
    DBDATETIME  *sybase_date;
    char        *datetime;
  {
    DBDATEREC   daterec;

    dbdatecrack(NULL, &daterec, sybase_date);

    sprintf(datetime,"%02d-%02d-%04d %02d:%02d:%02d",
          daterec.datedmonth, daterec.datemonth+1, daterec.dateyear,
          daterec.datehour, daterec.dateminute, daterec.datesecond);
  }
void sleep_before_retry()
  {
    sleep(rand()%3+1);
  }
int err_handler(dbproc, severity, errno, oserr)
    DBPROCESS   *dbproc;
    int         severity, errno, oserr;
  {
    TPSVCINFO   *msg;

    fprintf(stderr, "DB-LIBRARY Error %d:", errno);
    display_xction(dberrstr(errno));

    if (oserr != DBNOERR)
      {
       fprintf(stderr, "AIX Error :");
       display_xction(dboserrstr(oserr));
      }
    if (msg == NULL)
      exit(1);
    else
      {
       dbcancel(dbproc);
       dbcmd(dbproc, "rollback transaction");
       dbsqlexec(dbproc);

       while (dbresults(dbproc) != SUCCEED)
            dbcanquery(dbproc);

       payment->s_transtatus = FATAL_SQL_ERROR;
       tpreturn(TPSUCCESS, 0, payment, sizeof(struct payment_struct), 0);
      }
  }
int msg_handler(dbproc, msgno, msgstate, severity, msgtext, servername,
procname, line)
    DBPROCESS  *dbproc;
    int        msgno, msgstate, severity;
    char       *msgtext, *servername, *procname;
    int        line;
   {
    TPSVCINFO   *msg;
    /* Changing database messages */
    if (msgno == CONTEXT_SET || msgno == LANGUAGE_SET ||
        msgno == CHARACTER_SET)
      return FAIL;

    fprintf(stderr, "msg no %d", msgno);
    display_xction(msgtext);

    if (msgno == ABORT_ERROR)
      return FAIL;

    if (msgno == 1205)
      {
        display_xction(msgtext);
        deadlock = 1;
        return FAIL;
      }
    if (msg == NULL)
      exit(1);
    else
      {
       dbcancel(dbproc);
       dbcmd(dbproc, "rollback transaction");
       dbsqlexec(dbproc);

       while (dbresults(dbproc) != SUCCEED)
            dbcanquery(dbproc);

       payment->s_transtatus = FATAL_SQL_ERROR;
       tpreturn(TPSUCCESS, 0, payment, sizeof(struct payment_struct), 0);
      }
  }
void display_xction(msg)
    char *msg;
```

```
{
 int i;

 fprintf(stderr, "%s PAYMENT\n", msg);

 fprintf(stderr, "wid = %d, cwid = %d, did = %d, cdid = %d, cid = %d\n",
                                  payment->s_W_ID,
                                    payment->s_C_W_ID,
                                  payment->s_D_ID,
                                   payment->s_C_D_ID,
                                   payment->s_C_ID);
 fprintf(stderr, "clast = %s, cbalance = %f\n", payment->s_C_LAST,
                         payment->s_C_BALANCE);
```

The source code for the process to define, create and populate the SQL server 11 TPC-C database is included in this appendix

B.1 Init Create

devcreate.txt

```
disk init
  name = 'mcache',
  physname = '/dev/rcache_620w',
  vdevno = 2,
  size = 162816
go
disk init
  name = 'mdefault',
  physname = '/dev/rdef_620w',
  vdevno = 3,
  size = 143360
go
disk init
  name = 'tpcc_log1',
  physname = '/dev/rlog1',
  vdevno = 4,
  size = 919552
go
disk init
  name = 'tpcc_log2',
  physname = '/dev/rlog2',
  vdevno = 5,
  size = 919552
go
disk init
  name = 'tpcc_log3',
  physname = '/dev/rlog3',
  vdevno = 6,
  size = 919552
go
disk init
  name = 'tpcc_log4',
  physname = '/dev/rlog4',
  vdevno = 7,
  size = 919552
go
disk init
  name = 'orders1',
  physname = '/dev/rord1',
  vdevno = 8,
  size = 529408
go
disk init
  name = 'neworder1',
  physname = '/dev/rnord1',
  vdevno = 9,
  size = 164864
go
disk init
  name = 'history',
  physname = '/dev/rhist1',
  vdevno = 10,
  size = 793600
go
disk init
  name = 'ordl1',
  physname = '/dev/rordl1',
  vdevno = 11,
  size = 857088
go
disk init
  name = 'ordl2',
  physname = '/dev/rordl2',
  vdevno = 12,
  size = 857088
go
disk init
  name = 'ordl3',
  physname = '/dev/rordl3',
  vdevno = 13,
  size = 857088
go
disk init
  name = 'ordl4',
  physname = '/dev/rordl4',
  vdevno = 14,
  size = 857088
go
disk init
  name = 'ordl5',
  physname = '/dev/rordl5',
  vdevno = 15,
  size = 857088
go
disk init
  name = 'ordl6',
  physname = '/dev/rordl6',
  vdevno = 16,
  size = 857088
go
disk init
  name = 'ordl7',
  physname = '/dev/rordl7',
  vdevno = 17,
  size = 857088
go
disk init
  name = 'ordl8',
  physname = '/dev/rordl8',
  vdevno = 18,
  size = 857088
go
```

```
disk init
  name = 'ordl9',
  physname = '/dev/rordl9',
  vdevno = 19,
  size = 857088
go
disk init
  name = 'ordl10',
  physname = '/dev/rordl10',
  vdevno = 20,
  size = 857088
go
disk init
  name = 'stock1',
  physname = '/dev/rstk1',
  vdevno = 21,
  size = 182272
go
disk init
  name = 'stock2',
  physname = '/dev/rstk2',
  vdevno = 22,
  size = 182272
go
disk init
  name = 'stock3',
  physname = '/dev/rstk3',
  vdevno = 23,
  size = 182272
go
disk init
  name = 'stock4',
  physname = '/dev/rstk4',
  vdevno = 24,
  size = 182272
go
disk init
  name = 'stock5',
  physname = '/dev/rstk5',
  vdevno = 25,
  size = 182272
go
disk init
  name = 'stock6',
  physname = '/dev/rstk6',
  vdevno = 26,
  size = 182272
go
disk init
  name = 'stock7',
  physname = '/dev/rstk7',
  vdevno = 27,
  size = 182272
go
disk init
  name = 'stock8',
  physname = '/dev/rstk8',
  vdevno = 28,
  size = 182272
go
disk init
  name = 'stock9',
  physname = '/dev/rstk9',
  vdevno = 29,
  size = 182272
go
disk init
  name = 'stock10',
  physname = '/dev/rstk10',
  vdevno = 30,
  size = 182272
go
disk init
  name = 'stock11',
  physname = '/dev/rstk11',
  vdevno = 31,
  size = 182272
go
disk init
  name = 'stock12',
  physname = '/dev/rstk12',
  vdevno = 32,
  size = 182272
go
disk init
  name = 'stock13',
  physname = '/dev/rstk13',
  vdevno = 33,
  size = 182272
go
disk init
  name = 'stock14',
  physname = '/dev/rstk14',
  vdevno = 34,
  size = 182272
go
disk init
  name = 'stock15',
  physname = '/dev/rstk15',
  vdevno = 35,
  size = 182272
go
disk init
  name = 'stock16',
  physname = '/dev/rstk16',
  vdevno = 36,
  size = 182272
go
disk init
  name = 'stock17',
  physname = '/dev/rstk17',
  vdevno = 37,
  size = 182272
go
disk init
  name = 'stock18',
  physname = '/dev/rstk18',
  vdevno = 38,
```

```
  size = 182272
go
disk init
  name = 'stock19',
  physname = '/dev/rstk19',
  vdevno = 39,
  size = 182272
go
disk init
  name = 'stock20',
  physname = '/dev/rstk20',
  vdevno = 40,
  size = 182272
go
disk init
  name = 'stock21',
  physname = '/dev/rstk21',
  vdevno = 41,
  size = 182272
go
disk init
  name = 'stock22',
  physname = '/dev/rstk22',
  vdevno = 42,
  size = 182272
go
disk init
  name = 'stock23',
  physname = '/dev/rstk23',
  vdevno = 43,
  size = 182272
go
disk init
  name = 'stock24',
  physname = '/dev/rstk24',
  vdevno = 44,
  size = 182272
go
disk init
  name = 'stock25',
  physname = '/dev/rstk25',
  vdevno = 45,
  size = 182272
go
disk init
  name = 'stock26',
  physname = '/dev/rstk26',
  vdevno = 46,
  size = 182272
go
disk init
  name = 'stock27',
  physname = '/dev/rstk27',
  vdevno = 47,
  size = 182272
go
disk init
  name = 'stock28',
  physname = '/dev/rstk28',
  vdevno = 48,
  size = 182272
go
disk init
  name = 'stock29',
  physname = '/dev/rstk29',
  vdevno = 49,
  size = 182272
go
disk init
  name = 'stock30',
  physname = '/dev/rstk30',
  vdevno = 50,
  size = 182272
go
disk init
  name = 'stock31',
  physname = '/dev/rstk31',
  vdevno = 51,
  size = 182272
go
disk init
  name = 'stock32',
  physname = '/dev/rstk32',
  vdevno = 52,
  size = 182272
go
disk init
  name = 'stock33',
  physname = '/dev/rstk33',
  vdevno = 53,
  size = 182272
go
disk init
  name = 'stock34',
  physname = '/dev/rstk34',
  vdevno = 54,
  size = 182272
go
disk init
  name = 'stock35',
  physname = '/dev/rstk35',
  vdevno = 55,
  size = 182272
go
disk init
  name = 'stock36',
  physname = '/dev/rstk36',
  vdevno = 56,
  size = 182272
go
disk init
  name = 'stock37',
  physname = '/dev/rstk37',
  vdevno = 57,
  size = 182272
go
disk init
  name = 'stock38',
```

```
  physname = '/dev/rstk38',
  vdevno = 58,
  size = 182272
go
disk init
  name = 'stock39',
  physname = '/dev/rstk39',
  vdevno = 59,
  size = 182272
go
disk init
  name = 'stock40',
  physname = '/dev/rstk40',
  vdevno = 60,
  size = 182272
go
disk init
  name = 'stock41',
  physname = '/dev/rstk41',
  vdevno = 61,
  size = 182272
go
disk init
  name = 'stock42',
  physname = '/dev/rstk42',
  vdevno = 62,
  size = 182272
go
disk init
  name = 'stock43',
  physname = '/dev/rstk43',
  vdevno = 63,
  size = 182272
go
disk init
  name = 'stock44',
  physname = '/dev/rstk44',
  vdevno = 64,
  size = 182272
go
disk init
  name = 'stock45',
  physname = '/dev/rstk45',
  vdevno = 65,
  size = 182272
go
disk init
  name = 'stock46',
  physname = '/dev/rstk46',
  vdevno = 66,
  size = 182272
go
disk init
  name = 'stock47',
  physname = '/dev/rstk47',
  vdevno = 67,
  size = 182272
go
disk init
  name = 'stock48',
  physname = '/dev/rstk48',
  vdevno = 68,
  size = 182272
go
disk init
  name = 'stock49',
  physname = '/dev/rstk49',
  vdevno = 69,
  size = 182272
go
disk init
  name = 'stock50',
  physname = '/dev/rstk50',
  vdevno = 70,
  size = 182272
go
disk init
  name = 'stock51',
  physname = '/dev/rstk51',
  vdevno = 71,
  size = 182272
go
disk init
  name = 'stock52',
  physname = '/dev/rstk52',
  vdevno = 72,
  size = 182272
go
disk init
  name = 'stock53',
  physname = '/dev/rstk53',
  vdevno = 73,
  size = 182272
go
disk init
  name = 'stock54',
  physname = '/dev/rstk54',
  vdevno = 74,
  size = 182272
go
disk init
  name = 'stock55',
  physname = '/dev/rstk55',
  vdevno = 75,
  size = 182272
go
disk init
  name = 'stock56',
  physname = '/dev/rstk56',
  vdevno = 76,
  size = 182272
go
disk init
  name = 'stock57',
  physname = '/dev/rstk57',
  vdevno = 77,
  size = 182272
go
```

```
disk init
  name = 'stock58',
  physname = '/dev/rstk58',
  vdevno = 78,
  size = 182272
go
disk init
  name = 'stock59',
  physname = '/dev/rstk59',
  vdevno = 79,
  size = 182272
go
disk init
  name = 'stock60',
  physname = '/dev/rstk60',
  vdevno = 80,
  size = 182272
go
disk init
  name = 'customer1',
  physname = '/dev/rcust1',
  vdevno = 81,
  size = 237056
go
disk init
  name = 'customer2',
  physname = '/dev/rcust2',
  vdevno = 82,
  size = 237056
go
disk init
  name = 'customer3',
  physname = '/dev/rcust3',
  vdevno = 83,
  size = 237056
go
disk init
  name = 'customer4',
  physname = '/dev/rcust4',
  vdevno = 84,
  size = 237056
go
disk init
  name = 'customer5',
  physname = '/dev/rcust5',
  vdevno = 85,
  size = 237056
go
disk init
  name = 'customer6',
  physname = '/dev/rcust6',
  vdevno = 86,
  size = 237056
go
disk init
  name = 'customer7',
  physname = '/dev/rcust7',
  vdevno = 87,
  size = 237056
go
disk init
  name = 'customer8',
  physname = '/dev/rcust8',
  vdevno = 88,
  size = 237056
go
disk init
  name = 'customer9',
  physname = '/dev/rcust9',
  vdevno = 89,
  size = 237056
go
disk init
  name = 'customer10',
  physname = '/dev/rcust10',
  vdevno = 90,
  size = 237056
go
disk init
  name = 'customer11',
  physname = '/dev/rcust11',
  vdevno = 91,
  size = 237056
go
disk init
  name = 'customer12',
  physname = '/dev/rcust12',
  vdevno = 92,
  size = 237056
go
disk init
  name = 'customer13',
  physname = '/dev/rcust13',
  vdevno = 93,
  size = 237056
go
disk init
  name = 'customer14',
  physname = '/dev/rcust14',
  vdevno = 94,
  size = 237056
go
disk init
  name = 'customer15',
  physname = '/dev/rcust15',
  vdevno = 95,
  size = 237056
go
disk init
  name = 'customer16',
  physname = '/dev/rcust16',
  vdevno = 96,
  size = 237056
go
disk init
  name = 'customer17',
  physname = '/dev/rcust17',
  vdevno = 97,
```

```
  size = 237056
go
disk init
  name = 'customer18',
  physname = '/dev/rcust18',
  vdevno = 98,
  size = 237056
go
disk init
  name = 'customer19',
  physname = '/dev/rcust19',
  vdevno = 99,
  size = 237056
go
disk init
  name = 'customer20',
  physname = '/dev/rcust20',
  vdevno = 100,
  size = 237056
go
disk init
  name = 'customer21',
  physname = '/dev/rcust21',
  vdevno = 101,
  size = 237056
go
disk init
  name = 'customer22',
  physname = '/dev/rcust22',
  vdevno = 102,
  size = 237056
go
disk init
  name = 'customer23',
  physname = '/dev/rcust23',
  vdevno = 103,
  size = 237056
go
disk init
  name = 'customer24',
  physname = '/dev/rcust24',
  vdevno = 104,
  size = 237056
go
disk init
  name = 'customer25',
  physname = '/dev/rcust25',
  vdevno = 105,
  size = 237056
go
disk init
  name = 'customer26',
  physname = '/dev/rcust26',
  vdevno = 106,
  size = 237056
go
disk init
  name = 'customer27',
  physname = '/dev/rcust27',
  vdevno = 107,
  size = 237056
go
disk init
  name = 'customer28',
  physname = '/dev/rcust28',
  vdevno = 108,
  size = 237056
go
disk init
  name = 'customer29',
  physname = '/dev/rcust29',
  vdevno = 109,
  size = 237056
go
disk init
  name = 'customer30',
  physname = '/dev/rcust30',
  vdevno = 110,
  size = 237056
go
disk init
  name = 'c_index1',
  physname = '/dev/ricust1',
  vdevno = 111,
  size = 135168
go
disk init
  name = 'c_index2',
  physname = '/dev/ricust2',
  vdevno = 112,
  size = 135168
go
disk init
  name = 'c_index3',
  physname = '/dev/ricust3',
  vdevno = 113,
  size = 135168
go
disk init
  name = 'c_index4',
  physname = '/dev/ricust4',
  vdevno = 114,
  size = 135168
go
disk init
  name = 'c_index5',
  physname = '/dev/ricust5',
  vdevno = 115,
  size = 135168
go
create database tpcc
on master = 520, mcache = 318, mdefault = 280, orders1 = 1034, neworder1 =
322, history = 1550, ordl1 = 1674, ordl2 = 1674, ordl3 = 1674, ordl4 = 1674,
ordl5 = 1674, ordl6 = 1674, ordl7 = 1674, ordl8 = 1674, ordl9 = 1674, ordl10 =
1674, stock1 = 356, stock2 = 356, stock3 = 356, stock4 = 356, stock5 = 356,
stock6 = 356, stock7 = 356, stock8 = 356, stock9 = 356, stock10 = 356,
stock11 = 356, stock12 = 356, stock13 = 356, stock14 = 356, stock15 = 356,
stock16 = 356, stock17 = 356, stock18 = 356, stock19 = 356, stock20 = 356,
```

```
stock21 = 356, stock22 = 356, stock23 = 356, stock24 = 356, stock25 = 356,
stock26 = 356, stock27 = 356, stock28 = 356, stock29 = 356, stock30 = 356,
stock31 = 356, stock32 = 356, stock33 = 356, stock34 = 356, stock35 = 356,
stock36 = 356, stock37 = 356, stock38 = 356, stock39 = 356, stock40 = 356,
stock41 = 356, stock42 = 356, stock43 = 356, stock44 = 356, stock45 = 356,
stock46 = 356, stock47 = 356, stock48 = 356, stock49 = 356, stock50 = 356,
stock51 = 356, stock52 = 356, stock53 = 356, stock54 = 356, stock55 = 356,
stock56 = 356, stock57 = 356, stock58 = 356, stock59 = 356, stock60 = 356,
customer1 = 463, customer2 = 463, customer3 = 463, customer4 = 463,
customer5 = 463, customer6 = 463, customer7 = 463, customer8 = 463,
customer9 = 463, customer10 = 463, customer11 = 463, customer12 = 463,
customer13 = 463, customer14 = 463, customer15 = 463, customer16 = 463,
customer17 = 463, customer18 = 463, customer19 = 463, customer20 = 463,
customer21 = 463, customer22 = 463, customer23 = 463, customer24 = 463,
customer25 = 463, customer26 = 463, customer27 = 463, customer28 = 463,
customer29 = 463, customer30 = 463, c_index1 = 264, c_index2 = 264,
c_index3 = 264, c_index4 = 264, c_index5 = 264
log on tpcc_log4 = 1796
go
use tpcc
go
sp_addsegment Sc_index , tpcc , c_index1
go
sp_extendsegment Sc_index , tpcc ,  c_index2
go
sp_extendsegment Sc_index , tpcc ,  c_index3
go
sp_extendsegment Sc_index , tpcc ,  c_index4
go
sp_extendsegment Sc_index , tpcc ,  c_index5
go
sp_addsegment Scache , tpcc , mcache
go
sp_addsegment Scustomer , tpcc , customer1
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 ,  customer2
go
sp_extendsegment Scustomer , tpcc ,  customer20
go
sp_extendsegment Scustomer , tpcc ,  customer21
go
sp_extendsegment Scustomer , tpcc ,  customer22
go
sp_extendsegment Scustomer , tpcc ,  customer23
go
sp_extendsegment Scustomer , tpcc ,  customer24
go
sp_extendsegment Scustomer , tpcc ,  customer25
go
sp_extendsegment Scustomer , tpcc ,  customer26
go
sp_extendsegment Scustomer , tpcc ,  customer27
go
sp_extendsegment Scustomer , tpcc ,  customer28
go
sp_extendsegment Scustomer , tpcc ,  customer29
go
sp_extendsegment Scustomer , tpcc ,  customer3
go
sp_extendsegment Scustomer , tpcc ,  customer30
go
sp_extendsegment Scustomer , tpcc ,  customer4
go
sp_extendsegment Scustomer , tpcc ,  customer5
go
sp_extendsegment Scustomer , tpcc ,  customer6
go
sp_extendsegment Scustomer , tpcc ,  customer7
go
sp_extendsegment Scustomer , tpcc ,  customer8
go
sp_extendsegment Scustomer , tpcc ,  customer9
go
sp_addsegment Shistory , tpcc , history
go
sp_addsegment Snew_order , tpcc , neworder1
go
sp_addsegment Sorder_line , tpcc , ordl1
go
sp_extendsegment Sorder_line , tpcc ,  ordl10
go
sp_extendsegment Sorder_line , tpcc ,  ordl2
go
sp_extendsegment Sorder_line , tpcc ,  ordl3
go
sp_extendsegment Sorder_line , tpcc ,  ordl4
go
sp_extendsegment Sorder_line , tpcc ,  ordl5
go
sp_extendsegment Sorder_line , tpcc ,  ordl6
go
sp_extendsegment Sorder_line , tpcc ,  ordl7
go
sp_extendsegment Sorder_line , tpcc ,  ordl8
go
sp_extendsegment Sorder_line , tpcc ,  ordl9
go
sp_addsegment Sorders , tpcc , orders1
```

```
go
sp_addsegment Sstock , tpcc , stock1
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 ,  stock2
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 ,  stock3
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 ,  stock4
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_extendsegment Sstock , tpcc ,  stock45
go
sp_extendsegment Sstock , tpcc ,  stock46
go
sp_extendsegment Sstock , tpcc ,  stock47
go
sp_extendsegment Sstock , tpcc ,  stock48
go
sp_extendsegment Sstock , tpcc ,  stock49
go
sp_extendsegment Sstock , tpcc ,  stock5
go
sp_extendsegment Sstock , tpcc ,  stock50
go
sp_extendsegment Sstock , tpcc ,  stock51
go
sp_extendsegment Sstock , tpcc ,  stock52
go
sp_extendsegment Sstock , tpcc ,  stock53
go
sp_extendsegment Sstock , tpcc ,  stock54
go
sp_extendsegment Sstock , tpcc ,  stock55
go
sp_extendsegment Sstock , tpcc ,  stock56
go
sp_extendsegment Sstock , tpcc ,  stock57
go
sp_extendsegment Sstock , tpcc ,  stock58
go
sp_extendsegment Sstock , tpcc ,  stock59
go
sp_extendsegment Sstock , tpcc ,  stock6
go
sp_extendsegment Sstock , tpcc ,  stock60
go
sp_extendsegment Sstock , tpcc ,  stock7
go
sp_extendsegment Sstock , tpcc ,  stock8
```

```
go
sp_extendsegment Sstock , tpcc ,  stock9
go
sp_dropsegment 'default', tpcc , c_index1
go
sp_dropsegment 'system', tpcc , c_index1
go
sp_dropsegment 'default', tpcc , c_index2
go
sp_dropsegment 'system', tpcc , c_index2
go
sp_dropsegment 'default', tpcc , c_index3
go
sp_dropsegment 'system', tpcc , c_index3
go
sp_dropsegment 'default', tpcc , c_index4
go
sp_dropsegment 'system', tpcc , c_index4
go
sp_dropsegment 'default', tpcc , c_index5
go
sp_dropsegment 'system', tpcc , c_index5
go
sp_dropsegment 'default', tpcc , customer1
go
sp_dropsegment 'system', tpcc , customer1
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 , customer2
go
sp_dropsegment 'system', tpcc , customer2
go
sp_dropsegment 'default', tpcc , customer20
go
sp_dropsegment 'system', tpcc , customer20
go
sp_dropsegment 'default', tpcc , customer21
go
sp_dropsegment 'system', tpcc , customer21
go
sp_dropsegment 'default', tpcc , customer22
go
sp_dropsegment 'system', tpcc , customer22
go
sp_dropsegment 'default', tpcc , customer23
go
sp_dropsegment 'system', tpcc , customer23
go
sp_dropsegment 'default', tpcc , customer24
go
sp_dropsegment 'system', tpcc , customer24
go
sp_dropsegment 'default', tpcc , customer25
go
sp_dropsegment 'system', tpcc , customer25
go
sp_dropsegment 'default', tpcc , customer26
go
sp_dropsegment 'system', tpcc , customer26
go
sp_dropsegment 'default', tpcc , customer27
go
sp_dropsegment 'system', tpcc , customer27
go
sp_dropsegment 'default', tpcc , customer28
go
sp_dropsegment 'system', tpcc , customer28
go
sp_dropsegment 'default', tpcc , customer29
go
sp_dropsegment 'system', tpcc , customer29
go
sp_dropsegment 'default', tpcc , customer3
go
sp_dropsegment 'system', tpcc , customer3
go
sp_dropsegment 'default', tpcc , customer30
go
sp_dropsegment 'system', tpcc , customer30
```

```
go
sp_dropsegment 'default', tpcc , customer4
go
sp_dropsegment 'system', tpcc , customer4
go
sp_dropsegment 'default', tpcc , customer5
go
sp_dropsegment 'system', tpcc , customer5
go
sp_dropsegment 'default', tpcc , customer6
go
sp_dropsegment 'system', tpcc , customer6
go
sp_dropsegment 'default', tpcc , customer7
go
sp_dropsegment 'system', tpcc , customer7
go
sp_dropsegment 'default', tpcc , customer8
go
sp_dropsegment 'system', tpcc , customer8
go
sp_dropsegment 'default', tpcc , customer9
go
sp_dropsegment 'system', tpcc , customer9
go
sp_dropsegment 'default', tpcc , history
go
sp_dropsegment 'system', tpcc , history
go
sp_dropsegment 'default', tpcc , mcache
go
sp_dropsegment 'system', tpcc , mcache
go
sp_dropsegment 'default', tpcc , neworder1
go
sp_dropsegment 'system', tpcc , neworder1
go
sp_dropsegment 'default', tpcc , orders1
go
sp_dropsegment 'system', tpcc , orders1
go
sp_dropsegment 'default', tpcc , ordl1
go
sp_dropsegment 'system', tpcc , ordl1
go
sp_dropsegment 'default', tpcc , ordl10
go
sp_dropsegment 'system', tpcc , ordl10
go
sp_dropsegment 'default', tpcc , ordl2
go
sp_dropsegment 'system', tpcc , ordl2
go
sp_dropsegment 'default', tpcc , ordl3
go
sp_dropsegment 'system', tpcc , ordl3
go
sp_dropsegment 'default', tpcc , ordl4
go
sp_dropsegment 'system', tpcc , ordl4
go
sp_dropsegment 'default', tpcc , ordl5
go
sp_dropsegment 'system', tpcc , ordl5
go
sp_dropsegment 'default', tpcc , ordl6
go
sp_dropsegment 'system', tpcc , ordl6
go
sp_dropsegment 'default', tpcc , ordl7
go
sp_dropsegment 'system', tpcc , ordl7
go
sp_dropsegment 'default', tpcc , ordl8
go
sp_dropsegment 'system', tpcc , ordl8
go
sp_dropsegment 'default', tpcc , ordl9
go
sp_dropsegment 'system', tpcc , ordl9
go
sp_dropsegment 'default', tpcc , stock1
go
sp_dropsegment 'system', tpcc , stock1
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 , stock2
go
sp_dropsegment 'system', tpcc , stock2
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 , stock3
go
sp_dropsegment 'system', tpcc , stock3
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 , stock4
go
sp_dropsegment 'system', tpcc , stock4
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
sp_dropsegment 'default', tpcc , stock45
go
sp_dropsegment 'system', tpcc , stock45
go
sp_dropsegment 'default', tpcc , stock46
go
sp_dropsegment 'system', tpcc , stock46
go
sp_dropsegment 'default', tpcc , stock47
go
sp_dropsegment 'system', tpcc , stock47
go
sp_dropsegment 'default', tpcc , stock48
go
sp_dropsegment 'system', tpcc , stock48
go
sp_dropsegment 'default', tpcc , stock49
go
sp_dropsegment 'system', tpcc , stock49
go
sp_dropsegment 'default', tpcc , stock5
go
sp_dropsegment 'system', tpcc , stock5
go
sp_dropsegment 'default', tpcc , stock50
go
sp_dropsegment 'system', tpcc , stock50
go
sp_dropsegment 'default', tpcc , stock51
go
sp_dropsegment 'system', tpcc , stock51
go
sp_dropsegment 'default', tpcc , stock52
go
sp_dropsegment 'system', tpcc , stock52
go
sp_dropsegment 'default', tpcc , stock53
go
sp_dropsegment 'system', tpcc , stock53
go
sp_dropsegment 'default', tpcc , stock54
go
sp_dropsegment 'system', tpcc , stock54
go
sp_dropsegment 'default', tpcc , stock55
go
sp_dropsegment 'system', tpcc , stock55
go
sp_dropsegment 'default', tpcc , stock56
go
sp_dropsegment 'system', tpcc , stock56
go
sp_dropsegment 'default', tpcc , stock57
go
sp_dropsegment 'system', tpcc , stock57
go
sp_dropsegment 'default', tpcc , stock58
go
sp_dropsegment 'system', tpcc , stock58
go
sp_dropsegment 'default', tpcc , stock59
go
sp_dropsegment 'system', tpcc , stock59
go
sp_dropsegment 'default', tpcc , stock6
go
sp_dropsegment 'system', tpcc , stock6
go
sp_dropsegment 'default', tpcc , stock60
go
sp_dropsegment 'system', tpcc , stock60
go
sp_dropsegment 'default', tpcc , stock7
go
sp_dropsegment 'system', tpcc , stock7
go
sp_dropsegment 'default', tpcc , stock8
go
sp_dropsegment 'system', tpcc , stock8
go
use master
go
alter database tpcc
log on tpcc_log1 = 1796
go
use tpcc
go
sp_extendsegment logsegment, tpcc, tpcc_log1
go
use master
go
alter database tpcc
log on tpcc_log2 = 1796
go
use tpcc
go
sp_extendsegment logsegment, tpcc, tpcc_log2
go
use master
go
alter database tpcc
log on tpcc_log3 = 1796
go
use tpcc
go
sp_extendsegment logsegment, tpcc, tpcc_log3
go
use master
go
checkpoint
go
```

B.2 Cache Binding

devcreate.txt

```
#!/bin/sh -f
isql -Usa -P$PASSWORD << EOF
use master
go
sp_dboption tpcc, "single user", true
go
use tpcc
go
checkpoint
go
/*
** Cache c_log
*/
sp_bindcache "c_log", "tpcc", "syslogs"
go
/*
** Cache c_tinyhot
*/
sp_bindcache "c_tinyhot", "tpcc", "sysindexes"
go
sp_bindcache "c_tinyhot", "tpcc", "sysindexes", "sysindexes"
go
use master
go
sp_dboption tpcc, "single user", false
go
use tpcc
go
checkpoint
go
/*
** Cache c_tinyhot (continued)
*/
sp_bindcache "c_tinyhot", "tpcc", "item"
go
sp_bindcache "c_tinyhot", "tpcc", "item", "i_clu"
go
sp_bindcache "c_tinyhot", "tpcc", "warehouse"
go
sp_bindcache "c_tinyhot", "tpcc", "warehouse", "w_clu"
go
sp_bindcache "c_tinyhot", "tpcc", "district"
go
sp_bindcache "c_tinyhot", "tpcc", "district", "d_clu"
go
/*
** Cache c_no_ol
*/
sp_bindcache "c_no_ol", "tpcc", "new_order"
go
sp_bindcache "c_no_ol", "tpcc", "new_order", "no_clu"
go
sp_bindcache "c_no_ol", "tpcc", "order_line"
go
/*
** Cache c_ol_index
*/
sp_bindcache "c_ol_index", "tpcc", "order_line", "ol_clu"
go
/*
** Cache c_orders
*/
sp_bindcache "c_orders", "tpcc", "orders"
go
sp_bindcache "c_orders", "tpcc", "orders", "o_clu"
go
/*
** Cache c_stock_index
*/
sp_bindcache "c_stock_index", "tpcc", "stock", "s_clu"
go
/*
** Cache c_stock
*/
sp_bindcache "c_stock", "tpcc", "stock"
go
/*
** Cache c_customer
*/
sp_bindcache "c_customer", "tpcc", "customer"
go
/*
** 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
```

B.3 Data Load

bulk_sybase.c

```
/*************************************************************
Sybase Specific Routines
*************************************************************/
#include <stdio.h>
#include <sys/time.h>
#include <string.h>
#include "loader.h"
datetime(date)
DBDATETIME*date;
{
struct timeval time;
gettimeofday(&time, NULL);
date->dtdays = time.tv_sec / (60*60*24)+ (1970-1900)*365 + (1970-
1900)/4;
date->dttime = (time.tv_sec % (60*60*24))*300+
time.tv_usec*300/1000000;
}
/* define the type information for each field */
```

```
typedef struct
{
char *terminator;
int termlen;
int type;
} bind_parm;
bind_parm parm[MAX_T] =
{
/* COUNT */     {NULL,0,SYBINT4},   {NULL, 0, SYBINT4},
/* ID          */     {NULL,0,SYBINT4},   */   {NULL, 0, SYBINT4},
/* MONEY */     {NULL,0,SYBFTL8},  {NULL, 0, SYBFLT8},
/* FLOAT */     {NULL,0,SYBFTL8},   {NULL, 0, SYBFLT8},
/* TEXT */     {«  »,1,SYBCHAR}, {"",   1, SYBCHAR},
/* DATE */*     {NULL,0,SYBDATETIME},  {NULL, 0,
SYBDATETIME},
/* LOGICAL */*     {NULL,0,SYBINT4},    {NULL, 0,
SYBINT4}
};
#define MAXOPENS 10
DBPROCESS *dbproc[MAXOPENS];
int      count [MAXOPENS ];   count[MAXOPENS];
int bulk_open(database, table, password)
char database[];
char table[];
char password[];
{
LOGINREC *login;
int db;
/* make note we have established a connection */
for (db=0; db<MAXOPENS; db++)
if (dbproc[db] == NULL) break;
count[db] = 0;
/* Install an error and Message handler */
dbmsghandle(msg_handler);
dberrhandle(err_handler);
/* initialize dblib */
if (dbinit() != SUCCEED)
printf("Can't initialize the DB library\n");
/* allocate a login record and fill it in */
login = dblogin();
if (login == NULL)
printf("Can't allocate a login record.\n");
DBSETLUSER(login, "sa");
if(strlen(password) > 0)
DBSETLPWD(login, password);
DBSETLAPP(login, table);
BCP_SETL(login, TRUE);
/* Set Packet Size to 4096 */
DBSETLPACKET(login, 4096);
/* establish a connection with the server specified by
DSQUERY */
dbproc[db] = dbopen(login, NULL);
if (dbproc[db] == NULL)
/* select the database to use */
if (database != NULL)
if (dbuse(dbproc[db], database) != SUCCEED)
printf("Can't select database: %s\n", database);
/* release the login record */
dbloginfree(login);
/* prepare to do a bulk copy */
if (bcp_init(dbproc[db], table, NULL, NULL, DB_IN) !=
SUCCEED)
printf("Can't initialize the bulk copy to table %s\n",
table);
return db;
}
bulk_bind(db, column, name, address, type)
int db;
int column;
char name[];
void *address;
int type;
{
if (bcp_bind(dbproc[db], address, 0, -1,
parm[type].terminator,
parm[type].termlen, parm[type].type, column) != SUCCEED)
printf("Can't bind column %d to 0x%x, type=%d\n",
column,address,type);
}
bulk_null(db, column)
int db;
int column;
{
if (bcp_collen(dbproc[db], 0, column) != SUCCEED)
printf("Can't null column %d\n", column);
}
bulk_non_null(db, column)
int db;
int column;
{
if (bcp_collen(dbproc[db], -1, column) != SUCCEED)
printf("Can't non-null column %d\n", column);
}
bulk_load(db)
int db;
{
count[db]++;
if (bcp_sendrow(dbproc[db]) != SUCCEED)
printf("bulk_load: Can't load row\n");
if (count[db]%batch_size == 0 &&
(bcp_batch(dbproc[db]) == -1))
printf("bulk_load: Can't post rows\n");
if (count[db]%1000 == 0) write(1,".",1);
if (count[db]%50000 == 0) write(1,"\n",1);
}
bulk_close(db)
int db;
{
if (bcp_done(dbproc[db]) == -1)
printf("Problems completing the bulk copy.\n");
dbproc[db] = NULL;
if (count[db] >= 1000) write(1,"\n",1);
}
```

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:1.14/30/9119:47:32
**Standard error handler for RungenII and supporting code
**HMS [04/30/91]
*/
/* Required standard include files */
#include <stdio.h>
/* Required Sybase include files */
#include <sybfront.h>
#include <sybdb.h>
/* message numbers that we don't want to deal with */
#define   DUMB_MESSAGE  5701
#define   ABORT_ERROR     6104
int
err_handler(dbproc, severity, errno, oserr)
  DBPROCESS *dbproc;
  int severity;
  int errno;
  int oserr;
{
/* changing databases message */
if (errno == DUMB_MESSAGE || errno == ABORT_ERROR)
return(INT_CANCEL);
fprintf(stderr,"DB-LIBRARY Error:
\n\t%s\n",dberrstr(errno));
if (oserr != DBNOERR)
fprintf(stderr,"O/S Error: \n\t%s\n",dboserrstr(oserr));
/* exit on any error */
exit(-100);
}
int
msg_handler(dbproc,msgno,msgstate,severity,msgtext,servername,procn
ame,line)
DBPROCESS   *dbproc;
int    msgno;
int    msgstate;
int   severity;
char      *msgtext;
char           *servername;
char           *procname;
int            line;
{
/* changing database messages */
if (msgno == DUMB_MESSAGE || msgno == ABORT_ERROR || msgno
== 5703 || msgno == 5704)
return(SUCCEED);
/* Is this a deadlock message */
if (msgno == 1205)
{
/* Set the deadlock indicator */
*((DBBOOL *) dbgetuserdata(dbproc)) = TRUE;
/* Sleep a few seconds before going back */
sleep((unsigned) 2);
return(SUCCEED);
}
fprintf(stderr, "msg no %d -\n%s", msgno, msgtext);
/* exit on any error */
exit(-101);
}
```

load.c

```
typedef   unsigned long   BitVector;
#define WSZ  (sizeof(BitVector)*8)
/* For axposf use WAREBATCH of 144 */
#ifndef WAREBATCH
#define  WAREBATCH   960
#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;
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;
}
begin_warehouse_load()
{
inti = 1;
bulk_w = bulk_open("tpcc", "warehouse", password);
bulk_bind(bulk_w, i++, "w_id",   &w_id, ID_T);
bulk_bind(bulk_w, i++, "w_name", w_name, TEXT_T);
bulk_bind(bulk_w, i++, "w_street_1", w_street_1, TEXT_T);
bulk_bind(bulk_w, i++, "w_street_2", w_street_2, TEXT_T);
bulk_bind(bulk_w, i++, "w_city", w_city, TEXT_T);
bulk_bind(bulk_w, i++, "w_state", w_state, TEXT_T);
bulk_bind(bulk_w, i++, "w_zip", w_zip, TEXT_T);
bulk_bind(bulk_w, i++, "w_tax", &w_tax, FLOAT_T);
bulk_bind(bulk_w, i++, "w_ytd", &w_ytd, MONEY_T);
}
warehouse_load()
{
debug("Loading Warehouse %d\n", w_id);
bulk_load(bulk_w);
}
end_warehouse_load()
{
bulk_close(bulk_w);
}
/**********************************************************
District
**********************************************************/
ID d_id;
ID d_w_id;
TEXT d_name[10+1];
TEXT d_street_1[20+1];
TEXT d_street_2[20+1];
TEXT d_city[20+1];
TEXT d_state[2+1];
TEXT d_zip[9+1];
FLOAT d_tax;
MONEY d_ytd;
ID   d_next_o_id;
int bulk_d;
LoadDistrict(w1, w2)
ID w1, w2;
{
ID w_id;
begin_district_load();
for (w_id=w1; w_id<=w2; w_id++)
{
printf("Loading districts for warehouse %d\n", w_id);
d_w_id = w_id;
d_ytd = 30000.00 * 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;
}
begin_district_load()
{
inti = 1;
bulk_d = bulk_open("tpcc", "district", password);
bulk_bind(bulk_d, i++, "d_id", &d_id, ID_T);
bulk_bind(bulk_d, i++, "d_w_id", &d_w_id, ID_T);
bulk_bind(bulk_d, i++, "d_name", d_name, TEXT_T);
bulk_bind(bulk_d, i++, "d_street_1", d_street_1, TEXT_T);
bulk_bind(bulk_d, i++, "d_street_2", d_street_2, TEXT_T);
bulk_bind(bulk_d, i++, "d_city", d_city, TEXT_T);
bulk_bind(bulk_d, i++, "d_state", d_state, TEXT_T);
bulk_bind(bulk_d, i++, "d_zip", d_zip, TEXT_T);
bulk_bind(bulk_d, i++, "d_tax", &d_tax, FLOAT_T);
bulk_bind(bulk_d, i++, "d_ytd", &d_ytd, MONEY_T);
bulk_bind(bulk_d, i++, "d_next_o_id", &d_next_o_id, ID_T);
}
district_load()
{
debug("District %d  w_id=%d\n", d_id, d_w_id);
bulk_load(bulk_d);
}
end_district_load()
{
bulk_close(bulk_d);
}
/**********************************************************
Item
```

```
**********************************************************/
ID i_id;
ID i_im_id;
TEXT i_name[24+1];
MONEY i_price;
TEXT  i_data[50+1];
int bulk_i;
LoadItems()
{
  int perm[MAXITEMS+1];
int i, r, t;
printf("Loading items\n");
begin_item_load();
/* select exactly 10% of items to be labeled "original" */
RandomPermutation(perm, MAXITEMS);
/* do for each item */
for (i_id=1;   i_id <= MAXITEMS; i_id++)
{
/* Generate Item Data */
MakeAlphaString(14, 24, i_name);
i_price = RandomNumber(100,10000);
MakeAlphaString(26, 50, i_data);
if (perm[i_id] <= (MAXITEMS+9)/10)
Original(i_data);
/* Generate i_im_id for V 3.0 */
i_im_id = RandomNumber(1, 10000);
 item_load();
 }
end_item_load();
return;
}
begin_item_load()
{
inti = 1;
bulk_i = bulk_open("tpcc", "item", password);
/* bind the variables to the sybase columns */
bulk_bind(bulk_i, i++, "i_id", &i_id, ID_T);
bulk_bind(bulk_i, i++, "i_im_id", &i_im_id, ID_T);
bulk_bind(bulk_i, i++, "i_name", i_name, TEXT_T);
bulk_bind(bulk_i, i++, "i_price", &i_price, MONEY_T);
bulk_bind(bulk_i, i++, "i_data", i_data, TEXT_T);
}
item_load()
{
debug("i_id=%3d price=%5.2f data=%s\n",
   i_id,   i_price,   i_data);
bulk_load(bulk_i);
}
end_item_load()
{
bulk_close(bulk_i);
}
/**********************************************************
History
**********************************************************/
ID h_c_id;
ID h_c_d_id;
ID h_c_w_id;
ID h_d_id;
ID h_w_id;
DATE h_date;
MONEY h_amount;
TEXT h_data[24+1];
int bulk_h;
LoadHist(w1, w2)
ID w1, w2;
{
ID w_id;
ID d_id, c_id;
begin_history_load();
for (w_id=w1; w_id<=w2; w_id++)
{
for (d_id=1;  d_id <= DIST_PER_WARE; d_id++)
{
for (c_id=1; c_id <= CUST_PER_DIST; c_id++)
LoadCustHist(w_id, d_id, c_id);
}
printf("\nLoaded history for warehouse %d\n", w_id);
}
end_history_load();
}
LoadCustHist(w_id, d_id, c_id)
ID w_id, d_id, c_id;
{
h_c_id = c_id;
h_c_d_id = d_id;
h_c_w_id = w_id;
h_d_id = d_id;
h_w_id = w_id;
h_amount = 10.0 * 100;
MakeAlphaString(12, 24, h_data);
datetime(&h_date);
history_load();
}
begin_history_load()
{
inti = 1;
bulk_h = bulk_open("tpcc", "history", password);
bulk_bind(bulk_h, i++, "h_c_id",   &h_c_id,   ID_T);
bulk_bind(bulk_h, i++, "h_c_d_id", &h_c_d_id, ID_T);
bulk_bind(bulk_h, i++, "h_c_w_id", &h_c_w_id, ID_T);
bulk_bind(bulk_h, i++, "h_d_id",   &h_d_id,   ID_T);
bulk_bind(bulk_h, i++, "h_w_id",   &h_w_id,   ID_T);
bulk_bind(bulk_h, i++, "h_date",   &h_date,   DATE_T);
bulk_bind(bulk_h, i++, "h_amount", &h_amount, MONEY_T);
bulk_bind(bulk_h, i++, "h_data",   h_data,    TEXT_T);
}
history_load()
{
debug("h_c_id=%d  h_amount=%g\n", h_c_id, h_amount);
bulk_load(bulk_h);
}
end_history_load()
{
bulk_close(bulk_h);
}
/**********************************************************
Customer
```

```
**********************************************************/
/* static variables containing fields for customer record
*/
ID c_id;
ID c_d_id;
ID c_w_id;
TEXT c_first[16+1];
TEXT c_middle[2+1] = "OE";
TEXT c_last[16+1];
TEXT c_street_1[20+1];
TEXT c_street_2[20+1];
TEXT c_city[20+1];
TEXT c_state[2+1];
TEXT c_zip[9+1];
TEXT c_phone[16+1];
DATE c_since;
TEXT c_credit[2+1] = "?C";
MONEY c_credit_lim = 50000.0 * 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;
LoadCustomer(w1, w2)
ID w1, w2;
{
ID w_id;
begin_customer_load();
for (w_id=w1; w_id<=w2; w_id++)
{
Customer(w_id);
printf("\nLoaded customer for warehouse %d\n", w_id);
}
end_customer_load();
}
Customer(w_id)
int w_id;
{
BitVector badcredit[DIST_PER_WARE][(3000+WSZ-1)/WSZ], * bmp;
int i, j;
ID d_id;
/* Mark exactly 10% of customers as having bad credit */
for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
{
bmp = badcredit[d_id-1];
for (i=0; i<(3000+WSZ-1)/WSZ; i++)
bmp[i] = (BitVector)0x0000;
for (i=0; i<(3000+9)/10; i++)
{
do {
j = RandomNumber(0,3000-1);
} while (nthbit(bmp,j));
setbit(bmp,j);
}
}
c_w_id = w_id;
for (i=0; i<CUST_PER_DIST; i++)
{
c_id = i+1;
for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
{
c_d_id = d_id;
LastName(i<1000?i:NURandomNumber(255,NURAND_C,0,999),c_last);
MakeAlphaString(8, 16, c_first);
MakeAddress(c_street_1,c_street_2,c_city,c_state,c_zip);
MakeNumberString(16, 16, c_phone);
MakeAlphaString(300, 500, c_data);
datetime(&c_since);
c_credit[0] = nthbit(badcredit[d_id-1],i) ? 'B' : 'G';
c_discount = RandomNumber(0, 5000) / 10000.0;
customer_load();
}
}
}
begin_customer_load()
{
inti = 1;
bulk_c = bulk_open("tpcc", "customer", password);
bulk_bind(bulk_c, i++, "c_id",    &c_id,   ID_T);
bulk_bind(bulk_c, i++, "c_d_id",  &c_d_id, ID_T);
bulk_bind(bulk_c, i++, "c_w_id",  &c_w_id, ID_T);
bulk_bind(bulk_c, i++, "c_first",   c_first, TEXT_T);
bulk_bind(bulk_c, i++, "c_middle",  c_middle, TEXT_T);
bulk_bind(bulk_c, i++, "c_last",  c_last,   TEXT_T);
bulk_bind(bulk_c, i++, "street_1",  c_street_1 , TEXT_T);
bulk_bind(bulk_c, i++, "street_2",  c_street_2,  TEXT_T);
bulk_bind(bulk_c, i++, "c_city",  c_city,    TEXT_T);
bulk_bind(bulk_c, i++, "c_state",   c_state,   TEXT_T);
bulk_bind(bulk_c, i++, "c_zip",   c_zip,     TEXT_T);
bulk_bind(bulk_c, i++, "c_phone",   c_phone,   TEXT_T);
bulk_bind(bulk_c, i++, "c_since",   &c_since,   DATE_T);
bulk_bind(bulk_c, i++, "c_credit",  c_credit,  TEXT_T);
bulk_bind(bulk_c, i++, "c_credit_lim", &c_credit_lim, MONEY_T);
bulk_bind(bulk_c, i++, "c_discount", &c_discount, FLOAT_T);
bulk_bind(bulk_c, i++, "c_delivery_cnt", &c_delivery_cnt, COUNT_T);
bulk_bind(bulk_c, i++, "c_payment_cnt",&c_payment_cnt, COUNT_T);
bulk_bind(bulk_c, i++, "c_balance", &c_balance, MONEY_T);
bulk_bind(bulk_c, i++, "c_ytd_payment", &c_ytd_payment, MONEY_T);
bulk_bind(bulk_c, i++, "c_data_1", c_data1, TEXT_T);
bulk_bind(bulk_c, i++, "c_data_2", c_data2, TEXT_T);
}
customer_load()
{
debug("c_id=%-5d  d_id=%-5d  w_id=%-5d  c_last=%s\n",
   c_id,   c_d_id,  c_w_id,  c_last);
/* Break the string c_data into 2 pieces */
len = strlen(c_data);
if (len > 250)
{
memcpy(c_data1, c_data, 250);
c_data1[250]='\0';
memcpy(c_data2, c_data+250, len-250 +1);
}
```

```
else
{
memcpy(c_data1, c_data, 250+1);
strcpy(c_data2,"");
}
/* load the data */
bulk_load(bulk_c);
}
end_customer_load()
{
bulk_close(bulk_c);
}
/************************************************************Ord
er, Order line, New order
************************************************************/
/* Order row */
ID o_id;
ID o_c_id;
ID o_d_id;
ID o_w_id;
DATE o_entry_d;
ID o_carrier_id;
COUNT o_ol_cnt;
LOGICAL o_all_local;
/* Order line row */
ID   ol_o_id;
ID   ol_d_id;
ID   ol_w_id;
ID   ol_number;
ID   ol_i_id;
ID   ol_supply_w_id;
DATE ol_delivery_d;
COUNT ol_quantity;
MONEY ol_amount;
TEXT  ol_dist_info[24+1];
/* new order row */
ID no_o_id;
ID no_d_id;
ID no_w_id;
int  o_bulk;
int ol_bulk;
int no_bulk;
LoadOrd(w1, w2)
ID w1, w2;
{
ID w_id;
ID d_id;
begin_order_load();
begin_order_line_load();
for (w_id=w1; w_id<=w2; w_id++)
{
for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
Orders(w_id, d_id);
printf("\nLoaded order + order_line for warehouse %d\n", w_id);
}
end_order_line_load();
end_order_load();
}
LoadNew(w1, w2)
ID w1, w2;
{
ID w_id;
ID d_id;
begin_new_order_load();
for (w_id=w1; w_id<=w2; w_id++)
{
for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
{
no_d_id = d_id;
no_w_id = w_id;
for (no_o_id=2101;  no_o_id <= ORD_PER_DIST; no_o_id++)
new_order_load();
}
printf("\nLoaded new_order for warehouse %d\n", w_id);
}
end_new_order_load();
}
Orders(w_id, d_id)
ID w_id;
ID d_id;
{
int cust[ORD_PER_DIST+1];
int ol_cnt[ORD_PER_DIST+1], sum;
ID ol;
printf("\nLoading orders and order lines  for warehouse %d district %d\n",
 w_id, d_id);
RandomPermutation(cust, ORD_PER_DIST);
for (o_id = 1, sum=0;  o_id <= ORD_PER_DIST;  o_id++)
        sum += (ol_cnt[o_id] = RandomNumber(5, 15));
    while (sum > 10*ORD_PER_DIST)
    {
        do {
            o_id = RandomNumber(1,ORD_PER_DIST);
        } while (ol_cnt[o_id]==5);
        ol_cnt[o_id]--;
        sum--;
    }
    while (sum < 10*ORD_PER_DIST)
    {
        do {
            o_id = RandomNumber(1,ORD_PER_DIST);
        } while (ol_cnt[o_id]==15);
        ol_cnt[o_id]++;
        sum++;
    }
for (o_id = 1;  o_id <= ORD_PER_DIST;  o_id++)
{
o_c_id = cust[o_id];
o_d_id = d_id;
o_w_id = w_id;
datetime(&o_entry_d);
if (o_id <= 2100)
o_carrier_id = RandomNumber(1,10);
else o_carrier_id = -1;
o_ol_cnt = ol_cnt[o_id];
/* o_ol_cnt = RandomNumber(5, 15); */
o_all_local = 1;
order_load();
```

```
for (ol=1;  ol<=o_ol_cnt; ol++)
OrderLine(ol);
}
}
OrderLine(ol)
ID ol;
{
ol_o_id = o_id;
ol_d_id = o_d_id;
ol_w_id = o_w_id;
ol_number = ol;
ol_i_id = RandomNumber(1, MAXITEMS);
ol_supply_w_id = o_w_id;
ol_delivery_d = o_entry_d;
ol_quantity = 5;
if (o_id <= 2100) ol_amount = 0;
else  ol_amount = RandomNumber(1, 999999);
MakeAlphaString(24, 24, ol_dist_info);
order_line_load();
}
NewOrder(w_id, d_id)
ID w_id, d_id;
{
no_d_id = o_d_id;
no_w_id = o_w_id;
for (no_o_id=2101;  no_o_id <= ORD_PER_DIST; no_o_id++)
new_order_load();
}
begin_order_load()
{
inti = 1;
o_bulk = bulk_open("tpcc", "orders", password);
bulk_bind(o_bulk, i++, "o_id", &o_id, ID_T);
bulk_bind(o_bulk, i++, "o_c_id", &o_c_id, ID_T);
bulk_bind(o_bulk, i++, "o_d_id", &o_d_id, ID_T);
bulk_bind(o_bulk, i++, "o_w_id", &o_w_id, ID_T);
bulk_bind(o_bulk, i++, "o_entry_d", &o_entry_d, DATE_T);
bulk_bind(o_bulk, i++, "o_carrier_id", &o_carrier_id, ID_T);
bulk_bind(o_ol_cnt, i++, "o_ol_cnt", &o_ol_cnt, COUNT_T);
bulk_bind(o_all_local, i++, "o_all_local", &o_all_local, LOGICAL_T);
}
order_load()
{
debug("o_id=%d o_c_id=%d  count=%d\n", o_id, o_c_id, o_ol_cnt);
bulk_load(o_bulk);
}
end_order_load()
{
bulk_close(o_bulk);
}
begin_order_line_load()
{
inti = 1;
ol_bulk = bulk_open("tpcc", "order_line", password);
bulk_bind(ol_bulk, i++, "ol_o_id", &ol_o_id, ID_T);
bulk_bind(ol_bulk, i++, "ol_d_id", &ol_d_id, ID_T);
bulk_bind(ol_bulk, i++, "ol_w_id", &ol_w_id, ID_T);
bulk_bind(ol_bulk, i++, "ol_number", &ol_number, ID_T);
bulk_bind(ol_bulk, i++, "ol_i_id", &ol_i_id, ID_T);
bulk_bind(ol_bulk, i++, "ol_supply_w_id", &ol_supply_w_id, ID_T);
bulk_bind(ol_bulk, i++, "ol_delivery_d", &ol_delivery_d, DATE_T);
bulk_bind(ol_bulk, i++, "ol_quantity", &ol_quantity, COUNT_T);
bulk_bind(ol_bulk, i++, "ol_amount", &ol_amount, MONEY_T);
bulk_bind(ol_bulk, i++, "ol_dist_info", ol_dist_info, TEXT_T);
}
order_line_load()
{
static int ol_count = 0;
debug("  ol_o_id=%d  ol_number=%d  ol_amount=%g\n",
 ol_o_id,  ol_number,  ol_amount);
bulk_load(ol_bulk);
}
end_order_line_load()
{
bulk_close(ol_bulk);
}
begin_new_order_load()
{
inti = 1;
no_bulk = bulk_open("tpcc", "new_order", password);
bulk_bind(no_bulk, i++, "no_o_id", &no_o_id, ID_T);
bulk_bind(no_bulk, i++, "no_d_id", &no_d_id, ID_T);
bulk_bind(no_bulk, i++, "no_w_id", &no_w_id, ID_T);
}
new_order_load()
{
debug("  no_o_id=%d  \n", no_o_id);
bulk_load(no_bulk);
}
end_new_order_load()
{
bulk_close(no_bulk);
}
/*************************************************************Sto
ck
*************************************************************/
ID s_i_id;
ID s_w_id;
COUNT s_quantity;
TEXT s_dist_01[24+1];
TEXT s_dist_02[24+1];
TEXT s_dist_03[24+1];
TEXT s_dist_04[24+1];
TEXT s_dist_05[24+1];
TEXT s_dist_06[24+1];
TEXT s_dist_07[24+1];
TEXT s_dist_08[24+1];
TEXT s_dist_09[24+1];
TEXT s_dist_10[24+1];
COUNT s_ytd;
COUNT s_order_cnt;
COUNT s_remote_cnt;
TEXT s_data[50+1];
int bulk_s;
/*
** On loading stock in major order of item_id:
** 10% of the MAXITEMS items in each warehouse
need to marked as original
```

```
** (i.e., s_data like '%ORIGINAL%'.)  This is a bit
harder to do when we
** load by item number, rather than by warehouses.
The trick is to first
** generate a huge WAREBATCH * MAXITEMS
bitmap, initialize all bits to zero,
** and then set 10% of bits in each row to 1.  While
loading item i in
** warehouse w, we simply lookup bitmap[w][i] to see
whether it needs to
** be marked as original.
*/
LoadStock(w1, w2)
ID w1, w2;
{
ID w_id;
BitVector original[WAREBATCH][((MAXITEMS+(WSZ-1))/WSZ)], * bmp;
int w, i, j;
if (w2-w1+1 > WAREBATCH)
{
fprintf(stderr, "Can't load stock for %d warehouses.\n",
w2-w1+1);
fprintf(stderr, "Please use batches of %d.\n", WAREBATCH);
}
for (w=w1; w<=w2; w++)
{
bmp = original[w-w1];
/* Mark all items as not "original" */
for (i=0; i<(MAXITEMS+(WSZ-1))/WSZ; i++)
bmp[i] = (BitVector)0x0000;
/* Mark exactly 10% of items as "original" */
for (i=0; i<(MAXITEMS+9)/10; i++)
{
do {
j = RandomNumber(0,MAXITEMS-1);
} while (nthbit(bmp,j));
setbit(bmp,j);
}
}
printf("Loading stock for warehouse %d to %d.\n", w1, w2);
begin_stock_load();
/* do for each item */
for (s_i_id=1;   s_i_id <= MAXITEMS; s_i_id++)
{
for (w_id=w1; w_id<=w2; w_id++)
{
/* Generate Stock Data */
s_w_id = w_id;
s_quantity = RandomNumber(10,100);
MakeAlphaString(24, 24, s_dist_01);
MakeAlphaString(24, 24, s_dist_02);
MakeAlphaString(24, 24, s_dist_03);
MakeAlphaString(24, 24, s_dist_04);
MakeAlphaString(24, 24, s_dist_05);
MakeAlphaString(24, 24, s_dist_06);
MakeAlphaString(24, 24, s_dist_07);
MakeAlphaString(24, 24, s_dist_08);
MakeAlphaString(24, 24, s_dist_09);
MakeAlphaString(24, 24, s_dist_10);
s_ytd = 0;
s_order_cnt = 0;
s_remote_cnt = 0;
MakeAlphaString(26, 50, s_data);
if (nthbit(original[w_id-w1],s_i_id-1))
{
Original(s_data);
}
stock_load();
}
}
end_stock_load();
printf("\nLoaded stock for warehouses %d to %d.\n", w1, w2);
}
begin_stock_load()
{
inti = 1;
bulk_s = bulk_open("tpcc", "stock", password);
bulk_bind(bulk_s, i++, "s_i_id", &s_i_id, ID_T);
bulk_bind(bulk_s, i++, "s_w_id", &s_w_id, ID_T);
bulk_bind(bulk_s, i++, "s_quantity", &s_quantity, COUNT_T);
bulk_bind(bulk_s, i++, "s_ytd", &s_ytd,COUNT_T);
bulk_bind(bulk_s, i++, "s_order_cnt", &s_order_cnt, COUNT_T);
bulk_bind(bulk_s, i++, "s_remote_cnt", &s_remote_cnt, COUNT_T);
bulk_bind(bulk_s, i++, "s_dist_01", s_dist_01, TEXT_T);
bulk_bind(bulk_s, i++, "s_dist_02", s_dist_02, TEXT_T);
bulk_bind(bulk_s, i++, "s_dist_03", s_dist_03, TEXT_T);
bulk_bind(bulk_s, i++, "s_dist_04", s_dist_04, TEXT_T);
bulk_bind(bulk_s, i++, "s_dist_05", s_dist_05, TEXT_T);
bulk_bind(bulk_s, i++, "s_dist_06", s_dist_06, TEXT_T);
bulk_bind(bulk_s, i++, "s_dist_07", s_dist_07, TEXT_T);
bulk_bind(bulk_s, i++, "s_dist_08", s_dist_08, TEXT_T);
bulk_bind(bulk_s, i++, "s_dist_09", s_dist_09, TEXT_T);
bulk_bind(bulk_s, i++, "s_dist_10", s_dist_10, TEXT_T);
bulk_bind(bulk_s, i++, "s_data",   s_data,  TEXT_T);
}
stock_load()
{
debug("s_i_id=%d w_id=%d s_data=%s\n",
   s_i_id, s_w_id,   s_data);
bulk_load(bulk_s);
}
end_stock_load()
{
bulk_close(bulk_s);
}
test(){}
getargs(argc, argv)
/*************************************************************
configure configures the load stuff
   By default, loads all the tables for a the specified
warehouse.
*************************************************************/
int argc;
char **argv;
{
char ch;
/* define the defaults */
load_item = load_warehouse = load_district = load_history =
```

```
load_orders = load_new_order = load_order_line =
load_customer = load_stock = NO;
if     (strcmp(argv[1], "warehouse") == 0) load_warehouse = YES;
else if (strcmp(argv[1], "district") == 0)  load_district = YES;
else if (strcmp(argv[1], "stock") == 0)  load_stock = YES;
else if (strcmp(argv[1], "item") == 0)  load_item = YES;
else if (strcmp(argv[1], "history") == 0)  load_history = YES;
else if (strcmp(argv[1], "orders") == 0)  load_orders = YES;
else if (strcmp(argv[1], "customer") == 0)  load_customer = YES;
else if (strcmp(argv[1], "new_order") ==0) load_new_order = YES;
else
{
 printf("%s is not a valid table name\n", argv[1]);
 exit(0);
}
/* Set the w1 and w2 to argv[2] and arg[3] */
if (argc < 3)
{
printf("Usage: %s <table> <w_first> [<w_last>]\n", argv[0]);
exit(1);
}
{
w1 = atoi(argv[2]);
if (argc >= 3)
w2 = atoi(argv[3]);
else
w2 = w1;
}
/* Get the password for sa */
if (argc > 4)
strcpy(password,argv[4]);
/* Check if warehouse is within the range */
if (w1 <= 0 || w2 > 1000 || w1 > w2)
{
printf("Warehouse id is out of range\n");
exit(0);
}
}
double drand48();
MakeAddress(str1, str2, city, state, zip)
TEXT str1[20+1];
TEXT str2[20+1];
TEXT city[20+1];
TEXT state[2+1];
TEXT zip[9+1];
{
MakeAlphaString(10,20,str1);
MakeAlphaString(10,20,str2);
MakeAlphaString(10,20,city);
MakeAlphaString(2,2,state);
MakezipString(0,9999,zip);
/* Changed for TPCC V 3.0 */
strcat(zip, "11111");
}
LastName(num, name)
/*************************************************************
Lastname generates a lastname from a number.
*************************************************************/
int num;
char name[20+1];
{
int i;
static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
"ESE", "ANTI", "CALLY", "ATION", "EING"};
strcpy(name, n[(num/100)%10]);
strcat(name, n[(num/10) %10]);
strcat(name, n[(num/1)  %10]);
}
int MakeNumberString(min, max, num)
int min;
int max;
TEXT num[];
{
static char digit[]="0123456789";
int length;
int i;
length = RandomNumber(min, max);
for (i=0;  i<length;  i++)
num[i] = digit[RandomNumber(0,9)];
num[length] = '\0';
return length;
}
int MakezipString(min, max, num)
        int min;
        int max;
        TEXT num[];
{
        static char digit[]="0123456789";
        int length;
        int i;
        length = 4;

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

        return length;
}
int MakeAlphaString(min, max, str)
int min;
int max;
TEXT str[];
{
static char character[] =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456
789";
int length;
int i;
length = RandomNumber(min, max);
for (i=0;  i<length;  i++)
str[i] = character[RandomNumber(0, sizeof(character)-2)];
str[length] = '\0';
return length;
}
Original(str)
TEXT str[];
{
int pos;
```

```
int len;
len = strlen(str);
if (len < 8) return;
pos = RandomNumber(0,len-8);
str[pos+0] = 'O';
str[pos+1] = 'R';
str[pos+2] = 'I';
str[pos+3] = 'G';
str[pos+4] = 'I';
str[pos+5] = 'N';
str[pos+6] = 'A';
str[pos+7] = 'L';
}
RandomPermutation(perm, n)
int perm[];
int n;
{
int i, r, t;
/* generate the identity permutation to start with */
for (i=1; i<=n; i++)
perm[i] = i;
/* randomly shuffle the permutation */
for (i=1; i<=n; i++)
{
r = RandomNumber(i, n);
t = perm[i]; perm[i] = perm[r]; perm[r] = t;
}
}
int Randomize()
{
srand48(time(0)+getpid());
}
int RandomNumber(min, max)
int min;
int max;
{
int r;
r = (int)(drand48() * (max - min + 1)) + min;
return r;
}
int NURandomNumber(a, c, min, max)
int a;
int c;
int min;
int max;
{
int r;
r = ((RandomNumber(0, a) | RandomNumber(min, max)) + c)
% (max - min + 1) + min;
return r;
}
```

loader.h

```
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED
#include <sybfront.h>
#include <sybdb.h>
#include <time.h>
/* Population constants */
#ifdef CACHED
#define MAXITEMS 10000
#define CUST_PER_DIST 300
#define DIST_PER_WARE 10
#define ORD_PER_DIST 300
#else
#define MAXITEMS 100000
#define CUST_PER_DIST 3000
#define DIST_PER_WARE 10
#define ORD_PER_DIST 3000
#endif
#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 */
```

B.4 Table Index Definition

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
) 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,                                                    float
h_w_id    smallint,
h_date    datetime,
h_amount
h_data    char(24)
) on Shistory
go
alter table history partition 8
go
if exists ( select name from sysobjects where name = 'new_order' )
drop table new_order
go
create table new_order (
no_o_id   int,
no_d_id    tinyint,
no_w_id   smallint,
) on Scache
go
create unique clustered index no_clu
on new_order(no_w_id, no_d_id, no_o_id)
on Scache
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips, 100, new_order)
go
if exists ( select name from sysobjects where name = 'orders' )
drop table orders
go
create table orders (
o_id    int,
o_c_id   int,
o_d_id    tinyint,
o_w_id     smallint,
o_entry_d   datetime,
o_carrier_id   smallint,    /*- Updated by D */
o_ol_cnt   tinyint,
o_all_local   tinyint
) on Sorders
go
create unique clustered index o_clu
on orders(o_w_id, o_d_id, o_id)
on Sorders
go
dbcc tune(ascinserts, 1, orders)
go
dbcc tune(oamtrips, 100, orders)
```

```
go
if exists ( select name from sysobjects where name = 'order_line' )
drop table order_line
go
create table order_line (
ol_o_id    int,
ol_d_id    tinyint,
ol_w_id   smallint,
ol_number    tinyint,
ol_i_id    int,
ol_supply_w_id  smallint,
ol_delivery_d   datetime,    /*- Updated by D */
ol_quantity   smallint,
ol_amount     float,
ol_dist_info   char(24)
) on Sorder_line
go
create unique clustered index ol_clu
on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
on Sorder_line
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go
if exists ( select name from sysobjects where name = 'item' )
drop table item
go
create table item (
i_id    int,
i_im_id   int,
i_name   char(24),
i_price    float,   i_data   char(50)
) on Scache
go
create unique clustered index i_clu
on item(i_id)
on Scache
go
dbcc tune(indextrips, 10, item)
go
if exists ( select name from sysobjects where name = 'stock' )
drop table stock
go
create table stock (
s_i_id    int,
s_w_id   smallint,
s_quantity   smallint,   /*- Updated by NO */
s_ytd    int,    /*- Updated by NO */
s_order_cnt   smallint,    /*- Updated by NO */
s_remote_cnt   smallint,   /*- Updated by NO */
s_dist_01   char(24),
s_dist_02   char(24),
s_dist_03   char(24),
s_dist_04   char(24),
s_dist_05   char(24),
s_dist_06   char(24),
s_dist_07   char(24),
s_dist_08   char(24),
s_dist_09   char(24),
s_dist_10   char(24),
s_data     char(50)
) on Sstock
go
create unique clustered index s_clu
on stock(s_i_id, s_w_id)
on Sstock
go
dbcc tune(indextrips, 10, stock)
go
checkpoint
go
EOF
```

C.1 Aix Clients Tune

zostune.ver

keylock normal State of system keylock at boot time False
maxbuf 20 Maximum number of pages in block I/O BUFFER CACHE True
maxmbuf 0 Maximum Kbytes of real memory allowed for MBUFS True
maxuproc 10000 Maximum number of PROCESSES allowed per user True
autorestart false Automatically REBOOT system after a crash True
iostat true Continuously maintain DISK I/O history True
realmem 524288 Amount of usable physical memory in Kbytes False
conslogin enable System Console Login False
maxpout 0 HIGH water mark for pending write I/Os per file True
minpout 0 LOW water mark for pending write I/Os per file True
fullcore false Enable full CORE dump True

C.2 Aix Server Tune

zostune.ver

keylock normal State of system keylock at boot time False
maxbuf 20 Maximum number of pages in block I/O BUFFER CACHE True
maxmbuf 0 Maximum Kbytes of real memory allowed for MBUFS True
maxuproc 4096 Maximum number of PROCESSES allowed per user True
autorestart false Automatically REBOOT system after a crash True
iostat true Continuously maintain DISK I/O history True
realmem 2097152 Amount of usable physical memory in Kbytes False
conslogin enable System Console Login False
maxpout 0 HIGH water mark for pending write I/Os per file True
minpout 0 LOW water mark for pending write I/Os per file True
fullcore false Enable full CORE dump True

C.3 Sybase Tune

run.cfg

```
#########################################
##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
[Named Cache:c_customer]
cache size = 10M
cache status = mixed cache
 [2K I/O Buffer Pool]
pool size = 10M
wash size = 6144 K
[Named Cache:c_customer_index]
cache size = 186M
cache status = mixed cache
cache status = HK ignore cache
[2K I/O Buffer Pool]
pool size = 186M
wash size = 512 K
[Named Cache:c_log]
cache size = 8M
cache status = log only
[2K I/O Buffer Pool]
pool size = 1M
wash size = 512 K
[4K I/O Buffer Pool]
pool size = 1M
wash size = 512 K
[8K I/O Buffer Pool]
pool size = 6M
wash size = 3686 K
```

```
[Named Cache:c_no_ol]
cache size = 110M
cache status = mixed cache
[2K I/O Buffer Pool]
pool size = 110M
wash size = 8192 K
[Named Cache:c_ol_index]
cache size = 30M
cache status = mixed cache
[2K I/O Buffer Pool]
pool size = 30M
wash size = 4096 K
[Named Cache:c_orders]
cache size = 72M
cache status = mixed cache
[2K I/O Buffer Pool]
pool size = 68M
wash size = 2048 K
[16K I/O Buffer Pool]
pool size = 4M
wash size = 512 K
[Named Cache:c_stock]
cache size = 1189M
cache status = mixed cache
[2K I/O Buffer Pool]
pool size = 1189M
wash size = 40960 K
[Named Cache:c_stock_index]
cache size = 176M
cache status = mixed cache
cache status = HK ignore cache
[2K I/O Buffer Pool]
pool size = 176M
wash size = 512 K
[Named Cache:c_sysindexes]
cache size = 512K
cache status = mixed cache
cache status = HK ignore cache
[2K I/O Buffer Pool]
pool size = 512K
wash size = 256 K
[Named Cache:c_tinyhot]
cache size = 21.5 M
cache status = mixed cache
cache status = HK ignore cache
[2K I/O Buffer Pool]
pool size = 4M
wash size = 512 K
[16K I/O Buffer Pool]
pool size = 17.5 M
wash size = 512 K
[Named Cache:default data cache]
cache size = 8M
cache status = default data cache
cache status = HK ignore cache
[2K I/O Buffer Pool]
pool size = 6M
wash size = 2048 K
[16K I/O Buffer Pool]
pool size = 2M
wash size = 1024 K
[Disk I/O]
allow sql server async i/o = DEFAULT
disk i/o structures = DEFAULT
page utilization percent = DEFAULT
number of devices = 120
disable character set conversions = 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 = 1012
max async i/os per server = 1012
[Physical Resources]
[Physical Memory]
total memory = 998000
additional network memory = 1628160
lock shared memory = DEFAULT
shared memory starting address = DEFAULT
[Processors]
max online engines = 8
min online engines = DEFAULT
[SQL Server Administration]
number of open objects = DEFAULT
number of open databases = DEFAULT
audit queue size = DEFAULT
default database size = DEFAULT
identity burning set factor = DEFAULT
allow nested triggers = DEFAULT
allow updates to system tables = DEFAULT
print deadlock information = DEFAULT
default fill factor percent = DEFAULT
number of mailboxes = DEFAULT
number of messages = DEFAULT
number of alarms = DEFAULT
number of pre-allocated extents = DEFAULT
event buffers per engine = DEFAULT
cpu accounting flush interval = DEFAULT
i/o accounting flush interval = DEFAULT
sql server clock tick length = DEFAULT
runnable process search count = DEFAULT
i/o polling process count = DEFAULT
time slice = DEFAULT
deadlock retries = DEFAULT
cpu grace time = DEFAULT
number of sort buffers = DEFAULT
sort page count = DEFAULT
number of extent i/o buffers = DEFAULT
size of auto identity column = DEFAULT
```

```
identity grab size = DEFAULT
lock promotion HWM = DEFAULT
lock promotion LWM = DEFAULT
lock promotion PCT = DEFAULT
housekeeper free write percent = 0
partition groups = DEFAULT
partition spinlock ratio = DEFAULT
[User Environment]
number of user connections = 120
stack size = DEFAULT
stack guard size = DEFAULT
systemwide password expiration = DEFAULT
permission cache entries = DEFAULT
user log cache size = 4096
user log cache spinlock ratio = DEFAULT
[Lock Manager]
number of locks = 10000
deadlock checking period = DEFAULT
freelock transfer block size = DEFAULT
max engine freelocks = 25
address lock spinlock ratio = 5
page lock spinlock ratio = 5
table lock spinlock ratio = 1
```

C.4 Tuxedo Tune

ubbshm

```
#--------------------------------------------#
*RESOURCES                          #
#--------------------------------------------#
IPCKEY   70952   # IPC KEY from 32,768 to 16,777,215
UID          342   # user id as displayed by command "id"
GID          200   # group id as displayed by command "id"
PERM      0660   # UNIX permission from 0001 to 0777 in octal
MAXACCESSERS    3400   # max no of processes accesing bulleting board
MAXSERVERS         200  # maximum number of servers
MAXSERVICES        210  # maximum number of services
MASTER      SERVER   # machine on which master copy is found
SCANUNIT             60   # scan program wake-up time in secs.
MODEL    SHM   # SHM=single processor, MP=multi processor
LDBAL       Y        # load balancing, Y=yes, N=no
MAXGTT    2048    # maximum simultaneous global transactions
SANITYSCAN   5   # sanity scan wake-up
DBBLWAIT      1   # scanunit multiplier for DBBL max time wait
BBLQUERY   30     # check out wake-up time
BLOCKTIME    5    # blocking call time-out
TAGENT    "TAGENT"   # alphanumeric service code
#--------------------------------------------#
*MACHINES       #
#--------------------------------------------#
DEFAULT:
ROOTDIR="/usr/tuxedo"
APPDIR="/tpcc/tpcs_work/app/sybase/client"
TUXCONFIG="/tpcc/tpcs_work/app/sybase/tuxedo/tuxconfig"
#SYSTEMLOG="/tpcc/tpcs_work/app/sybase/tuxedo/SYSTEMLOG"
ULOGPFX="/tpcc/tpcs_work/app/sybase/tuxedo/SYSTEMLOG"
client8   LMID=SERVER
#--------------------------------------------#
*GROUPS           #
#--------------------------------------------#
GROUP1    GRPNO=1    # server group number
LMID=SERVER   # logical machine identifier
OPENINFO=NONE   # resource mgr. info
#--------------------------------------------#
*SERVERS   #
#--------------------------------------------#
DEFAULT:   SRVGRP=GROUP1    # server group
#DEFAULT:  REPLYQ=N   # reply queue, Y=yes, N=no
#DEFAULT:   MAXGEN=2   # re-startable times
#DEFAULT:   RESTART=Y   # re-start status, Y=yes, N=no
neword_server   SRVID=100 MIN=7 MAX=12 RQADDR="100"
            CLOPT="-s NEWORD_DSQL"

payment_server  SRVID=200 MIN=4 MAX=12 RQADDR="200"
            CLOPT="-s PAYMENT_DSQL"

ordstat_server  SRVID=300 MIN=2 MAX=12 RQADDR="300"
            CLOPT="-s ORDSTAT_DSQL"

stocklev_server SRVID=400 MIN=3 MAX=12 RQADDR="400"
            CLOPT="-s STOCKLEV_DSQL"

delivery_server SRVID=500 MIN=3 MAX=12 RQADDR="500"
            CLOPT="-s DELIVERY_DSQL"
#--------------------------------------------#
*SERVICES                #
#--------------------------------------------#
```

This appendix contains the source and makefiles for all client and server programs

D.1 RTE Parameters

rteparams

```
/* define hosts that will be used here */
MASTER driver1
SLAVES driver5, driver6, driver7
SUT   = "gandalf"

CLIENT client33 sa syBase
CLIENT client44 sa syBase
CLIENT client55 sa syBase
CLIENT client66 sa syBase
CLIENT client77 sa syBase
CLIENT client88 sa syBase

TELNET telnet 23

BEGIN_WAIT=05:00
RAMPUP=30:00
RAMPDOWN=15:00
RAMPDOWN_WAIT=07:30
RUNTIME=30:00
INTERVAL=05:00   /* Interval to calculate mix from */

NOBEGIN              = 0
KEYSTROKE_PACKET_SIZE = 200

MAX_CONCURRENT_SPAWN  = 20
SPAWN_COUNT          = 2

/* User variables : ThinkTime,
Emulex_Response_Delay, Emulex_Menu_Delay,
%desired, %min, %max */
NEWORDER = "12.2, 0.00, 0.00"
PAYMENT  = "12.2, 0.00, 0.00, 43.20, 42.00, 43.10"
ORDSTAT  = "10.6, 0.00, 0.00,  4.10,  3.80,  4.06"
DELIVERY = "05.1, 0.00, 0.00,  4.10,  3.90,  4.17"
STOCKLEV = "05.1, 0.00, 0.00,  4.10,  3.90,  4.08"

START client33 telnet 1010
START client44 telnet 1010
START client55 telnet 1010
START client66 telnet 1010
START client77 telnet 1010
START client88 telnet 1010
```

D.2 RTE Master

user_master.c

```
/*********************************/
/*      INCLUDES FILES           */
/*********************************/
#define _H_CUR01
#include <cur00.h>
#undef _H_CUR01

extern "C"
  {
    #include "data/cur01.h"
    int wrefresh (WINDOW *);
    int wclrtoeol(WINDOW *);
    int setupterm(char*,FILE*,int*);
    int nodelay(int);
    int keypad(int);
    int wgetch(WINDOW *);
  }
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include "data/rte.h"
#include "data/Stats.h"
#include "data/misc.h"
#include "user_tpcc.h"

/*********************************/
/*      STRUCTURE DEFS   */
/*********************************/
struct header_s
  {
    int slave;
    int num;
    int type;
    int num_timestamps;
    int user_data_length;
    int data_type;
  };
struct UserSpawnData
  {
    int Warehouse;
    int District;
  };
/*********************************/
/*      EXTERNAL FUNCTIONS       */
/*********************************/
extern "C" int strcasecmp   (char *s1, char *s2);
extern "C" int strncasecmp  (char *s1, char *s2, int n);
int       send_global_data   (void);
/*********************************/
/*      INTERNAL FUNCTIONS       */
```

```
/*********************************/
char    *get_variable   (char *name);
int        get_variable   (char *name, int *number);
int        make_ratios   (double *buffer);
int user_statistics_print
int user_spawn   (int *length, char *buffer);
int user_finished    (int length, char *buffer);
/*********************************/
/*      EXTERNAL VARIABLES      */
/*********************************/
extern Stats status[MAX_TRAN_TYPE][MAX_TIMES];
extern WINDOW   *statistics_win;
extern UserGlobal   *shmglobal;
extern int    ramp_up_complete;
extern int   interval_start_time,
interval_stop_time;
/*********************************/
/*      INTERNAL VARIABLES      */
/*********************************/
SlaveStatus   slave_status[MAX_SLAVES];
/* Transaction mix parameters */
double    ratio_desired[6],
ratio_min[6],
ratio_max[6],
ratio_range[6];
char *ratio_names[] =
{ "RTE", "NEWORDER", "PAYMENT", "ORDSTAT", "DELIVERY",
 "STOCKLEV", NULL };
char *Status_Names[] = {"Menu", "Keying", "Response", "Think"};
char *transaction_names[] = { "RTE", "New Order", "Payment", "Order Stat",
"Delivery", "Stock Level", NULL };
static int current_status = 2,
status_needs_refresh = 1;
/************************************************************
*      FUNCTION : user_statistics_print
*      PURPOSE :
 *            Print statistics on the main RTE board.
*      INPUT :
 *            no one
*      OUTPUT :
 *            no one
*      RETURN :
 *            RTE_OK
*      DESCRIPTION :
************************************************************/
int user_statistics_print(void)
  {
    int   i;
    static int   count = 0;
    double   ratios[6];

    if (status_needs_refresh)
      {
count      = 0;
status_needs_refresh = 0;
wmove   (statistics_win, 0, 0);
wprintw (statistics_win, "%11s %8s %8s %8s %8s %8s %6s %6s %6s",
Status_Names[current_status], "90%", "Avg", "Min", "Max",
"Samples", "Ratio", "Mix", "Think");
      }
    make_ratios(ratios);

    for (i = 1; i <= 5; i++)
{                                                                                                    (void);
 /* The reason we do this is because calculating the percentiles
    is expensive */
 if (count % 10 == 0)
   {
    wmove   (statistics_win, i, 0);
    wprintw (statistics_win, "%11s %8.2f",
     transaction_names[i],
     status[i][current_status].ninety()/1000.0);
   count = 0;
  }
 wmove   (statistics_win, i, 21);
 wprintw (statistics_win, "%8.2f %8.2f %8.2f %8d %6.2f %6.2f %6.2f",
  status[i][current_status].average()/1000.0,
  status[i][current_status].min()/1000.0,
  status[i][current_status].max()/1000.0,
  status[i][current_status].samples(),
  ratios[i], shmglobal->chances[i],
  status[i][3].average()/1000.0);
      }
    wmove   (statistics_win, 7, 0);

    extern int   runtime_counts[MAX_TRAN_TYPE];
    extern int   begin_time,
ramp_up,
run_time;
    int   start = interval_start_time;
    int   stop  = interval_stop_time;
    double   interval = ((double)(stop-start) / (1000*60));
    double   samples = status[1][2].samples();

    if (interval <= 0 || samples <= 0)
      {
wprintw (statistics_win, "TPM-C: %7s / ", "-------");
      }
    else
      {
wprintw (statistics_win, "TPM-C: %7.2f / ", samples/interval);
      }
    samples = runtime_counts[1];
    if (samples > 0)
      {
start = begin_time+((ramp_up>=0)?ramp_up:0);
if (run_time > 0 && stop > begin_time + ramp_up + run_time)
  {
   stop = begin_time + ramp_up + run_time;
  }
interval = (double)(stop - start)/(1000.0*60.0);
wprintw (statistics_win, "%7.2f", samples/interval);
      }
    else
      {
wprintw (statistics_win, "-------");
      }
    count++;
```

```
    return RTE_OK;
  }
const int MAX_WAREHOUSES = 1000;
/* All of this 10 stuff is district size.  Should be a constant.
   Maybe fix that later */
int num_warehouses       = -1;
int warehouses[MAX_WAREHOUSES * 10];
/*************************************************************
 *      FUNCTION : user_spawn
 *      PURPOSE :
 *            Set the new index for warehouse and
district when spawn new
 *      INPUT :
 *            number
 *            length
 *            buffer
 *      OUTPUT :
 *            no one
 *      RETURN :
 *            RTE_OK
 *      DESCRIPTION :
*************************************************************/
int user_spawn(int number, int *length, char *buffer)
  {
    int  i,
min_index;
    UserSpawnData   *ptr = (UserSpawnData *)buffer;

    *length = sizeof(*ptr);

    min_index = 0;
    for (i = 1; i < (num_warehouses)*10 && i <
MAX_WAREHOUSES*10; i++)
{
 if (warehouses[i] < warehouses[min_index])
    {
     min_index = i;
    }
       }
ptr->Warehouse = min_index / 10 + 1;
    ptr->District  = min_index % 10 + 1;
    warehouses[min_index]++;

    /* iprint (IPRINT_INFO, "Driver for Warehouse %d, District
%d started.  warehouses[%d]++ = %d\n",
           ptr->Warehouse, ptr->District, min_index,
warehouses[min_index]); */
return RTE_OK;
  }
/*************************************************************
 *      FUNCTION : user_finished
 *      PURPOSE :
 *      Set the new value for warehouse when user on client finished.
 *      INPUT :
 *            length
 *            buffer
 *      OUTPUT :
 *            no one
 *      RETURN :
 *            RTE_OK
 *      DESCRIPTION :
*************************************************************/
int user_finished(int length, char *buffer)
  {
    UserSpawnData  *ptr = (UserSpawnData *)buffer;
    int temp = (ptr->Warehouse-1)*10+ptr->District-1;

    warehouses[temp]--;
/* iprint (IPRINT_INFO, "Driver for Warehouse %d, District %d died.
warehouses[%d]-- = %d\n",
          ptr->Warehouse, ptr->District, temp, warehouses[temp]); */

    return RTE_OK;
  }
/*************************************************************
 *      FUNCTION : limit
 *      PURPOSE :
 *            Return value between the min and max limit.
 *      INPUT :
 *            min : lower born
 *            max : upper born
 *      OUTPUT :
 *            no one
 *      RETURN :
 *            Value between the min and max limit.
 *      DESCRIPTION :
*************************************************************/
double limit(double min, double max, double val)
  {
    if (val < min)
return min;
    if (val > max)
return max;
    return val;
  }
/*************************************************************
 *      FUNCTION : make_ratios
 *      PURPOSE :
 *            Set the ratios values for each of the five transactions
 *      cording to the current value of the number of transactions.
 *      INPUT :
 *            no one
 *      OUTPUT :
 *            buffer : Table of transactions ratios.
 *      RETURN :
 *            The total number of sample.
 *      DESCRIPTION :
*************************************************************/
int make_ratios(double *buffer)
  {
    int neword   = status[NEWORDER][0].samples();  // Nb of neword
    int payment  = status[PAYMENT] [0].samples();   // Nb of payment
    int ordstat  = status[ORDSTAT] [0].samples();   // Nb of ordstat
    int delivery = status[DELIVERY][0].samples();  // Nb of delivery
    int stocklev = status[STOCKLEV][0].samples();  // Nb of stocklev
    int total    = neword + payment + ordstat + delivery + stocklev;
    int i;
```

```
   if (total == 0)
      {
buffer[NEWORDER] = 100.0;
for (i = 2; i < 6; i++)
   {
    buffer[i]         = ratio_desired[i];
    buffer[NEWORDER] -= buffer[i];
   }
return 0;
      }
   buffer[PAYMENT]  = (double)payment  / (double)total * 100.0;
   buffer[ORDSTAT]  = (double)ordstat  / (double)total * 100.0;
   buffer[DELIVERY] = (double)delivery / (double)total * 100.0;
   buffer[STOCKLEV] = (double)stocklev / (double)total * 100.0;
   buffer[NEWORDER] = 100.0 - buffer[PAYMENT] - buffer[ORDSTAT] -
     buffer[DELIVERY] - buffer[STOCKLEV];
   return total;
  }
/*************************************************************
 *     FUNCTION : user_global_update
 *     PURPOSE :
 *           Make update for all users.
 *     INPUT :
 *           length :
 *           buffer :
 *     OUTPUT :
 *           no one
 *     RETURN :
 *           RTE_OK or RTE_ERROR.
 *     DESCRIPTION :
 *
*************************************************************/
int user_global_update(int *length, char *buffer)
  {
   UserGlobal   *shmglobal = (UserGlobal *)buffer;
   static double   last[6];
   static int    users_last = -1;
   double    ratios[6];
   double   current[6];
   int   i,
different = 0,
desired = 0;
   time_t    clock;
*length = sizeof(*shmglobal);
// Make the ratios.
   make_ratios(ratios);
   /* Calculate ratios we want for next time */
   /* Note: we just keep on with the desired values until ramp-up is complete
      this at least starts us out without any humps or spikes in the graph */
   // Verify if we are in Ramp-Up od Run window.
   if (ramp_up_complete)
      {
// time(&clock);
      // iprint(IPRINT_TRACE, "Date:%s -> ramp_up_complete\n",
ctime(&clock));
// We are in the Run window now.
// Set the neword ratio to 100%.
current[NEWORDER] = 100.0;
// Loop for all other transactions.
for (i = 2; i < 6; i++)
   {
    // Control if we have reach the ratio desired or not.
    if (ratio_desired[i] > ratios[i])
{
 // Set the current ratio to the max desired.
 current[i] = ratio_max[i];
      }
    else
{
 // Set the current ratio to new ratio.
 current[i] = 2*ratio_desired[i] - ratios[i];
 // Control that we have at least the minimum.
 if (current[i] < ratio_min[i])
   current[i] = ratio_min[i];
      }
   // Set the new ratio for neword.
   current[NEWORDER] -= current[i];
  }
     }
   else
     {
// We are still in the Ramp-Up window.
// Loop for all other transactions.
for (i = 1; i < 6; i++)
   {
    // Set the current ratio to desired we want to have.
    current[i] = ratio_desired[i];
   }
     }
/* Add up all the users */
   /* This needs to be changed to be more transparent */
   // Reset the total number of users.
   shmglobal->total_users = 0;
   // Loop for all slaves.
   for (i = 0; i < MAX_SLAVES; i++)
{
 shmglobal->total_users += slave_status[i].active;
 desired                += slave_status[i].desired;
      }
   /* Count up number of warehouses we WANT to have */
   if (num_warehouses < 0)
     {
num_warehouses = (desired -1 ) / 10 + 1;
      }
   shmglobal->max_warehouses = num_warehouses;
   // Loop for all transactions except neword.
   for (i = 2; i < 6; i++)
{
 if (current[i] != last[i])
   different = 1;
     }
   // Don't send if it's the same as last time
   if ( !different && shmglobal->total_users == users_last )
     {
// time(&clock);
// iprint(IPRINT_TRACE, "Date : %s\n", ctime(&clock));
// iprint(IPRINT_TRACE, "shmglobal->total_users = %d\n", shmglobal->total_users);
return RTE_ERROR;
```

```
      }
   // Set the last user number.
   users_last = shmglobal->total_users;
   // Loop for all transactions except neword.
   for (i = 1; i < 6; i++)
{
 shmglobal->chances[i] = last[i] = current[i];
      }
   // iprint(IPRINT_INFO, "send_global_data with :\n");
   // iprint(IPRINT_INFO, "\t chances = ");
   // for (i = 0; i < MAX_TRAN_TYPE; i++)
   //    iprint(IPRINT_INFO, "%6.2f ", shmglobal->chances[i]);
// Send global data to the slaves..
   send_global_data();
return RTE_OK;
  }
/*********************************************************
 *     FUNCTION : parse_array
 *     PURPOSE :
 *     INPUT :
 *           string :
 *           max    :
 *           buffer :
 *     OUTPUT :
 *           no one
 *     RETURN :
 *     DESCRIPTION :
*********************************************************/
int parse_array(char *string, int max, int *buffer)
  {
   int   i,
rc;
   char   *ptr;
   char   *temp = strdup(string);
   ptr = strtok(temp, ",");
   for (i = 0; ptr && i < max; i++)
{
 rc = sscanf(ptr, "%d", &buffer[i]);
 if (rc < 1)
   {
    free(temp);
    return i;
   }
 ptr = strtok(NULL, ",");
      }
   free(temp);
   return i;
  }
/*********************************************************
 *     FUNCTION : parse_array
 *     PURPOSE :
 *     INPUT :
 *           string :
 *           max    :
 *           buffer :
 *     OUTPUT :
 *           no one
 *     RETURN :
 *     DESCRIPTION :
*********************************************************/
int parse_array(char *string, int max, double *buffer)
  {
   int   i,
rc;
   char   *ptr;
   char   *temp = strdup(string);
   ptr = strtok(temp, ",");
   for (i = 0; ptr && i < max; i++)
{
 rc = sscanf(ptr, "%lf", &buffer[i]);
 if (rc < 1)
   {
    free(temp);
    return i;
   }
ptr = strtok(NULL, ",");
      }
   free(temp);
   return i;
  }
/*********************************************************
 *     FUNCTION : user_init
 *     PURPOSE :
 *  Get the configuration parameters in the rteparams  configuration  file.
 *     INPUT :
 *           no one
 *     OUTPUT :
 *           no one
 *     RETURN :
 *           no one
 *     DESCRIPTION :
 *     RTE_OK or exit.
*********************************************************/
int user_init()
  {
   double  dbuffer[32];
   int    rc,
i;
   char *ptr;
   // Get the KEYSTROKE_SLEEP value in the
rteparams config file.
   if (get_variable("KEYSTROKE_SLEEP", &shmglobal->keystroke_sleep)
!= RTE_OK)
      {
// If not in the rteparams config file set to 0.
shmglobal->keystroke_sleep = 0;
      }
   // Get the LOGIN_TIMEOUT value in the rteparams config file.
   if (get_variable("LOGIN_TIMEOUT", &shmglobal->login_timeout)
!= RTE_OK)
      {
// If not in the rteparams config file set to 2 minutes.
shmglobal->login_timeout = 120;
      }
   // Get the KEYSTROKE_PACKET_SIZE value in the rteparams config file.
   if (get_variable("KEYSTROKE_PACKET_SIZE", &shmglobal-
>keystroke_packet_size)
!= RTE_OK)
```

```
      {
// If not in the rteparams config file set to 0.
shmglobal->keystroke_packet_size = 0;
      }
   // Set the login timeout in ticks.
   shmglobal->login_timeout *= 1000;

   // Get the WAREHOUSES value in the rteparams config file.
   if (get_variable("WAREHOUSES", &num_warehouses) != RTE_OK)
     {
      // If not in the rteparams config file set to not used.
num_warehouses = -1;
      }
   iprint(IPRINT_INFO, "Login Timeout = %s\n", mstoa(shmglobal->login_timeout, 0));
   iprint(IPRINT_INFO, "Keystroke Sleep = %s\n", mstoa(shmglobal-
>keystroke_sleep*1000, 0));
   iprint(IPRINT_INFO, "Keystroke Packet Size= %d\n", shmglobal-
>keystroke_packet_size);
   if (num_warehouses >= 0) {
iprint(IPRINT_INFO, "Fixed Warehouses to = %d\n", num_warehouses);
   }
   // Get the NEWORDER string in the rteparams config file.
   if (!(ptr = get_variable("NEWORDER")))
     {
iprint_error ("Error. NEWORDER variable not
found\n");
exit (1);
      }
   // Control there are 3 values in the NEWORDER
string.
   if (parse_array(ptr, 3, dbuffer) != 3)
     {
iprint_error ("Error. NEWORDER should be think, emulex_menu,
emulex_response");
exit (1);
      }
   // Set the think time.
   shmglobal->think          [NEWORDER] = dbuffer[0];
   // Set the emulex menu time.
   shmglobal->emulex_menu    [NEWORDER] =
dbuffer[1];
   // Set the emulex menu response time.
   shmglobal->emulex_response[NEWORDER] =
dbuffer[2];
   // Loop for all transactions except neword.
   for (i = 2; i < 6; i++)
{
 // Get the transaction string and control there are for
each 6 values.
 if (!(ptr = get_variable(ratio_names[i])) ||
     (parse_array(ptr, 6, dbuffer)!=6))
   {
    iprint(__FILE__, __LINE__, IPRINT_ERROR,
      "Error. %s should be think, emulex_delay,
desired, min, max",
       ratio_names[i]);
    exit (1);
   }
// Set the think time.
shmglobal->think          = dbuffer[0];
// Set the emulex menu time.
shmglobal->emulex_menu[i]     = dbuffer[1];
// Set the emulex menu response time.
shmglobal->emulex_response[i] = dbuffer[2];
// Set the ratio desired.
ratio_desired[i]              = dbuffer[3];
// Set the minimum ratio to exceed.
ratio_min[i]                  = dbuffer[4];
// Set the maximum ratio to reach.
ratio_max[i]                  = dbuffer[5];
// Set the range ratio to work fine.
ratio_range[i]                = ratio_max[i] - ratio_min[i];
      }
   return RTE_OK;
  }
/*************************************************************
*     FUNCTION : user_extra_data
*     PURPOSE :
*     INPUT :
*     OUTPUT :
 *           no one
*     RETURN :
 *           no one
*     DESCRIPTION :
 *           RTE_OK or exit.
*************************************************************/
int user_extra_data(header_s *header)
  {
   int    i,
num_timestamps;
   if (header->data_type !=
RTE_ITEM_KEYSTROKE_TIMES)
return RTE_OK;
   int *times = (int *)((char *)header+sizeof(struct
header_s));
   num_timestamps = header->user_data_length / 4 - 1;
   iprint(IPRINT_TRACE, "Keystroke times = ");
   for (i = 0 ; i < num_timestamps; i++)
{
 iprint (IPRINT_TRACE, "%d ", times[i]);
      }
   iprint(IPRINT_TRACE, "\n", times[i]);

   return RTE_OK;
  }
/*************************************************************
*     FUNCTION : user_process_command
*     PURPOSE :
*     INPUT :
 *           command :
*     OUTPUT :
 *           no one
*     RETURN :
 *           no one
*     DESCRIPTION :
 *           RTE_OK or exit.
*************************************************************/
int user_process_command(char *command)
```

```
  {
   char   buffer[256],
*ptr;
   int  i,
found,
len;
   strncpy (buffer, command, 256);
   ptr = strtok (buffer, " \t");
   found = 0;
   if (!strcasecmp (ptr, "display"))
     {
      while (ptr && (ptr = strtok(NULL, " \t")))
  {
         if (*ptr == '\0')
            continue;
         for (i = 0; i < 5; i++)
{
            len = min(strlen(Status_Names[i]), strlen(ptr));
            if (!strncasecmp (ptr, Status_Names[i], len))
   {
               status_needs_refresh = found = 1;
               current_status = i;
               return RTE_OK;
              }
            }
         printf ("Unknown type to display: %s\n", ptr);
        }
     }
   printf ("Unknown Command: %s\n", command);
   return RTE_ERROR;
  }
int user_begin()
  {
   return RTE_OK;
  }
void user_make_header(char *buffer)
  {
   int     i;
   struct user_data_header *data = (struct
user_data_header *)buffer;
  }
```

D.3 RTE User

user_slave.c

```
/*********************************/
/*      INCLUDES FILES          */
/*********************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/time.h>
#include "rte_slave.h"
#include "user_tpcc.h"
/*********************************/
/*      EXTERNAL VARIABLES      */
/*********************************/
extern SHM_Slave    *shm;
extern TableEntrySlave   *shmentry;
extern DriverStatus      *status;
extern char      *expect_save;
/*********************************/
/*      EXTERNAL FUNCTIONS      */
/*********************************/
extern echo_trace    (char *);
extern echo_trace   ();
/*********************************/
/*      CONSTANTS DEFS  */
/*********************************/
const char *SQL_TPERRNO_MESSAGE = "tperrno";
const char *SQL_RTN_MESSAGE =     "rtn:";
const char *SQL_FATAL_MESSAGE =   "SQL Fatal Error";
const char *ROLLBACK_MESSAGE =    "Item number is not valid";
/*********************************/
/*      INTERNAL VARIABLES      */
/*********************************/
int WHSEID;      /* warehouse number for each users */
/*******************************************************************************
 *      FUNCTION : NURand
 *      PURPOSE :
 *    The "uniform()" function has range of the absolute value of the
 *    difference between the min. and the max values upto 2147483647.
 *    This Non-Uniform Random function is defined according to the
 *    2.1.4, 2.1.5 and 2.1.6 clauses.
 *      INPUT :
 *            A: 255 for C_LAST, 1023 for C_ID, 8191 for OL_I_ID
 *            x: 0 for C_LAST, 1 for C_ID and OL_I_ID
 *            y: 999 for C_LAST, 3000 for C_ID, 100000 for OL_I_ID
 *     OUTPUT :
 *           no one
 *     RETURN :
 *           value.
 *     DESCRIPTION :
*******************************************************************************/
long NURand(int A, int x, int y, long cval)
  {
   return ((((long) uniform((long) 0, (long) A) | (long) uniform((long) x, (long) y)) +
cval) % (y - x + 1)) + x;
  }
/*******************************************************************************
 *      FUNCTION : getname
 *      PURPOSE :
 *    Generates a random number from 0 to 999 inclusive
 *    a random name is generated by associating a random
 *    string with each digit of the generated number
 *    three strings are concatenated to generate lastname.
 *      INPUT :
 *            no one
 *     OUTPUT :
 *            no one
 *     RETURN :
 *            string.
 *     DESCRIPTION :
 *
 *******************************************************************************/
```

```
char *getname()
  {
   char *last_name_parts[] =
     {
      "BAR",
      "OUGHT",
      "ABLE",
      "PRI",
      "PRES",
      "ESE",
      "ANTI",
      "CALLY",
      "ATION",
      "EING"
     };
   char lastname[128];
   int  random_num;

   random_num = NURand(255, 0, 999, LASTC);
   strcpy(lastname, last_name_parts[random_num / 100]);
   random_num %= 100;
   strcat(lastname, last_name_parts[random_num / 10]);
   random_num %= 10;
   strcat(lastname, last_name_parts[random_num]);
   return (lastname);
  }
/*******************************************************************************
 *     FUNCTION : Delivery
 *     PURPOSE :
 *           Made the exchanged automate for the delivery transaction
 *           between the slave and the client and generate Input Data
 *           for Terminal I/O according to the 2.7.1 clause.
 *     INPUT :
 *           no one
 *     OUTPUT :
 *           no one
 *     RETURN :
 *           RTE_OK, RTE_ERROR or ERROR.
*     DESCRIPTION :
*******************************************************************************/
int Delivery()
  {
   static struct delivery_struct delivery, // Delivery structure.
  delivery_new; // Delivery structure.
   char   carrier[32]; // Carrier id.
   int   resp,  // Return code.
  charlen;   // Length of the carrier id.
   double  think_max;  // Max time for thinking.
  /* think_DEL; */
   char const   *ptr; // Return pointer.
   // Set invalid transactions to zero.
   delivery_new.invalid = 0;
   // Set the carrier number from 1 to 10 according to the 2.7.1.2 clause.
   delivery_new.carrier = uniform(1, 10);
   sprintf(carrier, "%d", delivery_new.carrier);
   // Set typing delay to 0 sec.
   set_typing_delay(0);
   // Sleep before acting the transaction.
   transaction_sleep_do();
   // Start the transaction, set that it is in progress and log for analyze later.
   transaction_start(DELIVERY, sizeof(delivery), &delivery);
   delivery = delivery_new;
   echo_trace("Waiting for Menu (DELIVERY)");
   // Send code for client to generate a delivery transaction to server.
   transmit("4\n");
   // Wait in return the screen position for delivery transaction.
   resp = expect("_\033=%0");
   // Control the return code.
   if (resp == ERROR)
     {
iprint (IPRINT_ERROR, "Slave %d: Failed to receive Delivery screen\n",
shmentry->num);
return(ERROR);
     }
   // Set typing delay according to the length of the carrier id.
   charlen = strlen(carrier);
   set_typing_delay(2000 / charlen + 1);
#ifdef EMULEX_P6000
   // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
   // terminal concentrators used in the priced configuration.
   usleep(shmglobal->emulex_menu[DELIVERY]*1000000.0+0.9);
#endif
   // End of menu, beginning of keying time.
   transaction_mark(WHERE_NOW);
   echo_trace("Keying");
   transmit(carrier);
   // End of keying, beginning of response time.
   transaction_mark(WHERE_NOW);
   echo_trace("Wait for Response");
   // Set typing delay to 0 sec.
   set_typing_delay(0);
   // Wait for the response.
   resp = expect("Delivery has been queued\033=! ");
   // Control the return code.
   if (resp == ERROR)
     {
iprint (IPRINT_ERROR, "Slave %d: Failed to receive Delivery response\n",
shmentry->num);
return (ERROR);
     }
#ifdef EMULEX_P6000
   // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
   // terminal concentrators used in the priced configuration.
   usleep(shmglobal->emulex_response[DELIVERY]*1000000.0+0.9);
#endif
// End of response, beginning of think time.
   transaction_mark(WHERE_NOW);
   // Control if we will received a TUXEDO error message.
   if (expect_after_match(SQL_TPERRNO_MESSAGE))
     {
delivery.invalid = 1;
if (ptr = expect_after_match(SQL_RTN_MESSAGE))
  {
   iprint (IPRINT_ERROR, "Slave %d: Delivery found '%s'\n",
         shmentry->num, ptr);
  }
else
  {
```

```
    iprint (IPRINT_ERROR, "Slave %d: Delivery found '%s'\n",
            shmentry->num, SQL_TPERRNO_MESSAGE);
   }
return RTE_ERROR;
      }
    // Control if we will received a SQL error message.
    if (expect_after_match (SQL_FATAL_MESSAGE))
      {
delivery.invalid = 1;
iprint (IPRINT_ERROR, "Slave %d: Delivery found '%s'\n", shmentry->num,
SQL_FATAL_MESSAGE);
return RTE_ERROR;
      }
    echo_trace ("Thinking");
    // Set the new time to wait before the next transaction.
    transaction_sleep_set(neg_exp_4(shmglobal->think[DELIVERY])*1000.0);
    return(RTE_OK);
   }
/*******************************************************************************
*      FUNCTION : NewOrder
*      PURPOSE :
 *             Made the exchanged automate for the new order transaction
 *             between the slave and the client and generate Input Data
 *            for Terminal I/O according to the 2.4.1 clause.
 *      INPUT :
 *             no one
*      OUTPUT :
 *             no one
*      RETURN :
 *             RTE_OK, RTE_ERROR or ERROR.
*      DESCRIPTION :
*******************************************************************************/
int NewOrder()
  {
    static struct neword_struct neword,   // New order structure.
neword_new;    // New order structure.
    int   Rbtrans,   // Rollback transactions.
remoteflag,     // Remote orders flag.
i,        // Local variable.
resp,      // Return code.
charlen,   // Length of buffer.
whses,     // Max nb of warehouses.
low_whse = 1;   // Ident of warehouse.
    double    think_max;   // Max time for thinking.
/* think_NO; */
    char   buff[1024],   // Buffer.
district[32],   // District id.
customer[32],    // Customer id.
warehouse[32],   // Warehouse id.
item[32],     // Item id.
quantity[32];   // Quantity id.
    char const *ptr;   // Return pointer.
    // Set rollback and invalid transactions to zero.
    neword_new.rollback = 0;
    neword_new.invalid  = 0;

    // Section to determine ROLLBACK transaction for 1% of new orders according
    // to the 2.4.1.4 clause.
    RBtrans = 0;
    if (uniform(1, 5000) <= 50)
    // if (uniform(1.0, 100.0) <= 1.0)
Btrans = 1;
    // Set the district number according to the 2.4.1.2 clause.
    neword_new.did = uniform(1, 10);
    sprintf(district, "%d\t", neword_new.did);
    strcpy(buff, district);
    // Set the customer number from 1 to 3000 according to the 2.4.1.2 clause.
    neword_new.cid = NURand(1023, 1, 3000, CUSTC);
    sprintf(customer, "%d", neword_new.cid);
    strcat(buff, customer);
    // Set the number of loop from 5 to 15 iterations according to the
    // 2.4.1.3 clause.
    neword_new.nloop    = uniform(5, 15);
    // Set the remote orders flag.
    remoteflag          = 0;
    // Set the find total number of remote order-lines.
    neword_new.olremote = 0;
    // Set the max number of warehouses.
    whses = shmglobal->max_warehouses;
// Loop on number of iterations.
    for (i = 0; i < neword_new.nloop; i++)
{
 // Set the warehouse id according to the 2.4.1.5.2 clause specifying
 // that 99% of the time it is a local warehouses and 1% a remote.
 neword_new.item[i].olswid = WHSEID;
 // Control the max number of warehouses.
 if (whses > 1 && (uniform(0.0, 100.0) < 1.0))
    {
     /* For 1% of items we must generate a uniform whse number that's
        different from WHSEID of the local user. */
     while (neword_new.item[i].olswid == WHSEID)
   {
    neword_new.item[i].olswid =
    (long) uniform((long) low_whse, (long)whses);
   }
     // Set the find total number of remote order-lines.
     neword_new.olremote++;
     // Set the remote orders flag.
     remoteflag = 1;
    }
 sprintf(warehouse, "\t%d\t", neword_new.item[i].olswid);
 strcat(buff, warehouse);
 // If last iteration of item and must rollback set item number to
 // unused according to the 2.4.1.5.1 clause.
 if (neword_new.nloop == i+1 && RBtrans)
     // And a RollBack transaction.
     neword_new.item[i].oliid = 999999;
 else
    {
     /* From item 1 to 100,000 */
     neword_new.item[i].oliid = NURand(8191, 1, 100000, ITEMC);
    }
sprintf(item, "%d\t", neword_new.item[i].oliid);
strcat(buff, (item));
// Set quantity from 1 to 10 according to the 2.4.1.5.3 clause.
neword_new.item[i].olquantity = uniform(1, 10);
sprintf(quantity, "%d", neword_new.item[i].olquantity);
strcat(buff, quantity);
```

```
    }   /* end of for n_loop */
  // Control the remote orders flag.
  if (remoteflag == 1)
neword_new.oremote = 1;
  else
neword_new.oremote = 0;
  // Sleep before acting the transaction.
  transaction_sleep_do();
  // Start the transaction, set that it is in progress and set the
  // TSTMP_MENUBEGIN timestamp.

  transaction_start(NEWORDER, sizeof(neword), &neword);

  neword = neword_new;
  echo_trace ("Waiting for Menu (NEWORDER)");
  // Set typing delay to 0 sec.
  set_typing_delay(0);

  // Send code for client to generate a new order transaction to server.
  // It correspond to the menu option 1.
  transmit("1\n");

  // Wait in return the screen position for new order transaction.
  resp = expect("\033=#<");
  // Control the return code.
  if (resp == ERROR)
    {
iprint (IPRINT_ERROR, "Slave %d: Failed to receive NewOrder screen\n",
shmentry->num);
return (ERROR);
    }
#ifdef EMULEX_P6000
  // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
  // terminal concentrators used in the priced configuration.
  usleep(shmglobal->emulex_menu[NEWORDER]*1000000.0+0.9);
#endif
  // End of menu, beginning of keying time set the TSTMP_MENUEND
timestamp.
  transaction_mark(WHERE_NOW);
  echo_trace ("Keying");
  // Set typing delay according to the length of the buffer.
  charlen = strlen(buff);
  set_typing_delay(18000 / charlen + 1);

  // Send the new order buffer to the client with keystroke timing or not.
  if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME)
    {
iprint (IPRINT_INFO, "Neword transaction: Keystroke timing call
transmit_timed()\n");
transmit_timed(buff);
    }
  else
    {
transmit(buff);
    }
  // End of keying, beginning of response time set the TSTMP_KEYDONE
  // timestamp.
transaction_mark(WHERE_NOW);
  echo_trace ("Wait for Response");
  // Set typing delay to 0 sec.
  set_typing_delay(0);
  // Wait for the response.
  resp = expect("\033=! ");
  // Control the return code.
  if (resp == ERROR)
    {
iprint(IPRINT_ERROR, "Slave %d: Failed to receive New Order response\n",
shmentry->num);
return (ERROR);
    }
#ifdef EMULEX_P6000
  // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
  // terminal concentrators used in the priced configuration.
  usleep(shmglobal->emulex_response[NEWORDER]*1000000.0+0.9);
#endif
  // End of response, beginning of think time set the TSTMP_RESPONSE
  // timestamp.
  transaction_mark(WHERE_NOW);
  /* Grab order id from return string */
  if (!(ptr = expect_after_match("\033=%.")))
    {
echo_trace ("Didn't find order-id for neworder");
iprint (IPRINT_ERROR, "Neworder didn't have Order-ID");
neword.oid = -1;
    }
  else
    {
neword.oid = atoi(ptr+4);
    }
  // Control if we will received a rollback message.
  if (expect_after_match(ROLLBACK_MESSAGE))
    {
neword.rollback = 1;
echo_trace ("Found rollback!\n");
    }
  // Control if we will received a TUXEDO error message.
  if (expect_after_match (SQL_TPERRNO_MESSAGE))
    {
neword.invalid = 1;
if (ptr = expect_after_match (SQL_RTN_MESSAGE))
  {
    iprint (IPRINT_ERROR, "Slave %d: New Order found '%s'\n",
          shmentry->num, ptr);
  }
else
    {
    iprint (IPRINT_ERROR, "Slave %d: New Order found '%s'\n",
          shmentry->num, SQL_TPERRNO_MESSAGE);
  }
return RTE_ERROR;
    }
  // Control if we will received a SQL error message.
  if (expect_after_match (SQL_FATAL_MESSAGE))
    {
iprint (IPRINT_ERROR, "Slave %d: New Order found '%s'\n", shmentry->num,
SQL_FATAL_MESSAGE);
neword.invalid = 1;
echo_trace ("Fatal Error Message");
```

```
return RTE_ERROR;
      }
   echo_trace ("Thinking");

   // Set the new time to wait before the next transaction.
   transaction_sleep_set(neg_exp_4(shmglobal->think[NEWORDER])*1000.0);

   // Control if keystroke time file used.
   if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME)
     {
iprint (IPRINT_INFO, "Neword transaction: Keystroke timing call log_data()\n");
log_data(RTE_ITEM_KEYSTROKE_TIMES,
keystroke_length*sizeof(int),keystroke_times);
      }
return (RTE_OK);
  }
/*************************************************************
 *     FUNCTION : OrderStatus
 *     PURPOSE :
 *           Made the exchanged automate for the orderstat transaction
 *           between the slave and the client and generate Input Data
 *           for Terminal I/O according to the 2.6.1 clause.
*     INPUT :
 *           no one
*     OUTPUT :
 *           no one
*     RETURN :
 *           RTE_OK, RTE_ERROR or ERROR.
*     DESCRIPTION :
*****************************************************************************/
int OrderStatus()
  {
   static struct ordstat_struct ordstat,      // Ordstat structure
 ordstat_new;  // Ordstat structure
   char      buff[1024],   // Buffer for sending to client.
 district[32], // District id.
 customer[32]; // Customer id.
   int            resp,         // Return code.
 charlen;      // Length of buffer.
   double          think_max;    // Max time for thinking.
 /* think_OS; */
   char const  *ptr;         // Return pointer

   // Set invalid transactions to zero.
   ordstat_new.invalid = 0;

   // Set district number from 1 to 10 according to the 2.6.1.2 clause.
   ordstat_new.did = uniform(1, 10);
   sprintf(district, "%d\t", ordstat_new.did);
   strcpy(buff, district);
// Set customer number from 60% of time by last name and 40% of time by number
   // according to the 2.6.1.2 clause.
   if (uniform(1, 100) <= 60)
    {
      // Set by customer last name.
strcpy(ordstat_new.clast, getname());
if (ordstat_new.clast[0] < 'A' || ordstat_new.clast[0] > 'Z')
        {
   iprint (IPRINT_ERROR, "ASSERTION: OrderStatus getname() returns invalid
name! '%s'\n", ordstat_new.clast);
  }
sprintf(customer, "\t%s", ordstat_new.clast);
// Set byname flag to OK.
ordstat_new.byname = 1;
      // Set cid  to 0.
ordstat_new.cid = 0;
      }
   else
      {
       // Set by customer by number from 1 to 3000.
ordstat_new.cid = NURand(1023, 1, 3000, CUSTC);
sprintf(customer, "%d\t", ordstat_new.cid);
       // Set byname flag to NOK.
ordstat_new.byname = 0;
       // Set clast to NULL.
ordstat_new.clast[0] = (char) NULL;
      }
   strcat(buff, customer);
   set_typing_delay(0);
   transaction_sleep_do();
transaction_start (ORDSTAT, sizeof(ordstat), &ordstat);  * logging */

   ordstat = ordstat_new;
   echo_trace ("Waiting for Menu (ORDSTAT)");
   transmit("3\n");   /* menu option 3 */
   resp = expect("_\033=#<");    /* screen position */
   if (resp == ERROR) {
iprint (IPRINT_ERROR, "Slave %d: Failed to receive Order Status screen\n",
shmentry->num);
return (ERROR);
   }
   charlen = strlen(buff);
   set_typing_delay(2000 / charlen + 1);
#ifdef EMULEX_P6000
   // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
   // terminal concentrators used in the priced configuration.
   usleep(shmglobal->emulex_menu[ORDSTAT]*1000000.0+0.9);
#endif
   transaction_mark(WHERE_NOW);  /* end of menu, beginning of keying
 * time */
   echo_trace ("Keying");
   transmit(buff);

   transaction_mark(WHERE_NOW);   /* end of keying, beginning of
 * response time */
   echo_trace ("Wait for Response");

   set_typing_delay(0);
   transmit("\r");

   resp = expect("\033=! ");   /* screen position */
   if (resp == ERROR) {
iprint (IPRINT_ERROR, "Slave %d: Failed to receive Order Status response\n",
shmentry->num);
return (ERROR);
   }
#ifdef EMULEX_P6000
```

```
    // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
    // terminal concentrators used in the priced configuration.
    usleep(shmglobal->emulex_response[ORDSTAT]*1000000.0+0.9);
#endif
    transaction_mark(WHERE_NOW);   /* end of response, beginning of
 * think time */
    if (expect_after_match (SQL_TPERRNO_MESSAGE)) {
ordstat.invalid = 1;
if (ptr = expect_after_match (SQL_RTN_MESSAGE)) {
    iprint (IPRINT_ERROR, "Slave %d: Order status found '%s'\n",
          shmentry->num, ptr);
} else  {
    iprint (IPRINT_ERROR, "Slave %d: Order status found '%s'\n",
          shmentry->num, SQL_TPERRNO_MESSAGE);
           }
return RTE_ERROR;
    }
    if (expect_after_match (SQL_FATAL_MESSAGE)) {
iprint (IPRINT_ERROR, "Slave %d: Order Status found '%s'\n", shmentry->num,
SQL_FATAL_MESSAGE);
ordstat.invalid = 1;
return RTE_ERROR;
    }
    echo_trace ("Thinking");
    transaction_sleep_set(neg_exp_4(shmglobal->think[ORDSTAT])*1000.0);
    return (RTE_OK);
  }
/*****************************************************************************
 *     FUNCTION : Payment
 *     PURPOSE :
 *           Made the exchanged automate for the payment transaction
 *           between the slave and the client and generate Input Data
 *           for Terminal I/O according to the 2.5.1 clause.
 *     INPUT :
 *           no one
 *     OUTPUT :
 *           no one
 *     RETURN :
 *           RTE_OK, RTE_ERROR or ERROR.
 *     DESCRIPTION :
*****************************************************************************/
int Payment()
  {
    static struct payment_struct payment,       // Payment structure
                          payment_new;   // Payment structure
    int                    dollars,       // Dollars
                          cents,         // Cents
                          resp,          // Return code.
                          charlen,       // Length of buffer.
                          whses,         // Max nb of warehouses.
                          low_whse = 1;  // Ident of warehouse.
    double                   think_max;     // Max time for thinking.
                          /* think_PAY; */
    char                   buff[1024],    // Buffer for sending to client.
                          district[32],  // District id.
                          customer[32],  // Customer id.
                          waredist[32],  // Warehouse id
                          amount[32];    // Amount.
    char                   const *ptr;    // Return pointer
    // Set invalid transactions to zero.
    payment_new.invalid = 0;

    // Set district number from 1 to 10 according to the 2.5.1.2 clause.
    payment_new.did = uniform(1, 10);
    sprintf(district, "%d\t", payment_new.did);
    strcpy(buff, district);
    // Set customer number from 60% of time by last name and 40% of time by
number
    // according to the 2.5.1.2 clause.
    if (uniform(1, 100) <= 60)
      {
       // Set by customer last name.
strncpy(payment_new.clast, getname(), 17);
if (payment_new.clast[0] < 'A' || payment_new.clast[0] > 'Z')
          {
    iprint (IPRINT_ERROR, "ASSERTION: payment_new getname() returns
invalid name! '%s'\n", payment_new.clast);
    exit (10);
   }
sprintf(customer, "%s\t", payment_new.clast);
// Set byname flag to OK.
        payment_new.byname = 1;
        // Set cid  to 0.
payment_new.cid = 0;
      }
    else
      {
       // Set by customer by number from 1 to 3000.
payment_new.cid = NURand(1023, 1, 3000, CUSTC);
sprintf(customer, "%d\t", payment_new.cid);
       // Set byname flag to NOK.
payment_new.byname = 0;
       // Set clast to NULL.
payment_new.clast[0] = (char) NULL;
      }
if (payment_new.byname)   /* using C_LAST */
strcat(buff, "\t");
    else          /* using C_ID */
strcat(buff, customer);
    // Set the warehouse number.
    whses = shmglobal->max_warehouses;
    // For 85 % of transactions.
    if (whses < 2 || uniform(1, 100) <= 85)
      {
       // Set customer warehouse number to local.
payment_new.cwid = WHSEID;
       // Set customet district number to its own.
payment_new.cdid = payment_new.did;
       // Set remote flag to NOK.
payment_new.remote = 0;
/*
 * Pwarehousesame++;                total number of home
 * transactions
 */
      }
    else
      {
```

```
/* For 15 % of transactions */
payment_new.cwid = WHSEID;
while (payment_new.cwid == WHSEID)
          {
           // Set customer warehouse number other than local.
    payment_new.cwid = (long) uniform((long)low_whse, (long) whses);
    /* warehouse 1 to max whses */
  }
       // Set remote flag to OK.
payment_new.remote = 1;
       // Set customer district number from 1 to 10.
payment_new.cdid = uniform(1, 10);
/*
 * Pwarehousediff++;                total number of remote
 * transactions
 */
      }

    sprintf(waredist, "%d\t%d\t", payment_new.cwid, payment_new.cdid);
    strcat(buff, waredist);

    if (payment_new.byname)   /* using C_LAST */
strcat(buff, customer);
    else             /* using C_ID */
strcat(buff, "\t");
    // Set the amount unit from 1 to 5000 $ according to the 2.5.1.3 clause and
the cents
    // unit from 0.00 to 0.99 according to the 2.5.1.3 clause.
    dollars = uniform(1, 5000);
    if (dollars == 5000)
cents = 0;
    else
cents = uniform(0, 99);
    payment_new.amount = ((double) dollars) + ((double) cents) / 100.0;
    sprintf(amount, "%d.%2.2d", dollars, cents);
    strcat(buff, amount);    /* 1.00 to 5000.00 */
    set_typing_delay(0);
    transaction_sleep_do();
    transaction_start(PAYMENT, sizeof(payment), &payment); /* logging */
    payment = payment_new;
    echo_trace ("Waiting for Menu (PAYMENT)");
    transmit("2\n");         /* menu option 2 */
    resp = expect("_\033=%S");  /* screen position */
    if (resp == ERROR) {
iprint (IPRINT_ERROR, "Slave %d: Failed to receive Payment screen\n",
shmentry->num);
return (ERROR);
    }
    charlen = strlen(buff);
    set_typing_delay(3000 / charlen + 1);
#ifdef EMULEX_P6000
    // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
    // terminal concentrators used in the priced configuration.
    usleep(shmglobal->emulex_menu[PAYMENT]*1000000.0+0.9);
#endif
    transaction_mark(WHERE_NOW);  /* end of menu, beginning of keying
 * time */
    echo_trace ("Keying");
    transmit(buff);
    transaction_mark(WHERE_NOW);   /* end of keying, beginning of

    echo_trace ("Wait for Response");       set_typing_delay(0);

    transmit("\r");

    resp = expect("\033=! ");   /* screen position */
    if (resp == ERROR) {
iprint (IPRINT_ERROR, "Slave %d: Failed to receive Payment response\n",
shmentry->num);
return (ERROR);
    }
#ifdef EMULEX_P6000
    // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
    // terminal concentrators used in the priced configuration.
    usleep(shmglobal->emulex_response[PAYMENT]*1000000.0+0.9);
#endif

    transaction_mark(WHERE_NOW);  /* end of response, beginning of

    if (expect_after_match (SQL_TPERRNO_MESSAGE)) {
payment.invalid = 1;
if (ptr = expect_after_match (SQL_RTN_MESSAGE)) {
    iprint (IPRINT_ERROR, "Slave %d: Payment status found '%s'\n",
          shmentry->num, ptr);
} else {
    iprint (IPRINT_ERROR, "Slave %d: Payment status found '%s'\n",
          shmentry->num, SQL_TPERRNO_MESSAGE);
}
return RTE_ERROR;
    }
    if (expect_after_match (SQL_FATAL_MESSAGE)) {
iprint (IPRINT_ERROR, "Slave %d: Payment found '%s'\n", shmentry->num,
SQL_FATAL_MESSAGE);
payment.invalid = 1;
return RTE_ERROR;
    }
    echo_trace ("Thinking");
    transaction_sleep_set(neg_exp_4(shmglobal->think[PAYMENT])*1000.0);
    return (RTE_OK);
  }
/*******************************************************************************
 *     FUNCTION : StockLevel
 *     PURPOSE :
 *           Made the exchanged automate for the stocklevel transaction
 *           between the slave and the client and generate Input Data
 *           for Terminal I/O according to the 2.5.1 clause.
 *     INPUT :
 *           no one
 *     OUTPUT :
 *           no one
 *     RETURN :
 *           RTE_OK, RTE_ERROR or ERROR.
 *     DESCRIPTION :
*******************************************************************************/
int StockLevel()
  {
    static struct stocklev_struct stocklevel,      // Stocklevel structure
```

```
                        stocklevel_new;  // Stocklevel structure
    char                  threshold[32];   // Threshold
    int                   resp,            // Return code.
                          charlen;         // Length of buffer.
    double                  think_max;       // Max time for thinking.
                          /* think_SL; */
    char                  const *ptr;     // Return pointer

    // Set invalid transactions to zero.
    stocklevel_new.invalid = 0;
    // Set threshold of minimum quantity in stock from 10 to 20 according to the
2.8.1.2
    // clause.
    stocklevel_new.threshold = uniform(10, 20);
    sprintf(threshold, "%d", stocklevel_new.threshold);

    set_typing_delay(0);
    transaction_sleep_do();

    transaction_start(STOCKLEV, sizeof(stocklevel), &stocklevel); /* logging */

    stocklevel = stocklevel_new;
    echo_trace ("Waiting for Menu (STOCKLEV)");
    transmit("5\n");   /* menu option 5 */
    resp = expect("_\033=%7");  /* screen position */
    if (resp == ERROR) {
iprint (IPRINT_ERROR, "Slave %d: Failed to receive Stock Level screen\n",
shmentry->num);
return (ERROR);
    }
    charlen = strlen(threshold);
    set_typing_delay(2000 / charlen + 1);
#ifdef EMULEX_P6000
    // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
    // terminal concentrators used in the priced configuration.
    usleep(shmglobal->emulex_menu[STOCKLEV]*1000000.0+0.9);
#endif

    transaction_mark(WHERE_NOW);  /* end of menu, beginning of keying
 * time */
    echo_trace ("Keying");
    transmit(threshold);

    transaction_mark(WHERE_NOW);  /* end of keying, beginning of
 * response time */
    echo_trace ("Wait for Response");
    set_typing_delay(0);

    transmit("\r");

    resp = expect("\033=! ");   /* screen position */
    if (resp == ERROR) {
iprint (IPRINT_ERROR, "Slave %d: Failed to receive Stock Level response\n",
shmentry->num);
return (ERROR);
    }
#ifdef EMULEX_P6000
    // Waiting time to simulate latency due to real Emulex P6000-88 Ethernet
    // terminal concentrators used in the priced configuration.
    usleep(shmglobal->emulex_response[STOCKLEV]*1000000.0+0.9);
#endif

    transaction_mark(WHERE_NOW);  /* end of response, beginning of
 * think time */
    if (expect_after_match (SQL_TPERRNO_MESSAGE)) {
stocklevel.invalid = 1;
if (ptr = expect_after_match (SQL_RTN_MESSAGE)) {
    iprint (IPRINT_ERROR, "Slave %d: Stock Level status found '%s'\n",
            shmentry->num, ptr);
} else {
    iprint (IPRINT_ERROR, "Slave %d: Stock Level status found '%s'\n",
            shmentry->num, SQL_TPERRNO_MESSAGE);
}
return RTE_ERROR;
    }
    if (expect_after_match (SQL_FATAL_MESSAGE)) {
iprint (IPRINT_ERROR, "Slave %d: Stock Level found '%s'\n", shmentry->num,
SQL_FATAL_MESSAGE);
stocklevel.invalid = 1;
return RTE_ERROR;
    }
    echo_trace ("Thinking");
    transaction_sleep_set(neg_exp_4(shmglobal->think[STOCKLEV])*1000.0);
    return (RTE_OK);
  }
/*******************************************************************************
 *     FUNCTION : user_transaction
 *     PURPOSE :
 *           Made the exchanged automate for all the 5 transactions
 *           between the slave and the client.
 *     INPUT :
 *           no one
 *     OUTPUT :
 *           no one
 *     RETURN :
 *           RTE_OK, RTE_ERROR or ERROR.
 *     DESCRIPTION :
*******************************************************************************/
int user_transaction()
  {
    char           logout[32];
    double          ntask;
    int            resp, i;

    if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME)
      {
int rc;
/* Wait for specified period of time */
sleep (shmglobal->keystroke_sleep);
/* Quit after one transaction */
shm->lock(shmentry->pid);
shmentry->flags |= TES_FLAG_DIE;
shm->unlock(shmentry->pid);
rc = NewOrder();
iprint (IPRINT_INFO, "Slave %d: Keystroke timing setting die flag\n", shmentry-
>num);
return rc;
      }
```

```
/*************************************************************************/
   /***  CHOOSE ONE OF THE TRANSACTIONS                                  ***/
   /*************************************************************************/
// iprint(IPRINT_INFO,"\n");
   // for (i=0; i<MAX_TRAN_TYPE; i++)
     // iprint(IPRINT_INFO,"%6.2f ", shmglobal->chances[i]);

   ntask = (double) uniform(0.0, 100.0);
   if (ntask <= shmglobal->chances[DELIVERY])
return Delivery();

   ntask -= shmglobal->chances[DELIVERY];
   if (ntask <= shmglobal->chances[ORDSTAT])
return OrderStatus();
   ntask -= shmglobal->chances[ORDSTAT];
   if (ntask <= shmglobal->chances[PAYMENT])
return Payment();

   ntask -= shmglobal->chances[PAYMENT];
   if (ntask <= shmglobal->chances[STOCKLEV])
return StockLevel();
   return NewOrder();
#if 0
   if (resp != RTE_OK)
     {          /* logoff if response is not correct */
strcpy(logout, "9\r");          /* menu option 9 */
transmit(logout);
resp = expect("tpcc_cstux_inf:");
return (ERROR);
      }
   else
return (RTE_OK);
#endif
  }          /* end of main */
int user_parameter_change(void)
  {
#if 0
   int i;
   iprint(IPRINT_TRACE, "Slave %d: total_users = %d\n", shmentry->num);
   iprint(IPRINT_TRACE, "Slave %d: chances     = ");
   for (i = 0; i < MAX_TRAN_TYPE; i++)
iprint(IPRINT_TRACE, "%6.2f ", shmglobal->chances[i]);
   iprint(IPRINT_TRACE, "\nSlave %d: think       = ");
   for (i = 0; i < MAX_TRAN_TYPE; i++)
iprint(IPRINT_TRACE, "%6.2f ", shmglobal->think[i]);
   iprint(IPRINT_TRACE, "\n");
#endif
   return RTE_OK;
  }
const int MAX_LOGIN_RETRY = 500;
int user_login(char *user, char *password, void *data)
  {
   extern char ttyname_original[TTYNAME_LEN];
   UserLocal   *localdata = (UserLocal *)data;
   int   rc;
   int   retry = 0;
   int   timeout_value = shmglobal->login_timeout;
   char   buffer[256];
   set_typing_delay(0);
   iprint(IPRINT_INFO, "shmglobal->login_timeout = %d\n", shmglobal-
>login_timeout);

   // Loop for login on the client according to the number of retry.
   do
    {
     // Control the number of retry.
     if (++retry > MAX_LOGIN_RETRY)
{
 // MAX_LOGIN_RETRY reach so break.
 iprint(IPRINT_ERROR, "Slave %d: Failed expecting 'login: '\n", shmentry-
>num);
 return RTE_ERROR;
}
     // Send the CR+LF to the client.
     rc = transmit("\r\n");
     // Wait for the login from client with timeout.
     rc = expect("login: ", timeout_value);
     // Verify the return code.
     if (rc == RTE_ERROR)
{
 iprint (IPRINT_ERROR, "Slave %d: didn't find login
prompt 'login: '\n", shmentry->num);
 }
    }
   while (rc == RTE_ERROR);
   // Send the user login to client.
   /* strcat(user, "\n"); */
   rc = transmit(user);
   // Send the CR to client.
   rc = transmit("\n");
   // Wait for the password from client with timeout.
   rc = expect("Password:", timeout_value);
   // Verify the return code.
   if (rc != RTE_OK)
     {
      iprint (IPRINT_ERROR, "Slave %d: Failed expecting 'Password:'\n",
shmentry->num);
return RTE_ERROR;
     }
   // Send the password.
   /* strcat(password, "\n"); */
   rc = transmit(password);
   // Send the CR to client.
   rc = transmit("\n");
   // Wait for the prompt from client without timeout.
   rc = expect("$");
   // Verify the return code.
   if (rc != RTE_OK)
     {
      iprint (IPRINT_ERROR, "Slave %d: Failed expecting '$ '\n", shmentry-
>num);
return RTE_ERROR;
     }
#if 0
   sprintf(buffer, "exec nice -n 5 start_test %d %d\n",
         localdata->Warehouse, localdata->District);
#else
   // Get the command to exec on the client which is the main process.
```

```
    sprintf(buffer, "exec start_test %d %d\n",
            localdata->Warehouse, localdata->District);
#endif
    // Send the command to exec on the client.
    rc = transmit(buffer);
// Wait for Exit from client with timeout for clearing menu.
    rc = expect("Exit", timeout_value);
// Verify the return code.
    if (rc != RTE_OK)
       {
       iprint (IPRINT_ERROR, "Slave %d: Failed expecting 'Exit'\n",
shmentry->num);
return RTE_ERROR;
       }
    return RTE_OK;
   }
int user_init()
   {
    extern int expect_save_active;
    WHSEID               = shmlocal->Warehouse;
    status->max_transmit = shmglobal->keystroke_packet_size;
    expect_save_active   = 1;
    return RTE_OK;
   }
int user_cleanup()
   {
    // Just to control is there is other thing to do before exiting.
    transaction_sleep_do();
    // Just something to clear out the buffer.
    transaction_start(0, 0, NULL);

    return RTE_OK;
   }

// CM : Add 14/09/95 for correctly terminating.
int user_bye_bye()
   {
    char           logout[3];
    double          ntask;
    int            resp,
    tm_out = 60000;          // 60 Seconds.
    strcpy(logout, "9\r");
    // Send code for client to log off for cleaning.
    // It correspond to the menu option 9.
    transmit(logout);
    // Wait in return for the `User End' acknowledgment.
    resp = expect("User End", tm_out);
    // Control the return code.
    if (resp == ERROR)
       {
        iprint (IPRINT_ERROR, "Slave %d: Failed to exit user\n", shmentry->num);
        return (ERROR);
       }

    return(RTE_OK);
   }
```

user_tpcc.h

```
/*********************************/
/*      CONSTANTS DEFS           */
/*********************************/
#ifndef USER_TPCC_H
#define USER_TPCC_H
/*****************************************************************************/
/***   run-time constant for customer last name from 0 to 255,          ***/
/***   run-time constant for customer id from 0 to 1023,                ***/
/***   run-time constant for item id from 0 to 8191.                    ***/
/*****************************************************************************/
#define   LASTC     117
#ifdef AVANT_FRAAB
#define  CUSTC   319
#else
#define CUSTC   23
#endif
#define  ITEMC   3849
/*****************************************************************************/
/***   response type   ***/
/*****************************************************************************/
/* #define OK  1  */
/* #define ERROR   -1 */
/*****************************************************************************/
/***   transaction type   ***/
/*****************************************************************************/
#define  NEWORDER    1
#define  PAYMENT    2
#define  ORDSTAT    3
#define  DELIVERY    4
#define  STOCKLEV   5
/*****************************************************************************/
/***   transaction structures      ***/
/*****************************************************************************/
struct neword_struct {
    char   invalid;  /* transaction completed sucessfully */
    long    did;
    long    cid;
    long    oid;  /* Order-ID returned from client */
    long    nloop;  /* number of order line, avg = 15 */
    char    oremote;  /* 1 for remote order, 10% */
    long    olremote;  /* number of remote order line, 1% */
    char    rollback;  /* actually saw rollback text on screen */
    struct items_struct {
long            olswid;
long            oliid;
long            olquantity;
    } item[15];
};
struct payment_struct {
    char   invalid;  /* transaction completed sucessfully */
    long    did;
    long    cid;
    long    cwid;
    long    cdid;
    char   clast[17];
    double    amount;
    char    byname;  /* 1 for by last name, 0 for by id */
    char    remote;    /* 1 for remote wardhouse, 0 otherwise */
};
```

```
struct ordstat_struct {
    char    invalid;   /* transaction completed sucessfully */
    long    did;
    long    cid;
    char    clast[17];
    char     byname;   /* 1 for by last name, 0 for by id */
};

struct delivery_struct {
    char   invalid;   /* transaction completed sucessfully */
    char   carrier;
};

struct stocklev_struct {
    char  invalid;   /* transaction completed sucessfully */
    long   threshold;
};

struct generic_struct {
    char invalid;   /* transaction completed sucessfully */
};

union transaction_info {
    char     invalid;
    struct generic_struct  generic;
    struct neword_struct   neword;
    struct payment_struct  payment;
    struct ordstat_struct  ordstat;
    struct delivery_struct delivery;
    struct stocklev_struct stocklev;
};

struct UserGlobal {
    int   total_users;
    int   max_warehouses;
    int   keystroke_sleep;
    int   login_timeout;
    int   keystroke_packet_size;
    double    chances  [MAX_TRAN_TYPE];
    double    think      [MAX_TRAN_TYPE];
    double    emulex_response   [MAX_TRAN_TYPE];
    double    emulex_menu       [MAX_TRAN_TYPE];
};
struct UserLocal {
    int Warehouse;
    int District;
};
struct user_data_header {
};
extern UserGlobal *shmglobal;
extern UserLocal  *shmlocal;
#endif
```

Appendix E: Disk Storage

The calculations used to determine the storage requirements for the 8 hour logical log are contained in these appendix.

180 DAY SPACE CALCULATION AND TOTAL DISK REQUIREMENT

Note : Numbers are in KBytes unless otherwise specified

Warehouses	620	**tpmC**	7303.67	**tpmC/W**	11.78

Table	**Rows**	**Data**	**Index**	**5% Space**	**8H Space**	**Total Space**
Warehouse	620	1 240	10	63		1 313
District	6 200	12 400	54	623		13 077
Item	100 000	9 524	86	192		9 802
New-order	5 633 856	62 328	736		12 400	75 464
History	18 603 539	1 041 824	0		157 981	1 199 805
Orders	18 627 929	504 680	6 056		77 447	588 183
Customer	18 600 000	12 400 000	1 037 288	268 746		13 706 034
Order-line	186 030 124	11 275 750	736		1 709 954	12 986 440
Stock	62 000 000	20 898 656	229 628	422 566		21 550 850
Totals		46 206 402	1 274 594	692 189	1 957 782	50 130 967

Segment	**LogDev Cnt.**	**Seg. Size**	**Needed**	**Overhead**	**Not Needed**
wdino	1	325 632	100 652	1 007	224 980
history	1	1 587 200	1 211 803	12 118	375 397
order	1	1 058 816	594 065	5 941	464 751
customer	30	14 223 360	13 843 094	138 431	380 266
order_line	10	17 141 760	13 116 304	131 163	4 025 456
stock	60	21 872 640	21 766 358	217 664	106 282
Totals		56 209 408	50 632 277	506 323	5 577 131

Dynamic space	12 424 764	Sum of Data for Order, Order-Line and History (excluding free extents)
Static space	36 254 743	Data + Index + 5% Space + Overhead - Dynamic space
Free space	1 952 769	Total Seg. Size - Dynamic Space - Static Space - Not Needed
Daily growth	2 341 842	(Dynamic space/W * 62.5)* tpmC
Daily spread	(1 559 994)	Free space - 1.5 * Daily growth (zero if negative)
180 day (KB)	457 786 301	Static space + 180 (daily growth + daily spread)
180 day (GB)	436.58	Excludes OS, Paging and RDBMS Logs
Log per N-O txn	1.81	Number of 2K blocks per New-Order transaction
8 Hour Log (GB)	12.10	*2 for mirrors

DISKS PRICED	**Count**	**Size (MB)**	**Capacity (GB)**	**SPACE USAGE**	**(GB)**
SSA	60	4296	251.72	180-day	436.58
SCSI	60	4088	239.53	Swap	2.50
				OS, ...	2.10
TOTAL	**120**		**491.25**	**TOTAL**	441.18
SSA	4	4296	16.78	Log(mirror)	24.20
SCSI	4	4088	15.97		
TOTAL	**8**		**32.75**		
TOTAL	**128**		**524.00**	**TOTAL**	**465.38**