IBM RISC System/6000 Workgroup Server F50 c/s

using

Sybase Adaptive Server Enterprise 11.5

TPC Benchmark™ C

Full Disclosure Report

IBM System Performance and Evaluation Center

Submitted For Review
February 12, 1998

Special Notices

The following terms used in this publication are trademarks of **International Business Machines** Corporation in the United States and/or other countries:

RISC System/6000
AIX
IBM

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

TPC Benchmark	Trademark of the Transaction Processing Performance Council
Sybase Adaptive Server Enterprise	Trademark of Sybase, Inc.
Sybase Open Client	Trademark of Sybase, Inc.
Tuxedo System/T	Trademark of BEA Systems, Inc.

First Edition May 12, 1997

Request for additional copies of this document should be sent to the following address:

TPC Benchmark Administrator
IBM System and Performance Evaluation Center
Mail Stop 9221
11400 Burnet Road
Austin, TX 78758
FAX Number (512) 838-1852

IBM RS/6000 Workgroup Server F50 c/s

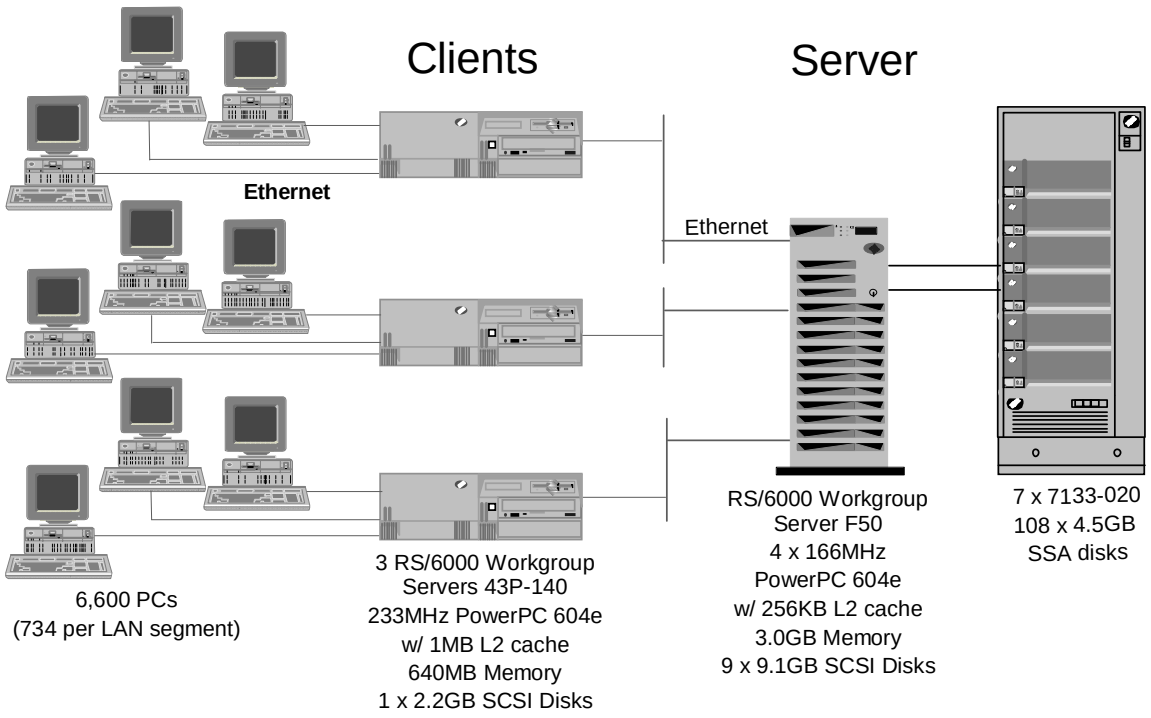
TPC-C Rev. 3.3

Report Date: May 12, 1997
Pricing Update: Feb. 12, 1998

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$510,643	**8,142.4 tpmC**	**$62.71/tpmC**	**September 30,1997***

Processors	Database Manager	Operating System	Other Software	Number of Users
4 x 166MHz IBM PowerPC 604e	Sybase Adaptive Server Enterprise 11.5	AIX 4.2.1	Tuxedo 6.2 Sybase Open Client /C	6,600

RS/6000 Workgroup Server F50



*Sybase Adaptive Server Enterprise 11.5 available September 30, 1997, All other components available today.

System Components	Clients		Server	
	Quantity	Description	Quantity	Description
Processor	3	233MHz PowerPC 604e w/ 1MB L2 cache	4	166MHz PowerPC 604e w/ 256KB L2 cache
Memory	640MB each		3.0GB	
Disk Controllers	3	SCSI-2 Adapters	3 3	SCSI-2 Adapters SSA Adapters
Disk Drives	3	2.2GB SCSI	108 9	4.5GB SSA Disks 9.1 SCSI F/W Disks
Total Storage	2.2 GB each client		529.31 GB	
Terminals	3	System Console	1	System Console
Term. Connect	303	CentreCOM 24 Port Ethernet Hubs + 10% spares		

IBM RS/6000 Workgroup Server F50 c/s

TPC-C Rev. 3.3

Report Date: May 12, 1997
Pricing Update: Feb. 12, 1998

Description	Part Number	Third Party Brand	Third Party Pricing	Unit Price	Qty	Extended Price	5 yr. Maint Price
Server Hardware							
RS/6000 Server Model F50	7025-F50		1	19,900	1	19,900	11,952
1 604e, 128mb Memory, 4,5GB Disk,							
2 Intg SCSI2 F/W Adptr, CDROM							
604e 2way Proc Card Select, 2-256kb L2	4303		1	4,000	1	4,000	3,120
604e 2way Proc Card, 2-256kb L2	4309		1	8,000	1	8,000	6,240
256MB DIMM Memory Select	4106		1	3,200	1	3,200	0
256MB DIMM Memory Modules	4110		1	6,400	11	70,400	0
Memory Expansion Feature 2nd Card	4093		1	1,038	1	1,038	0
9.1GB SCSI-2 F/W Hot Swap Disk	2911		1	3,850	8	30,800	0
9.1GB SCSI-2 F/W Hot Swap Disk Sel	3019		1	1,800	1	1,800	0
SCSI Hot Swap 6-Pack Kits	6519		1	750	1	750	0
16-bit Integrated SCSI Adapter Cable	2444		1	66	1	66	0
PCI Ethernet Adapter	2985		1	195	2	390	0
PCI SCSI-2 F/W Adapter	6208		1	360	1	360	0
PCI SSA 4port RAID Adapter	6215		1	3,000	3	9,000	0
System Rack Model R00	7015-R00		1	3,110	2	6,220	2,976
SSA Disk Subsystem w/ 4 4.5GB Disk	7133-020		1	19,350	7	135,450	67,200
4.5 GB Disk Drive Modules	3401		1	2,100	80	168,000	0
SSA Cables	5010		1	40	14	560	0
				Subtotal		459,934	91,488
Server Software							
AIX 4.2.1 F50 + Support	5765-C34		1	240	1	240	0
Adaptive Server Enterprise 11.5 incl Client		Sybase	2	35,995	1	35,995	28,000
				Subtotal		36,235	28,000
Client Hardware							
RS/6000 Model 43P-140, 233 MHz	7043-140		1	8,000	3	24,000	10,080
2.1gb Disk, Intg SCSI-2 FW, Intg Enet							
128MB DIMM Memory	4102		1	1,280	3	3,840	0
128MB DIMM Memory Expansion	4115		1	2,560	12	30,720	0
Async Terminal/Printer Cable	2934		1	45	4	180	0
Ethernet Adapter, PCI	2985		1	195	9	1,755	0
				Subtotal		60,495	10,080
Client Software							
AIX 4.1.5 Unlimited Users	5756-C34		1	3,660	3	10,980	0
Tuxedo EPT Ver 6.2, 43P Client		BEA	3	3,000	3	9,000	6,750
				Subtotal		19,980	6,750
User Connectivity							
Ethernet Hub (24port)	AT3024TR	CentreCOM	4	215.95	275	59,386	0
Ethernet Hub 10% Spares	AT3024TR	CentreCOM	4	215.95	28	6,047	0
IBM ASCII Terminal, Keyboard	3153-BG3		1	577	4	2,308	2,640
					Subtotal	67,741	2,640
					Discounts	(240,281)	(32,419)
					Total	**404,104**	**106,539**

Notes:
Pricing Sources:
1=Dickens Data, 2=Sybase, 3=BEA Systems, 4=Data Comm Warehouse
Audited by: Francois Raab, Information Paradigm

Five-Year Cost of Ownership: 510,643
tpmC Rating: 8,142.40
$/tpmC: 62.71

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

Numerical Quantities Summary for the IBM RS/6000 F50

MQTH, computed Maximum Qualified Throughput: 8,142.4 tpmC

Repeatability Run: 8,090.03 tpmC 0.6% difference

Response Times (in seconds)	**90th Percentile**	**Average**
New Order	1.54	0.66
Payment	1.45	0.58
Order-Status	1.52	0.7
Delivery (interactve)	0.61	0.27
Delivery (deferred)	2	1.14
Stock-Level	2.51	1.2
Menu	0.01	0.01

Transaction Mix, in percent of total transactions	**Percent**
New Order	44.59%
Payment	43.26%
Order-Status	4.08%
Delivery	4.02%
Stock-Level	4.02%

Keying/Think Times (in seconds)	**Min.**	**Average**	**Max.**
New Order	18.00/0.01	18.00/12.13	18.11/121.00
Payment	3.00/0.01	3.00/12.12	3.10/121.00
Order-Status	2.00/0.01	2.00/10.12	2.11/101.00
Delivery	2.00/0.01	2.00/5.12	2.10/51.00
Stock-Level	2.00/0.01	2.00/5.10	2.08/45.71

Test Duration	
Ramp-up Time	34 min 16 sec
Measurement interval	30 minutes
Transactions during measurement interval (all types)	547,709
Ramp-down time	15 minutes

Checkpointing	
Number of checkpoints	1
Checkpoint interval	30

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification Revision 3.3 dated April 8, 1997, for measurements on the IBM RISC System/6000 Workgroup Server F50. The phrase RS/6000 will be substituted for RISC System/6000 for the remainder of this document.

The software used on the RS/6000 Workgroup Server F50 includes AIX Version 4.2.1 operating system, Sybase Adaptive Server Enterprise Version 11.5 database manager, and Tuxedo System/T Version 6.2 transaction manager.

IBM RISC System/6000 Workgroup Server F50

Company Name	System Name	Data Base Software	Operating System Software
IBM	RS/6000 Workgroup Server F50	Sybase Adaptive Server Enterprise 11.5	AIX Version 4.2.1
Availability Date: September 30, 1997			

Total System Cost	TPC-C Throughput	Price/Performance
-Hardware -Software -5 Years Maintenance	Sustained maximum throughput of system running TPC-C expressed in transactions per minute	Total system cost/tpmC ($510,643/8,142.4)
$510,643	**8,142.40 tpm-C**	**$62.71 per tpm-C**

Preface

TPC Benchmark™ C Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on August 13, 1992 and updated with revision 3.3 on April 8, 1997.

This is the full disclosure report for benchmark testing of the IBM RS/6000 Workgroup Server F50 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 span a breadth of complexity
- On-line and deferred transaction execution modes
- Multiple on-line terminal sessions
- Moderate system and application execution time
- Significant disk input/output
- Transaction integrity (ACID properties)
- Non-uniform distribution of data access through primary and secondary keys
- 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, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-C should not be used as a substitiute for a specific customer application benchmarking when crititcal 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 processing an order, and each transaction is subject to a response time constraint. The performance metric for this benchmark is expressed in transactions-per-minute-C (tpmC). To be compliant with the TPC-C standard, all references to tpmC results must include the tpmC rate, the associated price-per-tpmC, and the availability date of the priced configuration.

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 the RS/6000 application code for the five TPC Benchmark™ C transactions. Appendix D contains the terminal functions and layouts.

1.2 Benchmark Sponsor

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

This benchmark was sponsored by **International Business Machines** Corporation.

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:

- *Data Base tuning options*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and application configuration parameters.*

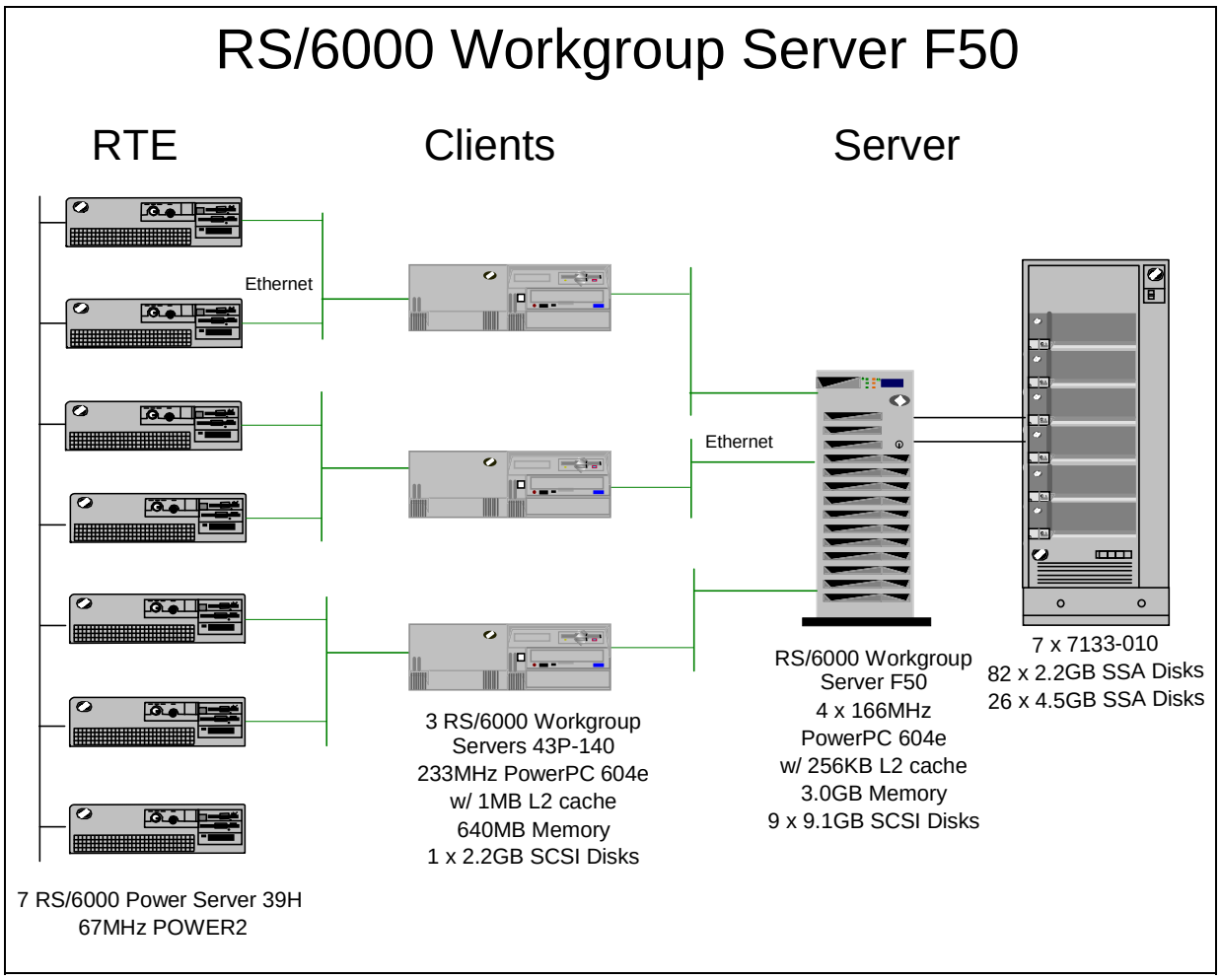
Appendix B contains the system, data base, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

1.4 Configuration Diagrams

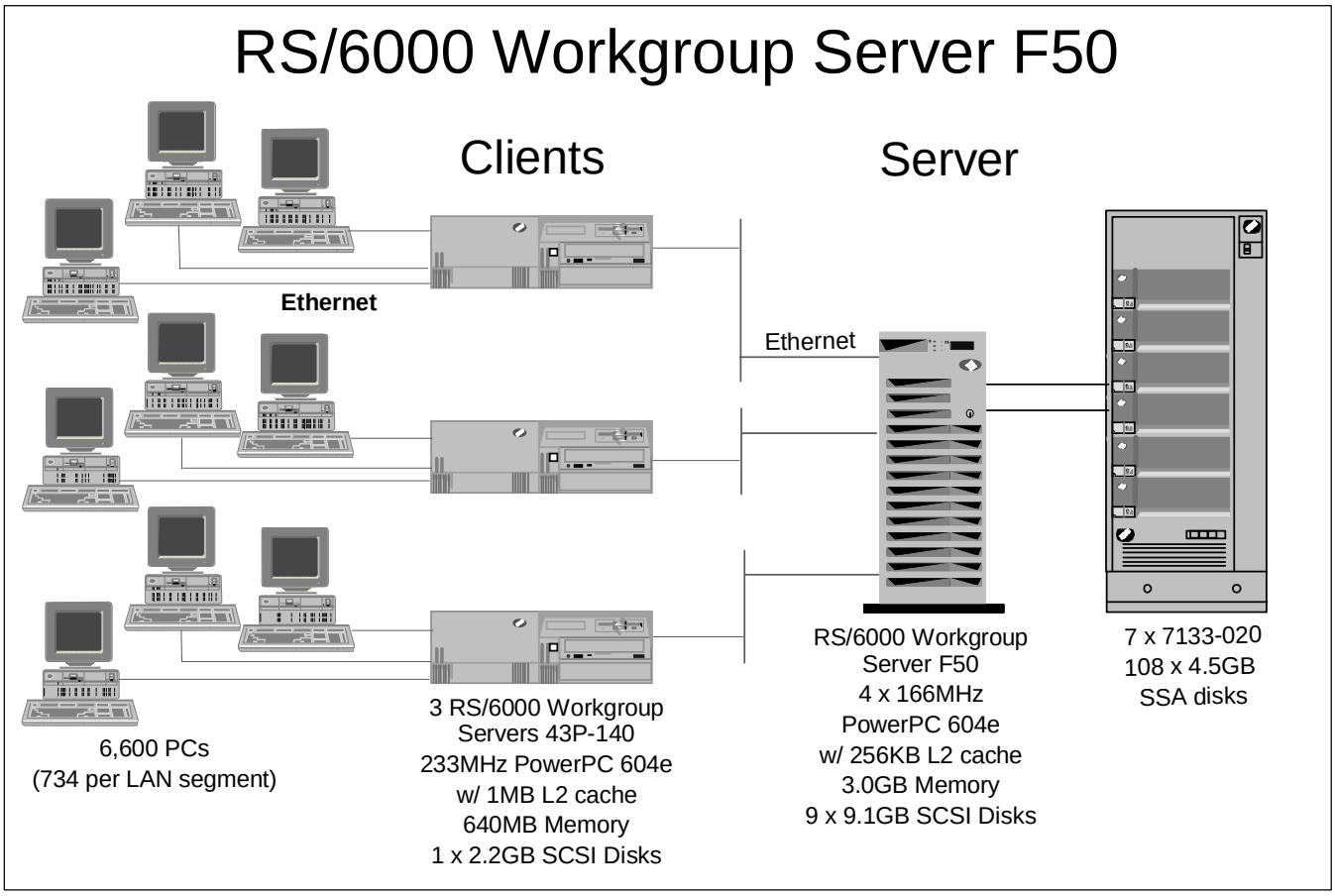
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *Number and type of processors*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test*
- *Number and type of disk units (and controllers, if applicable)*
- *Number of channels or bus connections to disk units, including the protocol type*
- *Number of LAN (e.g. Ethernet) connections, including routers, work stations, terminals, etc, that were physically used in the test or are incorporated into the pricing structure (see Clause 8.1.8)*
- *Type and run-time execution location of software components (e.g. DBMS, client processes, transaction monitors, software drivers, etc)*

RISC System/6000 Workgroup Server F50 Benchmark Configuration



RISC System/6000 Workgroup Server F50 Priced Configuration



2. Clause 1: Logical Data Base Design Related Items

2.1 Table Definitions

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

Appendix C contains the table definitions and the database load programs used to build the data base.

2.2 Database Organization

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

Physical space was allocated to Sybase Adaptive Server Enterprise on the server disks according to the details provided in section C.1 in Appendix C. The size of the space segments on each disk was calculated to provide even distribution of data across the disk subsystem. Clustered indicies were defined on all tables except the history table. In addtion, a non-clustered index was defined on the Customer table. The indices were defined at table definition and were built at the initial table load by executing the database build script in section C.2 of Appendix C.

2.3 Insert and/or 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. Futhermore, any restriction in the SUT data base implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows.

There were no restrictions on insert and/or delete operations to any of the tables. The space required for an additional five percent of the initial table cardinality was allocated to Sybase Adaptive Server Enterprise and priced as static space.

2.4 Horizontal or Vertical Partitioning

While there are 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 for any of the measurement reported in this full disclosure.

3. Clause 2: Transaction and Terminal Profiles Related Items

3.1 Verification for the Random Number Generator

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

The srandom(), getpid() and 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

double neg_exp_4(double average {
                return - average * (1/RANDOM_4_Z * log (1 - RANDOM_4_K * drand48()));
        }
```

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 random values produced by the database generation program. After a measurement, the HISTORY, ORDER, and ORDER_LINE tables were queried to verify the randomness of values generated by the driver. The rows were counted and grouped by customer and item numbers.

Here is an example of one SQL query 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 Input/Output Screens

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

Appendix D contains the screen layouts for all transactions corresponding to the specification in clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3 and 2.8.3.

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).

The emulated workstations, IBM RS/6000 Model 43P-100s, are commercially available and support all of the requirements in Clause 2.2.2.4.

3.4 Presentation Managers

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

The RS/6000 Model 43P-140 workstations did not involve screen presentations, message bundling or local storage of TPC-C rows. All screen processing was handled by the client system. All data manipulation was handled by the server system.

3.5 Home and Remote Order-lines

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

Tables 3-1 show the percentage of home and remote transactions that occurred during the measurement period for the New-Order transactions.

3.6 New-Order Rollback Transactions

The percentage of New-Order transactions that were rolled back as a result of an illegal item number must be disclosed.

Tables 3-1 show the percentage of New-Order transactions that were rolled back due to an illegal item being entered.

3.7 Number of Items per Order

The number of items per order entered by New-Order transactions must be disclosed.

Tables 3-1 show the average number of items ordered per New-Order transaction.

3.8 Home and Remote Payment Transactions

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

Tables 3-1 show the percentage of home and remote transactions that occurred during the measurement period for the Payment transactions.

3.9 Non-Primary Key Transactions

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

Tables 3-1 show the percentage of non-primary key accesses to the data base by the Payment and Order-Status transactions.

3.10 Skipped Delivery Transactions

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

Tables 3-1 show the percentage of Delivery transactions missed due to a shortage of supply of rows in the NEW-ORDER table.

3.11 Mix of Transaction Types

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

Tables 3-1 show the mix percentage for each of the transaction types executed by the SUT.

3.12 Queueing Mechanism of Delivery

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

The Delivery transaction was 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 data base transaction for deferred execution. Please see the application code in Appendix A for details.

Table 3-1 Numerical Quantities for Transaction and Terminal Profiles

New Order	RS/6000 Workgroup Server F50
Percentage of Home order lines	99.01%
Percentage of Remote order lines	0.99%
Percentage of Rolled Back Transactions	0.99%
Number of Items per order	9.99
Payment	
Percentage of Home transactions	85.00%
Percentage of Remote transactions	15.00%
Non-Primary Key Access	
Percentage of Payment using C_LAST	60.00%
Percentage of Order-Status using C_LAST	60.12%
Delivery	
Delivery transactions skipped	0
Transaction Mix	
New-Order	44.59%
Payment	43.26%
Order-Status	4.08%
Delivery	4.02%
Stock-Level	4.02%

4. Clause 3: Transaction and System Properties

The results of the ACID test must be disclosed along with a description of how the ACID requirements were met.

All ACID tests were conducted according to specification.

4.1 Atomicity Requirements

The system under test must guarantee that data base 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.

4.1.1 Atomicity of Completed Transaction

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

The following steps were performed to verify the atomicity of completed transactions.

1. The customer balance was retrieved from the CUSTOMER table and ytd balances were retrieved from the DISTRICT and WAREHOUSE tables for a random Customer, District and Warehouse.
2. The Payment transaction was executed for the Cutomer, District and Warehouse used in step 1.
3. The customer balance and ytd balances were retrieved again for the same Customer, District and Warehouse used in step 1. It was verified that the balances changed appropriately.

4.1.2 Atomicity of Aborted Transactions

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number) 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 steps were performed to verify the atomicity of the aborted Payment transaction:

1. The customer balance was retrieved from the CUSTOMER table and ytd balances were retrieved from the DISTRICT and WAREHOUSE tables for a random Customer, District and Warehouse
2. The Payment application code was changed to execute a rollback of the transaction instead of performing the commit. The Payment transaction was executed for the Customer, District and Warehouse used in step 1.
3. The customer balance and ytd balances were retrieved again for the same Customer, District and Warehouse used in step 1. It was verified that the balances before the execution of the Payment transaction and the balances after the execution of the Payment transaction were the same.

4.2 Consistency Requirements

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

Verify that the data base is initially consistent by verifying that it meets the consistency conditions defined in Clauses 3.3.2.1 to 3.3.2.4. Describe the steps used to do this in sufficient detail so that the steps are independently repeatable.

The queries defined in 4.2.1 through 4.2.4 were run after initial data base build and prior to executing any transactions. All queries showed that the data base was in a consistent state.

After executing transactions at full load for approximately sixty minutes the queries defined in 4.2.1 through 4.2.4 were run again. All queries showed that the data base was still in a consistent state.

4.2.1 Consistency Condition 1

Entries in the WAREHOUSE and DISTRICT tables must satisfy the relationship:

- *W_YTD = sum(D_YTD)*

for each warehouse defined by (W_ID = D_W_ID)

The following SQL query was executed before and after a measurement to show that the data base was always in a consistent state both initially and after a measurement.

```
Select D_W_ID, W_YTD, SUM(D_YTD), W_YTD_SUM(D_YTD)
        from DISTRICT, WAREHOUSE
        where D_W_ID = W_ID
        group by D_W_ID, W_YTD
        order by D_W_ID;
```

4.2.2 Consistency Condition 2

Entries in the DISTRICT, ORDER, and NEW-ORDER tables must satisfy the relationship:

- *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). This condition does not apply to the NEW-ORDER table for any districts which have no outstanding new orders.

The following SQL queries were executed before and after a measurement to show that the data base was always in a consistent state both initially and after a measurement.

```
Create view consist
    as (Select D_W_ID, D_ID, D_NEXT_O_ID -1 as onext, MAX(O_ID) as omax
            from DISTRICT, ORDERS
            where D_ID = O_D_ID and D_W_ID = O_W_ID
            group by D_W_ID, D_ID,
            )
      Select * from consist where onext != omax

Create view consist
    as ( Select D_W_ID, D_ID, D_NEXT_O_ID -1 as onext, MAX(NO_O_ID) as omax
            from DISTRICT, NEW_ORDER
            where D_ID = NO_D_ID and D_W_ID = NO_W_ID
            group by D_W_ID, D_ID,
            )
Select * from consist where onext != omax
```

4.2.3 Consistency Condition 3

Entries in the New-Order table must satisfy the relationship:

- *max(NO_O_ID) - min(NO_O_ID) + 1 = [number of rows in the New-Order table for this district]*

for each district defined by NO_W_ID and NO_D_ID. This condition does not apply to any districts which have no outstanding new orders.

The following SQL query was executed before and after a measurement to show that the data base was always in a consistent state both initially and after a measurement.

```
Create view consist
as (Select COUNT(*) as nocount, (MAX(NO_O_ID) - MIN(NO_O_ID) +1) as total
        from NEW_ORDER,
        group by NO_W_ID, NO_D_ID
        )

Select * from consist where nocount != total


Crteate table temp_w (T_W_ID smallint)
   Insert into temp_w Select round(MAX(W_ID* 0.1,0) from warehouse
   Insert into temp_w Select round(MAX(W_ID * 0.3,0) from warehouse
   Insert into temp_w Select round(MAX(W_ID * 0.6,0) from warehouse
   Insert into temp_w Select round(MAX(W_ID * 1.0,0) from warehouse
```

4.2.4 Consistency Condition 4

Entries in the ORDER and ORDER-LINE tables must satisfy the relationship:

- *sum(O_OL_CNT) = [number of rows in the ORDER-LINE table for this district]*

for each district defined by (O_W_ID = OL_W_ID) and (O_D_ID = OL_D_ID).

The following SQL queries was executed before and after a measurement to show that the data base was always in a consistent state both initially and after a measurement.

```
Create view consist1
as (Select O_W_ID,O_D_ID,SUM(O_OL_CNT) as ol_sum
    from orders,temp_w
    where O_W_ID=T_W_ID
    group by O_W_ID,O_D_ID
    )

Create view consist2
as (Select OL_W_ID,OL_D_ID,COUNT(*) as ol_count
    from ORDER_LINE,temp_w
    where O_W_ID=T_W_ID
    group by OL_W_ID,OL_D_ID
     )

Select * from consist1, consist2
where O_W_ID=OL_W_ID
and O_D_ID=OL_D_ID
and ol_sum != ol_count
```

4.3 Isolation Requirements

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

4.3.1 Isolation Test 1

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

The following steps were performed to satisfy the test of isolation for Order-Status and New-Order transactions:

1. 1st terminal: Start a New-Order transaction with the necessary inputs. The transaction is delayed for 20 seconds just prior to the Commit.
2. 2nd terminal: Start an Order-Status transaction for the same customer as used in the New-Order transaction.
3. 2nd terminal: The Order-Status transaction attempts to read the file but is locked out by the New-Order transaction waiting to complete.
4. 1st terminal: The New-Order transaction is released and the Commit is executed releasing the record. With the CUSTOMER record now released, the Order-Status transaction can now complete.
5. 2nd terminal: Verify that the Order-Status transaction delays for approximately 20 seconds and that the results displayed for the Order-Status transaction match the input for the New-Order transaction.

4.3.2 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 following steps were performed to satisfy the test of isolation for Order-Status and a rolled back New-Order transactions:

1. 1st terminal: Perform a New-Order transaction with the necessary inputs. The New_order application does a rollback instead of a commit. The transaction is delayed for 20 seconds just prior to the Rollback.
2. 2nd terminal: Start an Order-Status transaction for the same customer used in the New-Order transaction. The Order-Status transaction attempts to read the CUSTOMER table but is locked by the New-Order transaction.
3. 1st terminal: The New-Order transaction rolls back. With the CUSTOMER record now released, the Order-Status transaction completes.
4. Verify that the Order-Status transaction delays for approximately 20 seconds..

4.3.3 Isolation Test 3

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

The following steps were performed to verify isolation of two New-Order transactions:

1. 1st terminal: Start a New-Order transaction using the necessary inputs. The transaction is delayed for 20 seconds just prior to the Commit.
2. 2nd terminal: Start a second New-Order transaction for the same customer used by the 1st terminal. This transaction is forced to wait while the 1st terminal holds a lock on the CUSTOMER record requested by the 2nd terminal.
3. 1st terminal: The New-Order transaction is allowed to complete and Commit the transaction. With the CUSTOMER record released, the 2nd terminal New-Order transaction will complete.
4. Verify the order number from the 2nd terminal New-Order transaction is 1 greater than the order number from the 1st terminal. Verify D_NEXT_O_ID is 1 greater than the order number of the New-Order transaction from the 2nd terminal.

4.3.4 Isolation Test 4

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

The following steps were performed to verify the isolation of two New-Order transactions after one is rolled back:

1. 1st terminal: Start a New-Order transaction using the necessary inputs to cause a roll back (invalid item number). The transaction is delayed for 30 seconds just prior to the Roll back.
2. 2nd terminal: Start a second New-Order transaction for the same customer used by the 1st terminal. This transaction is forced to wait while the 1st terminal holds a lock on the CUSTOMER record requested by the 2nd terminal.
3. 1st terminal: The New-Order transaction is allowed to complete and the transaction is rolled back due to the invalid item number.

4. 2nd terminal: With the CUSTOMER record released, the 2nd terminal New-Order transaction will complete normally.
5. Verify the order number from the 2nd terminal New-Order transaction is equal to the next order number before either New-Order transaction was started. Verify D_NEXT_O_ID is 1 greater than D_NEXT_O_ID before either transaction was started and is 1 greater than the order number for the 2nd New-Order transaction.

4.3.5 Isolation Test 5

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

The following steps were performed to successfully conduct this test:

1. 1st terminal: A Delivery transaction is started. The Delivery transaction batch job is started with code added to the transaction to delay the transaction just prior to the Commit.
2. 2nd terminal: Start a Payment transaction for a customer that will have an order delivered in the transaction started in step 1. The Payment transaction is forced to wait while the Delivery transaction holds a lock on the CUSTOMER record.
3. 1st terminal: After the delay time period is expired for the Delivery transaction, the transaction is Committed.
4. 2nd terminal: With the CUSTOMER record released, the Payment transaction is now able to complete.
5. The record is viewed from the CUSTOMER table for the customer and C_BALANCE is verified to reflecting the changes from the Delivery and Payment transactions.

4.3.6 Isolation Test 6

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

The following steps were performed to successfully conduct this test:

1. 1st terminal: Start a Delivery transaction using the Delivery batch job with the delay coded in to halt the transaction just prior to the Roll back.
2. 2nd terminal: Start a Payment transaction for a customer that will have an order delivered in the transaction started in step 1. The Payment transaction is forced to wait while the Delivery transaction holds a lock on the CUSTOMER record.
3. 1st terminal: The Delivery transaction is rolled back after the time period expires.
4. 2nd terminal: With the CUSTOMER record released, the Payment transaction is now able to complete.
5. The record is viewed from the CUSTOMER table for the customer and C_BALANCE is verified to reflecting the change only from the Payment transaction.

4.3.7 Isolation Test 7

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

1. 1st terminal: Using interactive SQL, view and record prices for two items (x,y) selected at random.
2. 2nd terminal: A New-Order transaction is started that contains item x twice and y once. This transaction is stopped after reading the price of item x from the item file the first time.
3. 1st terminal: Using interactive SQL, an update transaction is started for items x and y, increasing their price by 10%. Case A, transaction 3 stalls.
4. 2nd terminal: The New-Order transaction is allowed to complete. It is verified that the prices for items x and y are the same throughout the entire transaction and that they match the results of step 1.
5. 1st terminal: After the New-Order transaction completes, the update transaction completes and is committed.
6. 1st terminal: Step 1 is repeated, noting that the prices of items x and y now match those set in step 3.

4.3.8 Isolation Test 8

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

1. Remove all rows for a randomly selected district and warehouse from the NEW-ORDER table.
2. Start a Delivery transaction T1 for a selected warehouse.
3. Stop T1 immediately after reading the NEW-ORDER table for the selected district. No qualifying rows should be found and none were.
4. Start a New-Order transaction T2 for the same warehouse and district.
5. Transaction T2 stalls. Continue transaction T1 by repeating the read of the NEW-ORDER table for the selected district.
6. Verify that there is still no qualifying rows found. There are none.
7. Complete and COMMIT transaction T1.
8. Transaction T2 should now complete and it does.

4.3.9 Isolation Test 9

This test demonstrates isolation for phantom protection between an Order-Status and a New-Order transaction.

1. Start an Order-Status transaction T1 for a selected customer.
2. Stop T1 immediately after reading the ORDER table for the selected customer. The most recent order for that customer is found.
3. Start a New-Order transaction T2 for the same customer.
4. Transaction T2 stalls. Continue transaction T1 by repeating the read of the ORDER table for the selected customer.
5. Verify that the order found is the same as in step 3.
6. Complete and COMMIT transaction T1.
7. Transaction T2 should now complete and it does.

4.3.10 Isolation Test 10

Verify that the data read by the Stock-Level transaction is not older than the most recently committed data prior to the time the transaction was initiated.

1. Run the Stock-Level transaction and note the number of qualifying items.
2. Display the list of Item_ids used by the first 20 d_next_o_ids.
3. Display the items in the 21st d_next_o_id.
4. Choose an Item_id in the 21st next-order and modiry its inventory level so that it would qualify for the Stock-Level transaction run at Step 1.
5. Rerun the Stock-Level transaction using the same threshold used in Step 1 - there should be no change in the result and there is none.
6. Change the inventory for an Item_id from the list generated in Step 2.
7. Rerun the Stock-Level transaction - it should have a result +1 higher than Step 1 and it does.

4.4 Durability Requirements

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

4.4.1 Permanent Unrecoverable Failure of any Single Durable Medium

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

Failure of Durable Medium containing recovery log data and Instananeous Interruption and Memory Failure.

This test was conducted on a fully scaled database. The following steps were performed successfully.

1. The current count of the total number of orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving SUM_1.
2. A test was started and allowed to run for twelve minutes.
3. One of the disks containing the Sybase transaction log data was powered off. Since this disk was mirrored, Sybase continued to process the transactions successfully.
4. The test continued for another 1 1/2 minutes.
5. The system was immediately shut down by switching the Emergency Power Off , thereby removing system power.
6. The disk from step 3 was powered back on.
7. The system was powered back on and rebooted.
8. Step 1 is performed returning the value for SUM_2. It was verified that SUM_2 was equal to SUM_1 plus the completed New_Order transactions recorded by the RTE and that no entries existed for rolled-back transactions.
9. Consistency condition 3 was verified.

Failure of Durable Medium containing TPC-C data base tables.

The following steps were successfully performed to pass the Durability test of failure of a disk unit with data base tables:

1. The contents of a disk containing a TPCC table was backed up by copying it to another disk.
2. The current count of the total number of orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving SUM_1.
3. A test was started and allowed to run for fifteen minutes.
4. The disk containing the TPCC table was powered off.
5. The run was aborted on the RTE.
6. The disk from step 5 was powered back on and the system was rebooted.
7. The failed disk was restored from the backup copy in step 2.
8. Sybase was restarted and its transaction log was used to roll forward the transactions that had completed but weren't written to disk before the failure.
9. Step 3 was performed returning SUM_2. It was verified that SUM_2 was equal to SUM_1 plus the completed New_Order transactions recorded by the RTE and that no entries existed for rolled-back transactions.
10. Consistency condition 3 was verified.

.

5. Clause 4: Scaling and Data Base Population Related Items

5.1 Cardinality of Tables

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

Tables 5-1 portray the TPC Benchmark™ C defined tables and the number of rows for each table as they were built initially.

Table 5-1 Initial Cardinality of Tables (RS/6000 F50)

Table Name	Number of Rows
Warehouse	660
District	7,000
Customer	21,000,000
History	21,000,000
Order	21,000,000
New Order	6,300,000
Order Line	210,000,000
Stock	70,000,000
Item	100,000

5.2 Distribution of Tables and Logs

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

The following table depicts the data base configuration of the system tested.

Table 5-2. RS/6000 F50 Data Distribution Benchmark Configuration

Controller	Disk	Contents	Capacity
scsi0	hdisk0	OS, paging	9 GB
	hdisk1	customer index	9 GB
	hdisk2	log	9 GB
	hdisk3	log	9 GB
scsi1	hdisk4	customer index	9 GB
	hdisk5	180 day space	9 GB
	hdisk6	log	9 GB
scsi2	hdisk7	customer index	9 GB
	hdisk8	log	9 GB
ssa0	hdisk9	whse, dist, item, new-order, Sybase master, tempdb	4 GB

Controller	Disk	Contents	Capacity
	hdisk10	whse, dist, item, new-order, Sybase master, tempdb	4 GB
	hdisk11	customer index	2 GB
	hdisk12	stock	2 GB
	hdisk13	stock	2 GB
	hdisk14	stock	2 GB
	hdisk15	stock	2 GB
	hdisk16	customer	2 GB
	hdisk17	customer	2 GB
	hdisk18	customer	2 GB
	hdisk19	customer	2 GB
	hdisk20	customer	2 GB
	hdisk21	customer	2 GB
	hdisk22	customer	2 GB
	hdisk23	customer	4 GB
	hdisk24	customer	4 GB
	hdisk25	customer	4 GB
	hdisk26	orderline, orders, & history	4 GB
	hdisk27	orderline, orders, & history	4 GB
	hdisk28	orderline, orders, & history	4 GB
	hdisk29	orderline, orders, & history	2 GB
	hdisk30	orderline, orders, & history	2 GB
	hdisk31	orderline, orders, & history	2 GB
	hdisk32	orderline, orders, & history	2 GB
	hdisk33	orderline, orders, & history	2 GB
	hdisk34	orderline, orders, & history	2 GB
	hdisk35	orderline, orders, & history	2 GB
	hdisk36	stock	2 GB
	hdisk37	stock	2 GB
	hdisk38	stock	2 GB
	hdisk39	stock	2 GB
	hdisk40	stock	2 GB
	hdisk41	stock	2 GB
	hdisk42	stock	2 GB
	hdisk43	stock	2 GB
	hdisk44	stock	2 GB
	hdisk45	stock	2 GB
	hdisk46	stock	2 GB
	hdisk47	stock	2 GB

Controller	Disk	Contents	Capacity
	hdisk48	stock	2 GB
	hdisk49	stock	2 GB
	hdisk50	stock	2 GB
	hdisk51	stock	4 GB
	hdisk52	stock	4 GB
	hdisk53	stock	4 GB
	hdisk54	stock	4 GB
	hdisk55	stock	4 GB
	hdisk56	stock	4 GB
	hdisk57	customer	2 GB
	hdisk58	customer	2 GB
	hdisk59	customer	2 GB
ssa1	hdisk60	orderline, orders, & history	2 GB
	hdisk61	orderline, orders, & history	2 GB
	hdisk62	orderline, orders, & history	2 GB
	hdisk63	orderline, orders, & history	2 GB
	hdisk64	orderline, orders, & history	2 GB
	hdisk65	orderline, orders, & history	2 GB
	hdisk66	orderline, orders, & history	2 GB
	hdisk67	orderline, orders, & history	4 GB
	hdisk68	orderline, orders, & history	4 GB
	hdisk69	orderline, orders, & history	4 GB
	hdisk70	customer	4 GB
	hdisk71	customer	4 GB
	hdisk72	customer	4 GB
	hdisk73	customer	2 GB
	hdisk74	customer	2 GB
	hdisk75	customer	2 GB
	hdisk76	customer	2 GB
	hdisk77	customer	2 GB
	hdisk78	customer	2 GB
	hdisk79	customer	2 GB
	hdisk80	stock	2 GB
	hdisk81	stock	2 GB
	hdisk82	stock	2 GB
	hdisk83	stock	2 GB
	hdisk84	stock	2 GB
	hdisk85	stock	2 GB
	hdisk86	stock	2 GB

Controller	Disk	Contents	Capacity
	hdisk87	stock	2 GB
	hdisk88	stock	2 GB
	hdisk89	stock	2 GB
	hdisk90	stock	2 GB
	hdisk91	stock	2 GB
	hdisk92	stock	2 GB
	hdisk93	stock	2 GB
	hdisk94	stock	2 GB
	hdisk95	stock	2 GB
	hdisk96	stock	2 GB
	hdisk97	stock	4 GB
	hdisk98	stock	4 GB
	hdisk99	stock	4 GB
	hdisk100	stock	4 GB
	hdisk101	stock	4 GB
	hdisk102	stock	4 GB
	hdisk103	stock	2 GB
	hdisk104	stock	2 GB
	hdisk105	stock	2 GB
	hdisk106	stock	2 GB
	hdisk107	stock	2 GB
	hdisk108	stock	2 GB
	hdisk110	stock	2 GB
	hdisk111	stock	2 GB
	hdisk112	stock	2 GB
	hdisk113	stock	2 GB
	hdisk114	stock	2 GB
	hdisk115	stock	2 GB
	hdisk116	stock	2 GB
	hdisk117	stock	2 GB

- The Order_line, Orders and History tables are striped across a set of twenty disks.
- The master database, tempdb and the Warehouse, District, Item and New_order tables are striped across a set of two disks.
- The customer index table is striped across a set of four disks.
- The Stock table is striped acorss a set of sixty-two disks.
- The Customer table is striped across a set of twenty-three disks.
- The eighty-two 2 GB disks are replaced with 4 GB disks in the pricing to satisfy the 180-day space requirements.

5.3 Data Base Model Implemented

A statement must be provided that describes the data base model implemented by the DBMS used.

The database manager used for this testing was Sybase Adaptive Server Enterprise 11.5 from Sybase Inc. Sybase Adaptive Server Enterprise 11.5 is a relational DBMS.

5.4 Partitions/Replications Mapping

The mapping of data base partitions/replications must be explictly described.

IBM did not implement horizontal or vertical partitioning for this TPC-C test.

5.5 180 day space calculations (RS/6000 Workgroup Server F50)

Table	Rows	Data	Index	5% Space	8H Space	TotalSpace
Warehouse	700	1248	10	63		1,321
District	7000	12484	54	627		13,165
Item	100000	9524	86	192		9,802
New-order	6300000	68854	828		14,000	83,682
History	21000000	1176080	0		175,587	1,351,667
Orders	21000000	567588	6866		85,765	660,219
Customer	21000000	14000000	1222098	304,442		15,526,540
Order-line	210000000	12727274	167466		1,925,162	14,819,902
Stock	70000000	23533332	130026	473,267		24,136,625
Totals		52,096,384	1,527,434	778,591	2,200,513	56,602,923

SegmentLogDev Cnt.	Seg.Size	Needed	Overhead	NotNeeded
wdino	1	117,760	109,050	1,090	7,620
history	2	1,433,600	1,365,183	13,652	54,765
order	11	692,224	666,821	6,668	18,735
customer	21	16,239,616	15,681,805	156,818	400,993
order_line	20	15,575,040	14,968,101	149,681	457,258
stock	42	25,116,672	24,377,991	243,780	494,901
Totals		59,174,912	57,168,952	571,690	1,434,271

Dynamic space	14,022,343	Sum of Data for Order, Order-Line and History (excluding free extents
Static space	40,951,756	Data + Index + 5% Space + Overhead - Dynamic space
Free space	2,766,543	Total Seg. Size - Dynamic Space - Static Space - NotNeeded
Daily growth	2,609,085	(Dynamic space/W * 62.5)* tpmC
Daily spread	(1,147,085)	Free space - 1.5 * Daily growth (zero if negative)
180 day (KB)	510,587,099	Static space + 180 (daily growth + daily spread)
180 day (GB)	486.93	Excludes OS, Paging and RDBMS Logs
Log per N-O txn	1.79	Numbe of 2K blocks per New-Order transaction
8 Hour Log (GB)	13.34	

Space Required	(GB)		8 Hr Log + Mir	Capacity	Disks
180 Day	486.93		(gb)	(gb)	
8 Hr Log + Mir	0.00	>>>	26.68	8.47	4.00
AIX/Paging	8.47				
Total:	495.40				

Space Available				TotDisks
Disk Type	FormatCap	Quantity	(GB)	
4.5 GB	4,296	108	453	108
9.1 GB	8672	5	42	5
Total:			495.44	117

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

6.1 Response Times

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

Tables 6-1 list the response times and the ninetieth percentiles for each of the transaction types for the measured system.

6.2 Keying and Think Times

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

Table 6-1 list the TPC-C keying and think times for the measured system.

Table 6-1. RS/6000 Workgroup Server F50 Response, Think and Keying Times

Response Times	New Order	Payment	Order Status	Delivery (int./def.)	Stock Level	Menus
90 %	1.54	1.45	1.52	0.61/2.00	2.51	0.01
Average	0.66	0.58	0.7	0.27/1.14	1.2	0.01
Maximum	13.32	14.07	9.85	9.34/10.00	11.43	0.29
			Think Times			
Minimum	0.01	0.01	0.01	0.01	0.01	N/A
Average	12.13	12.12	10.12	5.12	5.1	N/A
Maximum	121	121	101	51	45.71	N/A
			Keying Times			
Minimum	18	3	2	2	2	N/A
Average	18	3	2	2	2	N/A
Maximum	18.11	3.1	2.11	2.1	2.08	N/A

6.3 Response Time Frequency Distribution

Response time frequency distribution curves must be reported for each transaction type.

Figure 6-3-1. RS/6000 F50 New-Order Response Time Distribution

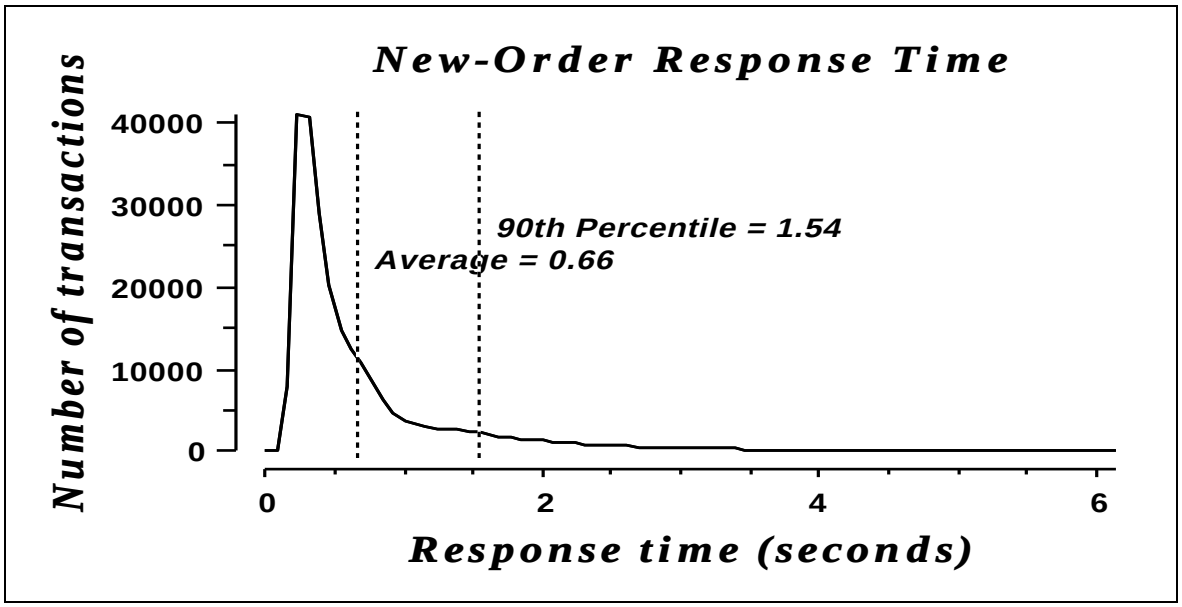


Figure 6-3-2. RS/6000 F50 Payment Response Time Distribution

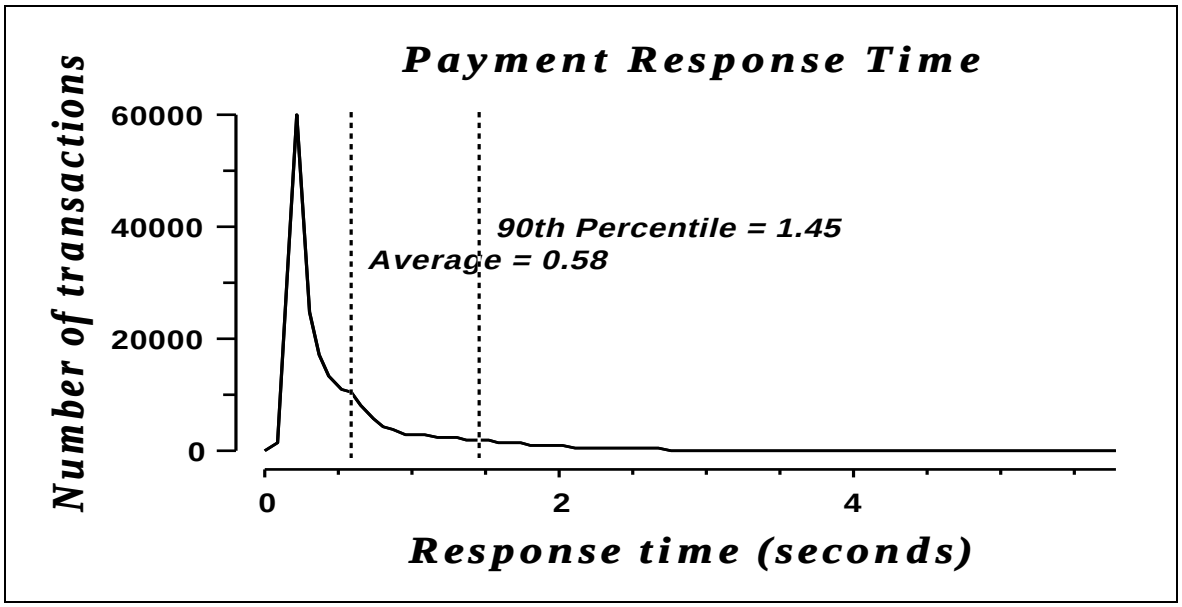


Figure 6-3-3. RS/6000 F50 Order-Status Response Time Distribution

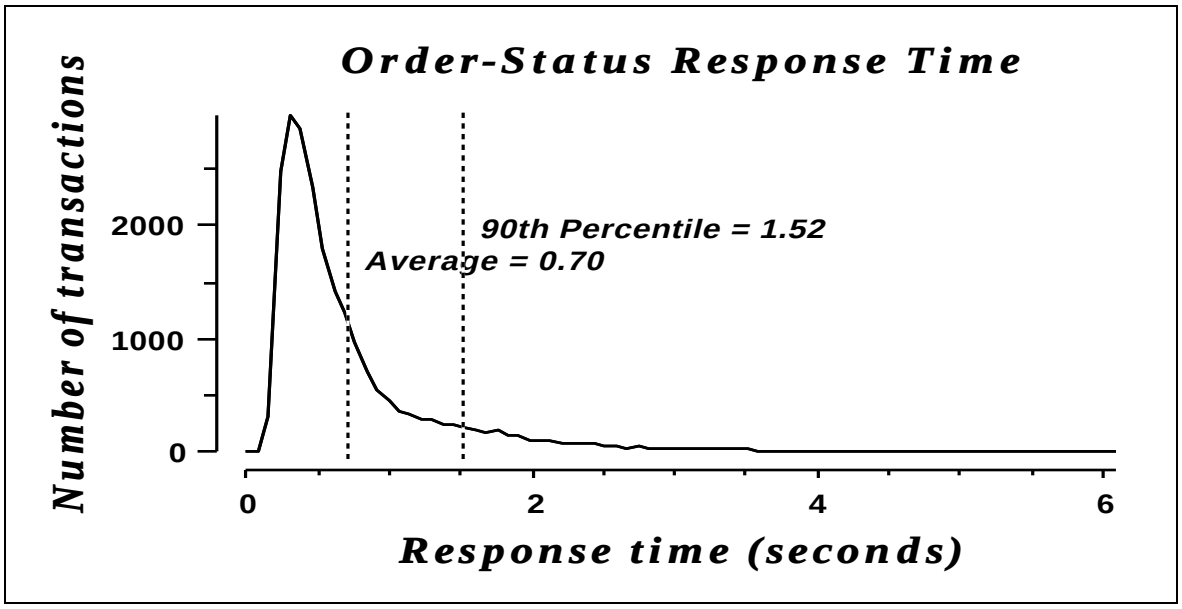


Figure 6-3-4. RS/6000 F50 Delivery (Interactive) Response Time Distribution

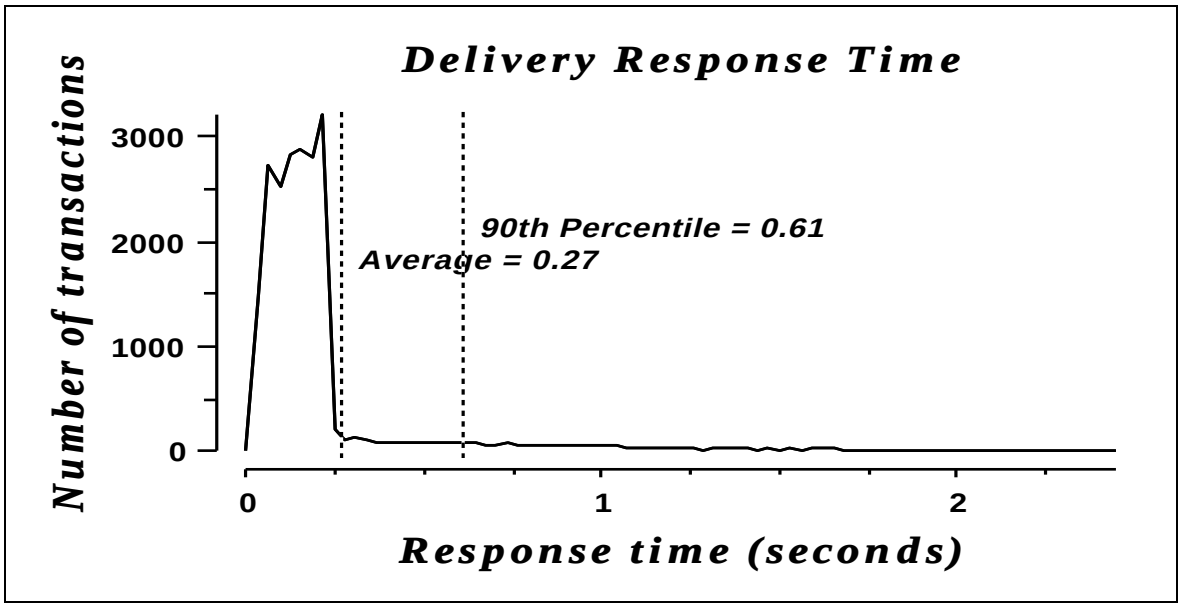


Figure 6-3-5. RS/6000 F50 Delivery (Deferred) Response Time Distribution

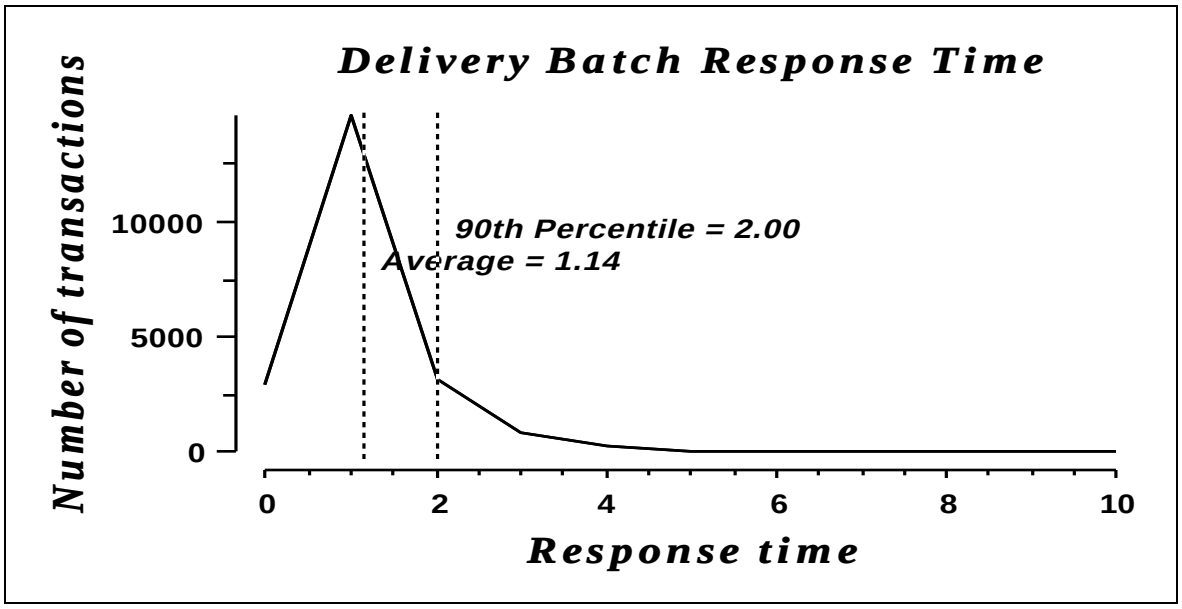
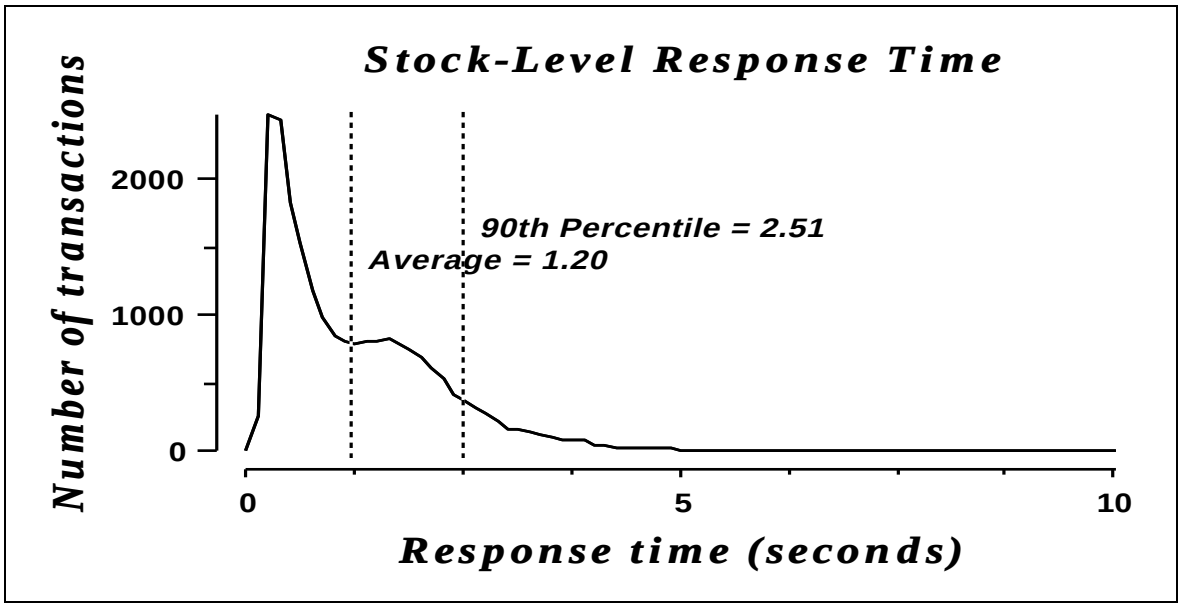


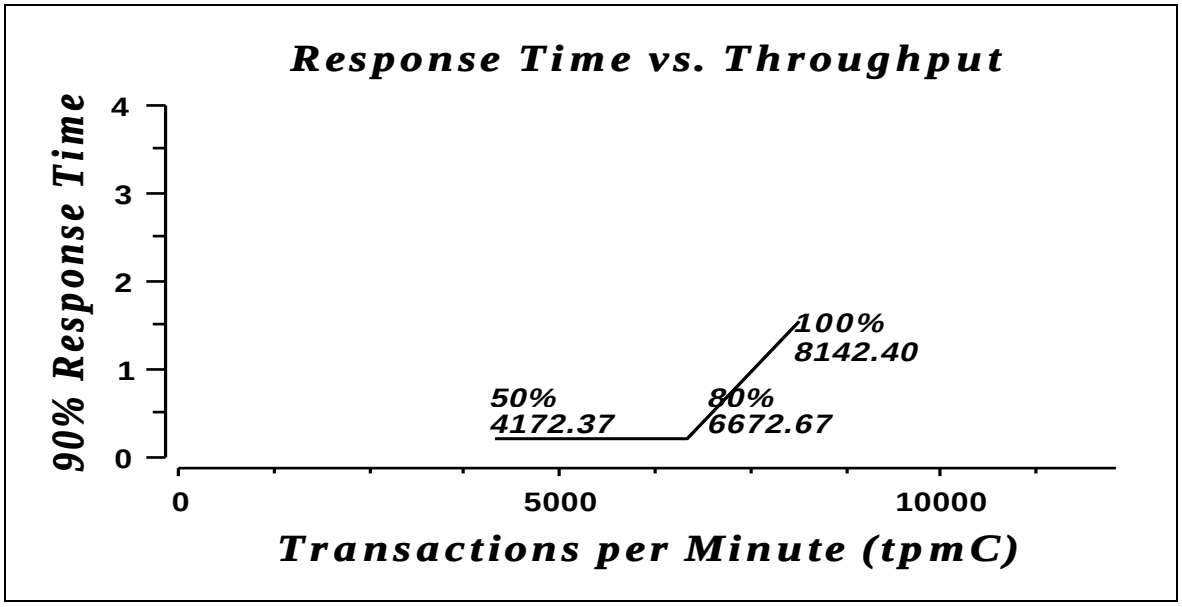
Figure 6-3-6. RS/6000 F50 Stock Level Respone Time Distribution



6.4 Performance Curve for Response Time versus Throughput

The performance curve for response times versus throughput must be reported for the New-Order transaction.

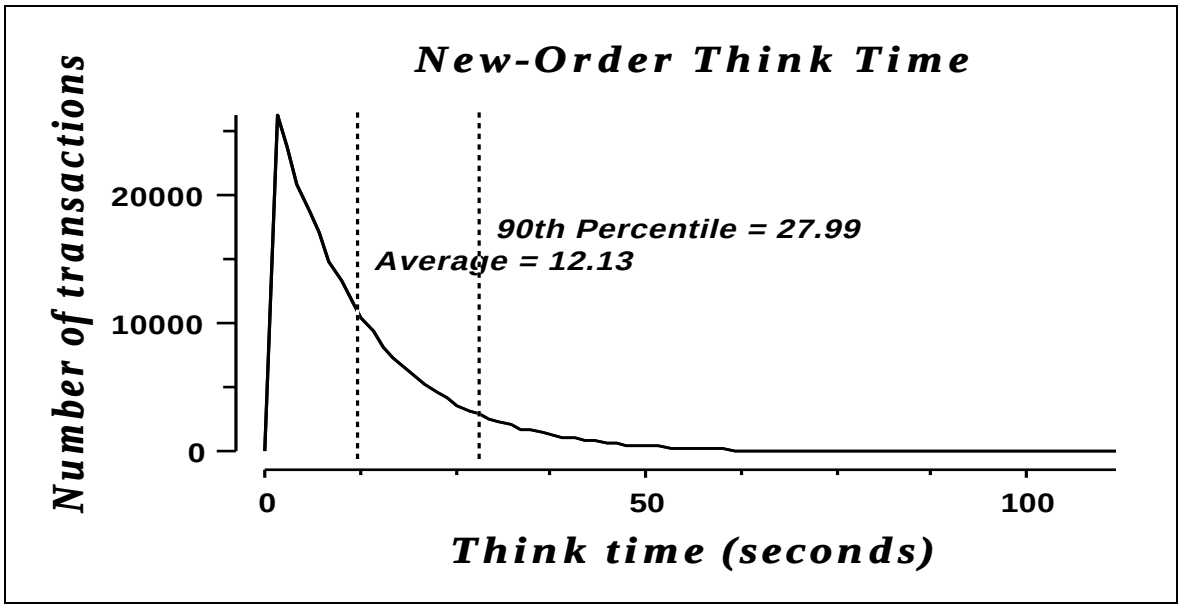
Figure 6-4-1. RS/6000 F50 New-Order Response Time vs. Throughput



6.5 Think Time Frequency Distribution

A graph of the think time frequency distribution must be reported for the New-Order transaction.

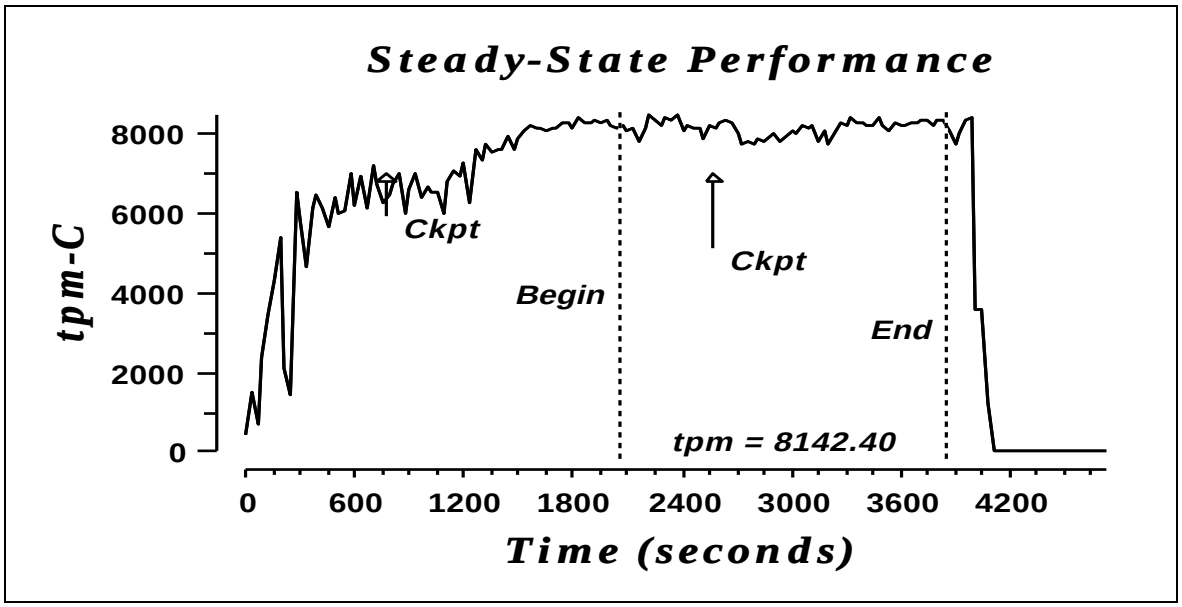
Figure 6-5-1. RS/6000 F50 New-Order Think Time Distribution



6.6 Throughput versus Elapsed Time

A graph of throughput versus elapsed time must be reported for the New-Order transaction.

Figure 6-6-1. New-Order Throughput vs. Elasped Time



6.7 Steady State Determination

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

All the emulated users were allowed to logon and do transactions. The timestamping interval was set to start after several minutes of rampup. Refer to the Numerical Quantities Summary pages for the rampup time. Figure 6.7.1 New-Order throughput versus Elapsed Time graph show that the system was in steady state at the beginning of the Measurement Interval.

6.8 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.

6.8.1 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.
- The TPC-C ***main*** program connects to Tuxedo System/T before starting any transaction operation.
- The TPC-C ***main*** program displays the TPC-C transaction menu on the user terminal.
- The TPC-C user chooses the transaction type and proceeds to fill the screen fields required for that transaction.
- The TPC-C ***main*** program accepts all values entered by the user and transmits those values to one of the TPC-C back end programs. The transmission is performed through a Tuxedo System/T function call. Each TPC-C back end program has a "service-name". This service-name is specified whenever the TPC-C main program requests a Tuxedo System/T service. Tuxedo System/T routes that message 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 terminal is 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.
- The TPC-C ***main*** program takes the message content and writes the transaction output on the user terminal.

6.8.2 Database Transaction

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

Using Sybase Open Client DB-Library calls, the TPC-C back-end program interacts with Sybase Adaptive Server Enterprise 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 Adaptive Server Enterprise proceeds to update the database as follows:

When Sybase Adaptive Server Enterprise 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 addtional data pages, Sybase Adaptive Server Enterprise 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.

6.8.3 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 measurement interval.

6.9 Reproducibility

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

A repeatability measurement was taken for the same length of time as the measured run. The repeatability measurement was 8,090.03 tpmC.

6.10 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 Availability

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

IBM used an internally developed RTE for these tests. Appendix E contains the scripts used in the testing.

7.2 Functionality and Performance of Emulated Components

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

In the benchmark configuration the Remote Terminal Emulator (RTE) communicates with the client system over Ethernet. One RS/6000 Model 39H emulates a network of 1,100 RS/6000 Model 43P-140 workstations. The communications mechanism used in the benchmarked and priced configurations are the same. In the benchmark configuration a separate Ethernet LAN was used to connect two driver systems to a client system. In other words, there was a separate LAN segment every two drivers to a client. Each LAN segment in the priced configuration is used to connect 734 workstations.

7.4 Network Bandwidth

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

The Ethernet used in the LAN complies with the IEEE 802.3 standard and has a bandwidth of 10 Megabits per second. Each LAN segment in the RS/6000 Workgroup Server F50 configuration connected 734 workstations.

7.4 Operator Intervention

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

No operator intervention is required to sustain the reported throughput during the eight hour period.

8. Clause 7: Pricing Related Items

8.1 Hardware and Programs Used

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

The detailed list of all hardware and programs for the priced configuration is listed in the pricing sheets (please refer to Section 8.2 for details) for each system reported. The prices for all products and features that are provided by Dickens Data Systems are available the same day as product or feature availability.

Prices were quoted by Sybase, Inc. based on machine-type category which Sybase has classified as a Midrange System.

Pricing for Tuxedo EPT was quoted by BEA Systems, Inc. Service for five years is 15% annually of list price.

Pricing for CentreCom Workgroup 24 port Ethernet Hubs was quoted by Allied Telesyn International. These have a lifetime return-to-factory warranty.

8.2 Five Year Cost of System Configuration

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

The price sheets for the RS/6000 are contained on the following page. The basis for the discounts used are:

- Extended Maintenance Option (EMO):
  - ►This is a discount for prepayment of maintenance costs for the system unit, disk, and the terminals. A discount of seventeen percent is available for this configuration based on payment for five years maintenance at time of purchase.
- Mid-Range Service Option (MRSO):
  - ►This discount is available for customers when agreement is reached for the customer to perform specified service duties (consult marketing representative for details). This discount is applied to the balance after the Extended Maintenance Option Discount is applied. For the TPC Benchmark™C configurations the MRSO discount is seventeen percent for the system, disk and terminals.
- Dickens Data Systems provides complete hardware and software solutions to end-users and offers customers dollar volume discounts.

8.3 Availability Dates

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

All products are generally available today except the following:

Product	Availability Date
Sybase Adaptive Server Enterprise 11.5	September 30, 1997

8.4 Statement of tpmC and Price/Performance

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

.

System	tmpC	5-year System Cost	$/tmpC	Availability Date
RS/6000 Workgroup Server F50	8,142.4	$510,643	$62.71	September 30, 1997

RS/6000 Workgroup Server F50 Five Year System Price Configuration

Description	Part Number	Third Party Brand	Third Party Pricing	Unit Price	Qty	Extended Price	5 yr. Maint Price
Server Hardware							
RS/6000 Server Model F50	7025-F50		1	19,900	1	19,900	11,952
1 604e, 128mb Memory, 4,5GB Disk,							
2 Intg SCSI2 F/W Adptr, CDROM							
604e 2way Proc Card Select, 2-256kb L2	4303		1	4,000	1	4,000	3,120
604e 2way Proc Card, 2-256kb L2	4309		1	8,000	1	8,000	6,240
256MB DIMM Memory Select	4106		1	3,200	1	3,200	0
256MB DIMM Memory Modules	4110		1	6,400	11	70,400	0
Memory Expansion Feature 2nd Card	4093		1	1,038	1	1,038	0
9.1GB SCSI-2 F/W Hot Swap Disk	2911		1	3,850	8	30,800	0
9.1GB SCSI-2 F/W Hot Swap Disk Sel	3019		1	1,800	1	1,800	0
SCSI Hot Swap 6-Pack Kits	6519		1	750	1	750	0
16-bit Integrated SCSI Adapter Cable	2444		1	66	1	66	0
PCI Ethernet Adapter	2985		1	195	2	390	0
PCI SCSI-2 F/W Adapter	6208		1	360	1	360	0
PCI SSA 4port RAID Adapter	6215		1	3,000	3	9,000	0
System Rack Model R00	7015-R00		1	3,110	2	6,220	2,976
SSA Disk Subsystem w/ 4 4.5GB Disk	7133-020		1	19,350	7	135,450	67,200
4.5 GB Disk Drive Modules	3401		1	2,100	80	168,000	0
SSA Cables	5010		1	40	14	560	0
				Subtotal		459,934	91,488
Server Software							
AIX 4.2.1 F50 + Support	5765-C34		1	240	1	240	0
Adaptive Server Enterprise 11.5 incl Client		Sybase	2	35,995	1	35,995	28,000
				Subtotal		36,235	28,000
Client Hardware							
RS/6000 Model 43P-140, 233 MHz	7043-140		1	8,000	3	24,000	10,080
2.1gb Disk, Intg SCSI-2 FW, Intg Enet							
128MB DIMM Memory	4102		1	1,280	3	3,840	0
128MB DIMM Memory Expansion	4115		1	2,560	12	30,720	0
Async Terminal/Printer Cable	2934		1	45	4	180	0
Ethernet Adapter, PCI	2985		1	195	9	1,755	0
				Subtotal		60,495	10,080
Client Software							
AIX 4.1.5 Unlimited Users	5756-C34		1	3,660	3	10,980	0
Tuxedo EPT Ver 6.2, 43P Client		BEA	3	3,000	3	9,000	6,750
				Subtotal		19,980	6,750
User Connectivity							
Ethernet Hub (24port)	AT3024TR	CentreCOM	4	215.95	275	59,386	0
Ethernet Hub 10% Spares	AT3024TR	CentreCOM	4	215.95	28	6,047	0
IBM ASCII Terminal, Keyboard	3153-BG3		1	577	4	2,308	2,640
					Subtotal	67,741	2,640
					Discounts	(240,281)	(32,419)
					Total	**404,104**	**106,539**

Notes:
Pricing Sources:
1=Dickens Data, 2=Sybase, 3=BEA Systems, 4=Data Comm Warehouse
Audited by: Francois Raab, Information Paradigm

Five-Year Cost of Ownership: 510,643
tpmC Rating: 8,142.40
$/tpmC: 62.71

8. Clause 9: Audit Related Items

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

The auditor's attestation letter is included at the end of the appendix section of this report.

Appendix A: TPC-C Application Source

A.1 Client/Terminal Handler code

```
/*************************************************************************/
/* client.cpp                                    03/31/97 */
/*************************************************************************/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

#include "screen.h"
#include "tuxedo.h"

InOut io(0, 1);
Tuxedo tux;

User_data user_data;

int main () {
   Menu  menu;
   Login log(&user_data);
   log.handle();
   menu.present();

   Payment pay(&user_data);
   Delivery del(&user_data);
   OrderStatus os(&user_data);
   StockLevel sl(&user_data);
   NewOrder no(&user_data);

   while (1) {
                int key = Field::get_key();
                switch (key) {
                   case '1': case 'N': case 'n': no.handle();  break;
                   case '2': case 'P': case 'p': pay.handle(); break;
                   case '3': case 'O': case 'o': os.handle();  break;
                   case '4': case 'D': case 'd': del.handle(); break;
                   case '5': case 'S': case 's': sl.handle();  break;
                   case '\030': position(1, 1); io.flush(); break;
                   case '9': case 'Q': case 'q': case 'E': case 'e':
                                position(1, 24); io.flush(); return 0;
                   default: io.write("\a", 1); break;
                }
   }
}

/*************************************************************************/
/* tpcc.h                                        03/31/97 */
/*************************************************************************/
#if !defined(TPCC_H_INCLUDED)
#define TPCC_H_INCLUDED
/*************************************************************************/
/*                                                       */
/*     File: tpcc.h                                      */
/*                                                       */
/*     program description:                              */
/*                                                       */
/*     This module contains global variables and data definitions     */
/*     for the tpcc application.                                      */
/*                                                       */
/*************************************************************************/

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

#define TPCCH

/*-------------------------------------------------------*/
/*            Global numbers, constants,...              */
/*-------------------------------------------------------*/

#if 0
short    ware_num;
short    dist_num;
#endif

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

#define FORM_DATE                   1
#define FORM_DATETIME               2

#define MAX_ITEMS 15

/*-------------------------------------------------------*/
/*            transaction structures                     */
/*-------------------------------------------------------*/

struct neword_struct {
   DBSMALLINT               W_ID;
   DBTINYINT   D_ID;
   DBINT       C_ID;
   char        C_LAST[17];
   char        C_CREDIT[3];
   DBREAL      C_DISCOUNT;
   DBTINYINT   O_OL_CNT;
   DBINT       O_ID;
   char        O_ENTRY_DATE[20];
   char        statusline[25];
   DBFLT8      total_amount;
   DBTINYINT                all_local;
   short       status;
   DBREAL      W_TAX;
   DBREAL      D_TAX;
   struct itemstruct {
               DBSMALLINT   OL_SUPPLY_W_ID;
               DBINT        OL_I_ID;
               char         I_NAME[25];
               DBTINYINT    OL_QUANTITY;
               DBSMALLINT   QUANTITY;
               char         brand_generic[2];
               DBFLT8       I_PRICE;
               DBFLT8       OL_AMOUNT;
   } item[15];
};

struct payment_struct {
   DBSMALLINT      W_ID;
   DBTINYINT       D_ID;
   DBINT           C_ID;
   DBTINYINT       C_D_ID;
   DBSMALLINT      C_W_ID;
   short           status;
   DBFLT8          H_AMOUNT;
   char    H_DATE[20];
   char    W_STREET_1[21];
   char    W_STREET_2[21];
   char    W_CITY[21];
   char    W_STATE[3];
   char    W_ZIP[10];
   char    D_STREET_1[21];
   char    D_STREET_2[21];
   char    D_CITY[21];
   char    D_STATE[3];
   char    D_ZIP[10];
   char    C_FIRST[17];
   char    C_MIDDLE[3];
   char    C_LAST[17];
   char    C_STREET_1[21];
   char    C_STREET_2[21];
   char    C_CITY[21];
   char    C_STATE[3];
   char    C_ZIP[10];
   char    C_PHONE[17];
   char    C_SINCE[20];
   char    C_CREDIT[3];
   DBFLT8  C_CREDIT_LIM;
   DBREAL  C_DISCOUNT;
   DBFLT8  C_BALANCE;
   char    C_DATA[201];
};

struct ordstat_struct {
   DBSMALLINT     W_ID;
   DBTINYINT      D_ID;
   DBINT          C_ID;
   char    C_FIRST[17];
   char    C_MIDDLE[3];
   char    C_LAST[17];
   DBFLT8  C_BALANCE;
   DBINT   O_ID;
   char    O_ENTRY_DATE[20];
   DBSMALLINT     O_CARRIER_ID;
   short   ol_cnt;
   short           status;
   struct oitemstruct {
               DBSMALLINT     OL_SUPPLY_W_ID;
               DBINT          OL_I_ID;
               DBTINYINT      OL_QUANTITY;
               DBFLT8         OL_AMOUNT;
               char           OL_DELIVERY_DATE[20];
   } item[15];
};

struct delivery_struct {
   DBSMALLINT     W_ID;
   DBSMALLINT     O_CARRIER_ID;
   int            queued_time;
   short          status;
   char    exec_status[50];
};

struct stocklev_struct {
   DBSMALLINT     W_ID;
   DBTINYINT      D_ID;
   DBSMALLINT     threshold;
```

```
    DBINT         low_stock;
    short         status;
};

typedef struct ordstat_struct OrderStatus_data;
typedef struct neword_struct NewOrder_data;
typedef struct stocklev_struct StockLevel_data;
typedef struct delivery_struct Delivery_data;
typedef struct payment_struct Payment_data;

/*****************************************************************************
 Compatibility for older .sqc files
 *****************************************************************************/
#define s_C_BALANCE C_BALANCE
#define s_C_CITY C_CITY
#define s_C_CREDIT C_CREDIT
#define s_C_CREDIT_LIM C_CREDIT_LIM
#define s_C_DATA C_DATA
#define s_C_DISCOUNT C_DISCOUNT
#define s_C_D_ID C_D_ID
#define s_C_FIRST C_FIRST
#define s_C_ID C_ID
#define s_C_LAST C_LAST
#define s_C_MIDDLE C_MIDDLE
#define s_C_PHONE C_PHONE
#define s_C_SINCE C_SINCE
#define s_C_STATE C_STATE
#define s_C_STREET_1 C_STREET_1
#define s_C_STREET_2 C_STREET_2
#define s_C_W_ID C_W_ID
#define s_C_ZIP C_ZIP
#define s_D_CITY D_CITY
#define s_D_ID D_ID
#define s_D_STATE D_STATE
#define s_D_STREET_1 D_STREET_1
#define s_D_STREET_2 D_STREET_2
#define s_D_TAX D_TAX
#define s_D_ZIP D_ZIP
#define s_H_AMOUNT H_AMOUNT
#define s_H_DATE H_DATE
#define s_I_NAME I_NAME
#define s_I_PRICE I_PRICE
#define s_OL_AMOUNT OL_AMOUNT
#define s_OL_DELIVERY_D OL_DELIVERY_DATE
#define s_OL_I_ID OL_I_ID
#define s_OL_QUANTITY OL_QUANTITY
#define s_OL_SUPPLY_W_ID OL_SUPPLY_W_ID
#define s_O_CARRIER_ID O_CARRIER_ID
#define s_O_ENTRY_D O_ENTRY_DATE
#define s_O_ID O_ID
#define s_O_OL_CNT O_OL_CNT
#define s_S_QUANTITY S_QUANTITY
#define S_QUANTITY QUANTITY
#define s_W_CITY W_CITY
#define s_W_ID W_ID
#define s_W_STATE W_STATE
#define s_W_STREET_1 W_STREET_1
#define s_W_STREET_2 W_STREET_2
#define s_W_TAX W_TAX
#define s_W_ZIP W_ZIP
#define s_all_local all_local
#define s_brand_generic brand_generic
#define s_exec_status exec_status
#define s_low_stock low_stock
#define s_ol_cnt ol_cnt
#define s_queued_time queued_time
#define s_status_line status_line
#define s_threshold threshold
#define s_total_amount total_amount
#define s_transtatus status

#if 0
#define NEWORDER_SERVICE    "NEWORD"
#define PAYMENT_SERVICE     "PAYMENT"
#define DELIVERY_SERVICE    "DELIVERY"
#define STOCKLEVEL_SERVICE  "STOCKLEV"
#define ORDERSTATUS_SERVICE "ORDSTAT"
#else
#define NEWORDER_SERVICE    "neword_sql"
#define PAYMENT_SERVICE     "payment_sql"
#define DELIVERY_SERVICE    "delivery_sql"
#define STOCKLEVEL_SERVICE  "stocklev_sql"
#define ORDERSTATUS_SERVICE "ordstat_sql"
#endif

#endif /* TPCC_H_INCLUDED */

/*****************************************************************************/
/* screen.cpp                                    03/31/97 */
/*****************************************************************************/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

#include "screen.h"
#include "tuxedo.h"
#include "format.h"

#define USE_INSULTS

extern InOut io;
extern Tuxedo tux;
extern char const * const blanks;
extern char const * const underscores;
extern char const * const backspaces;

int position(int x, int y, char *buf);
int position(int x, int y);
int clear_eos();
int clear_eos(char *buf);
int string_empty(char const *text);
int pos_zero(int const *val);
int pos_nonzeros(int const **val);

/*****************************************************************************
Screen
*****************************************************************************/
int Screen::reset() {
    has_data=0;
    pos=0;
    memset(dataptr, 0, data_len);
    for (int i = 0; fields[i] != NULL; i++) {
                fields[i]->reset();
    }
    return 0;
};
int Screen::present_empty_fields() {
    if (empty_fields)
                io.write(empty_fields, empty_fields_len);
    return 0;
}
int Screen::present() {
    io.write(screen, screen_len);
    io.write(session_data, session_data_len);
    if (has_data) {
                for (int i = 0; fields[i] != NULL; i++) {
                    fields[i]->display_field(1);
                }
    } else {
                present_empty_fields();
    }
    return 0;
};
int Screen::user_input() {
    int key;
    has_data = 1;
    fields[pos]->start_position();
    io.flush();
    key = fields[pos]->get_field(0);
    do {
                switch (key) {
                case Field::NEXT_FIELD:
                    if (fields[++pos] == NULL) {
                                pos = 0;
                    }
                    break;
                case Field::PREV_FIELD:
                    if (--pos < 0) {
                                while (fields[++pos] != NULL);
                                pos--;
                    }
                    break;
                case Field::REDISPLAY:
                    present();
                    break;
                case Field::ABORT:
                    io.write(end_str);
                    return 0;
                case Field::ENTER:
                    if (validate()) {
                                return 1;
                    }
                    break;
                }
                key = fields[pos]->get_field();
    } while (1);
    return 0;
}
Screen::~Screen() {
    if (fields != NULL) {
                for (int pos = 0; fields[pos] != NULL; pos++) {
                    delete fields[pos];
                }
                delete [] fields;
    }
    fields=NULL;
}

int Screen::display_status(int status) {
    position(status_x, status_y);
    io.write("Execution Status: ");
    if (status == TRAN_OK) {
                io.write("Transaction Committed");
    } else if (status == INVALID_ITEM) {
```

```
                io.write("Item number is not valid");
    } else {
                io.write("Rollback -- ");
                char buf[6];
                format_int(buf, 5, status);
                io.write(buf, 5);
    }
    return 0;
}

int Screen::handle() {
    io.debug("%s - reset\n", tran_type);
    reset();

    io.debug("%s - present\n", tran_type);
    io.hold();
    present();
    io.write(TRIGGER);
    io.debug("%s - user_input\n", tran_type);
    if (!user_input()) {
                io.write(end_str);
                io.write(TRIGGER);
                return -1;
    }
    io.flush();
    io.hold();
    io.debug("%s - process\n", tran_type);
    if (process()) {
                io.write(end_str);
                io.write(TRIGGER);
                return -1;
    }
    io.debug("%s - respond\n", tran_type);
    respond();
    io.write(end_str);
    io.write(TRIGGER);
    io.flush();
    return 0;
}

/*****************************************************************************
NewOrder
*****************************************************************************/
int NewOrder::reset() {
    Screen::reset();
    pos=start_field;
    memset(dataptr, 0, sizeof(*data));
    return 0;
};
NewOrder::NewOrder(User_data *ud) : Screen(ud) {
    tran_type = NEWORDER_SERVICE;
    dataptr = data = new NewOrder_data;
    data_len = sizeof(NewOrder_data);
    user_data = ud;

    sprintf(static_session_data, "%s%4d", POS(12,4), user_data->warehouse);

    status_x = 1;
    status_y = 24;

    screen          = static_screen;
    empty_fields    = static_empty_fields;
    session_data    = static_session_data;
    screen_len      = static_screen_len;
    empty_fields_len = static_empty_fields_len;
    session_data_len = static_session_data_len;

    int pos = 0;
    fields = new Field *[2+MAX_ITEMS*3+1];
    for (int i = 0; i < MAX_ITEMS; i++) {
                fields[pos++] = genfield( 3, 9+i, 4, &data->item[i].OL_SUPPLY_W_ID);
                fields[pos++] = genfield(10, 9+i, 6, &data->item[i].OL_I_ID);
                fields[pos++] = genfield(45, 9+i, 2, &data->item[i].OL_QUANTITY);
#if defined(USE_SMART_FIELDS)
                if (i > 0) {
                    int **tmp = new int *[4];
                    tmp[0] = &fields[pos-6]->pos;
                    tmp[1] = &fields[pos-5]->pos;
                    tmp[2] = &fields[pos-4]->pos;
                    tmp[3] = NULL;
                    fields[pos-3]->ok_func = (int(*)(void*))pos_nonzeros;
                    fields[pos-3]->ok_data = tmp;
                    fields[pos-2]->ok_func = (int(*)(void*))pos_nonzeros;
                    fields[pos-2]->ok_data = tmp;
                    fields[pos-1]->ok_func = (int(*)(void*))pos_nonzeros;
                    fields[pos-1]->ok_data = tmp;
                }
#endif
    }
    start_field = pos;
    fields[pos++] = genfield(29, 4, 4, &data->D_ID); /* District */
    fields[pos++] = genfield(12, 5, 4, &data->C_ID); /* Customer */
    fields[pos++] = NULL;
    reset();
};

int NewOrder::validate() {
    if (!fields[start_field]->pos) {
                pos=start_field;
                message("District ID is a required field");
                return 0;
    }
    if (!fields[start_field+1]->pos) {
                pos=start_field+1;
                message("Customer ID is a required field");
                return 0;
    }

    int last=-1;
    data->all_local = 1;
    data->W_ID = user_data->warehouse;
    for (int i = 0; i < MAX_ITEMS*3; i+=3) {
                if (fields[i]->pos || fields[i+1]->pos || fields[i+2]->pos){
                    if (last>=0) {
                                pos=last;
                                message("Warehouse ID is a required field");
                                return 0;
                    }
                    if (!fields[i]->pos) {
                                pos=i;
#if defined(USE_INSULTS)
                                message("Yeah, I think this is a bogus field too.");
#else
                                message("Warehouse ID is a required field");
#endif
                                return 0;
                    }
                    if (!fields[i+1]->pos) {
                                pos=i+1;
#if defined(USE_INSULTS)
                                message("Umm, WHAT did you want?");
#else
                                message("Item ID is a required field");
#endif
                                return 0;
                    }
                    if (data->item[i/3].OL_QUANTITY <= 0) {
                                pos=i+2;
#if defined(USE_INSULTS)
                                message("So something plus nothing is...");
#else
                                message("Please enter a quantity greater than 0");
#endif
                                return 0;
                    }
                    if (data->item[i/3].OL_SUPPLY_W_ID != data->W_ID) {
                                data->all_local=0;
                    }
                } else if (last < 0) {
                    last = i;
                }
    }
    data->O_OL_CNT = (last < 0)?MAX_ITEMS:last/3;
    if (data->O_OL_CNT <= 0) {
                pos=0;
#if defined(USE_INSULTS)
                message("It's kind of pointless without ordering something isn't it?");
#else
                message("Please enter an item to order");
#endif
                return 0;

    }

    return 1;
}

int NewOrder::respond() {
    int i;
    double amount, total_amount, cost;
    char buf[9];
    position( 1, 9); clear_eos();
    position(25, 5); io.write(data->C_LAST);
    position(52, 5); io.write(data->C_CREDIT);
    position(15, 6); format_int(buf, 9, data->O_ID); io.write(buf, 8);

    display_status(data->status);
    if (data->status != TRAN_OK) {
                return -1;
    }

    position(25, 5); io.write( data->C_LAST);
    position(52, 5); io.write( data->C_CREDIT);
    position(15, 6); format_int( buf, 9, data->O_ID); io.write(buf, 8);
    position(48, 6); format_int( buf, 3, data->O_OL_CNT); io.write(buf, 2);
    position(61, 4); format_date(buf, 20, data->O_ENTRY_DATE); io.write(buf, 19);
    position(64, 5); format_float(buf, 6, 2, data->C_DISCOUNT * 100); io.write(buf, 5);
    position(59, 6); format_float(buf, 6, 2, data->W_TAX*100); io.write(buf, 5);
    position(74, 6); format_float(buf, 6, 2, data->D_TAX*100); io.write(buf, 5);
    total_amount = 0;
    for (i=0; i < data->O_OL_CNT; i++) {
                position( 3, 9+i); format_int(buf, 5, data->item[i].OL_SUPPLY_W_ID); io.write( buf, 4 );
                position(10, 9+i); format_int(buf, 7, data->item[i].OL_I_ID); io.write( buf, 6 );
                position(19, 9+i); io.write( data->item[i].I_NAME);
                position(45, 9+i); format_int(buf, 3, data->item[i].OL_QUANTITY); io.write(buf, 2);
                position(51, 9+i); format_int(buf, 4, data->item[i].S_QUANTITY); io.write(buf, 3);
                position(58, 9+i); io.write(&data->item[i].brand_generic, 1);
                position(62, 9+i); format_money(buf, 8, data->item[i].I_PRICE); io.write(buf, 7);
```

```
                position(71, 9+i); format_money(buf, 10, data->item[i].OL_AMOUNT); io.write(buf, 9);
  }
  /* Clear the screen of any empty input fields */
  position(63, 24); io.write( "Total:");
  position(70, 24); format_money( buf, 10, data->total_amount ); io.write( buf, 9 );

  return 0;
}

/*****************************************************************************
Payment
*****************************************************************************/
Payment::Payment(User_data *ud)  : Screen(ud) {
  tran_type = PAYMENT_SERVICE;
  dataptr = data = new Payment_data;
  data_len = sizeof(Payment_data);

  sprintf(static_session_data, "%s%4d", POS(12,6), user_data->warehouse);

  int pos = 0;
  screen          = static_screen;
  empty_fields    = static_empty_fields;
  session_data    = static_session_data;
  screen_len      = static_screen_len;
  empty_fields_len = static_empty_fields_len;
  session_data_len = static_session_data_len;

  fields = new Field *[7];
  fields[pos++] = genfield( 52,  6,  2, &data->D_ID); /* District */
  fields[pos++] = genfield( 11, 11,  4, &data->C_ID); /* Customer # */
  fields[pos++] = genfield( 29, 12, 16, data->C_LAST); /* Name */
  fields[pos++] = genfield( 33, 11,  4, &data->C_W_ID); /* Cust-Warehouse */
  fields[pos++] = genfield( 54, 11,  2, &data->C_D_ID); /* Cust-District */
  fields[pos++] = genfield( 23, 17,  8, &data->H_AMOUNT); /* Amount Paid */
  fields[pos++] = NULL;
#if defined(USE_SMART_FIELDS)
  fields[1]->ok_func = (int(*)(void*))pos_zero;
  fields[1]->ok_data = &fields[2]->pos;
  fields[2]->ok_func = (int(*)(void*))pos_zero;
  fields[2]->ok_data = &fields[1]->pos;
#endif
};

int Payment::validate() {
  if (!fields[0]->pos) {
                pos=0;
                message("District ID is a required field");
                return 0;
  }
  if (fields[1]->pos) {
#if defined(USE_BYNAME)
                data->byname = 0;
#endif
  } else if (fields[2]->pos) {
#if defined(USE_BYNAME)
                data->byname = 1;
#endif
  } else {
                pos=1;
                message("Customer ID or Name is required");
                return 0;
  }

  if (!fields[3]->pos) {
                pos=3;
                message("Customer Warehouse is a required field");
                return 0;
  }

  if (!fields[4]->pos) {
                pos=4;
                message("Customer District is a required field");
                return 0;
  }

  if (data->H_AMOUNT <= 0) {
                pos=5;
                message("Enter a positive amount");
                return 0;
  }

  data->W_ID = user_data->warehouse;

  return 1;
}

int Payment::respond() {
  if (data->status != TRAN_OK) {
                display_status(data->status);
                return -1;
  }

  char buf[32];
  position( 52, 6); format_int(buf, 3, data->D_ID);   io.write(buf, 2);
  position( 33,11); format_int(buf, 5, data->C_W_ID); io.write(buf, 4);
  position( 54,11); format_int(buf, 3, data->C_D_ID); io.write(buf, 2);
  position(  7, 4); io.write( data->H_DATE );
  position(  1, 7); io.write( data->W_STREET_1);
  position( 42, 7); io.write( data->D_STREET_1);
  position(  1, 8); io.write( data->W_STREET_2);
  position( 42, 8); io.write( data->D_STREET_2);
  position(  1, 9); io.write( data->W_CITY);
  position( 22, 9); io.write( data->W_STATE);
  position( 25, 9); format_zip(buf, 10, data->W_ZIP); io.write(buf, 10);
  position( 42, 9); io.write( data->D_CITY);
  position( 63, 9); io.write( data->D_STATE);
  position( 66, 9); format_zip(buf, 10, data->D_ZIP); io.write(buf, 10);
  position( 11,11); format_int( buf, 5, data->C_ID); io.write(buf, 4);
  position(  9,12); io.write( data->C_FIRST);
  position( 26,12); io.write( data->C_MIDDLE);
  position( 29,12); io.write( data->C_LAST);
  position( 58,12); format_date(buf, 10, data->C_SINCE); io.write( buf, 10);
  position(  9,13); io.write( data->C_STREET_1);
  position( 58,13); io.write( data->C_CREDIT);
  position(  9,14); io.write( data->C_STREET_2);
  position( 58,14); format_float(buf, 6, 2, data->C_DISCOUNT*100); io.write(buf, 6);
  position(  9,15); io.write( data->C_CITY);
  position( 30,15); io.write( data->C_STATE);
  position( 33,15); format_zip(buf, 10, data->C_ZIP); io.write(buf, 10);
  position( 58,15); format_phone(buf, 18, data->C_PHONE ); io.write(buf, 18);
  position( 17,17); format_money( buf, 15, data->H_AMOUNT); io.write(buf, 14);
  position( 55,17); format_money( buf, 16, data->C_BALANCE); io.write(buf, 15);
  position( 17,18); format_money( buf, 15, data->C_CREDIT_LIM); io.write(buf, 14);

  if (data->C_CREDIT[0] == 'B' && data->C_CREDIT[1] == 'C') {
                int i, size = strlen(data->C_DATA);
                for (i = 0; i < 4; i++) {
                  position(12,20+i);
                  io.write(data->C_DATA, (size > 50)?50:size);
                  size -= 50;
                  if (size <= 0) break;

                }
  }

  return 0;
}

/*****************************************************************************
OrderStatus
*****************************************************************************/
OrderStatus::OrderStatus(User_data *ud) : Screen(ud) {
  tran_type = ORDERSTATUS_SERVICE;
  dataptr = data = new OrderStatus_data;
  data_len = sizeof(OrderStatus_data);

  sprintf(static_session_data, "%s%4d", POS(12,4), user_data->warehouse);

  status_x=1;
  status_y=25;

  int pos = 0;
  screen          = static_screen;
  empty_fields    = static_empty_fields;
  session_data    = static_session_data;
  screen_len      = static_screen_len;
  empty_fields_len = static_empty_fields_len;
  session_data_len = static_session_data_len;

  fields = new Field *[4];
  fields[pos++] = genfield( 29,  4,  2, &data->D_ID ); /* District */
  fields[pos++] = genfield( 11,  5,  4, &data->C_ID ); /* Customer ID */
  fields[pos++] = genfield( 44,  5, 16, data->C_LAST ); /* Customer Name */
  fields[pos++] = NULL;
#if defined(USE_SMART_FIELDS)
  fields[1]->ok_func = (int(*)(void*))pos_zero;
  fields[1]->ok_data = &fields[2]->pos;
  fields[2]->ok_func = (int(*)(void*))pos_zero;
  fields[2]->ok_data = &fields[1]->pos;
#endif
};

int OrderStatus::validate() {
  if (!fields[0]->pos) {
                pos=0;
                message("District ID is a required field");
                return 0;
  }
  if (fields[1]->pos) {
#if defined(USE_BYNAME)
                data->byname = 0;
#endif
  } else if (fields[2]->pos) {
#if defined(USE_BYNAME)
                data->byname = 1;
#endif
  } else {
                pos=1;
                message("Customer ID or Name is required");
                return 0;
  }

  data->W_ID = user_data->warehouse;

  return 1;
}

int OrderStatus::respond() {
```

```
    display_status(data->status);
    if (data->status != TRAN_OK)
                    return -1;

    char buf[16];
    position(11, 5); format_int(buf, 5, data->C_ID); io.write(buf, 4);
    position(24, 5); io.write(data->C_FIRST);
    position(41, 5); io.write(data->C_MIDDLE);
    position(44, 5); io.write(data->C_LAST);
    position(15, 6); format_money(buf, 11, data->C_BALANCE); io.write(buf, 10);
    position(15, 8); format_int(buf, 9, data->O_ID); io.write(buf, 8);
    position(38, 8); format_date(buf, 19, data->O_ENTRY_DATE); io.write(buf);
    if (data->O_CARRIER_ID > 0) {
                    position(76, 8);
                    format_int(buf, 3, data->O_CARRIER_ID);
                    io.write(buf, 2);
    }
    for (int i=0; i < data->ol_cnt; i++) {
                    position( 3, i+10);
                    format_int(buf, 5, data->item[i].OL_SUPPLY_W_ID);
                    io.write(buf, 4);

                    position(14, i+10);
                    format_int(buf, 7, data->item[i].OL_I_ID);
                    io.write(buf, 6);

                    position(25, i+10);
                    format_int(buf, 3, data->item[i].OL_QUANTITY);
                    io.write(buf, 2);

                    position(25, i+10);
                    format_int(buf, 3, data->item[i].OL_QUANTITY);
                    io.write(buf, 2);

                    position (32, i+10);
                    format_money(buf, 10, data->item[i].OL_AMOUNT);
                    io.write(buf, 9);

                    position (47, i+10);
                    format_date(buf, 20, data->item[i].OL_DELIVERY_DATE);
                    io.write(buf, 19);
    }

    return 0;
}

/*****************************************************************************
Delivery
*****************************************************************************/
Delivery::Delivery(User_data *ud) : Screen(ud) {
    tran_type = DELIVERY_SERVICE;
    dataptr = data = new Delivery_data;
    data_len = sizeof(Delivery_data);

    sprintf(static_session_data, "%s%4d", POS(12,4), user_data->warehouse);

    status_x = 1;
    status_y = 8;

    int pos = 0;
    screen          = static_screen;
    empty_fields    = static_empty_fields;
    session_data    = static_session_data;
    screen_len      = static_screen_len;
    empty_fields_len = static_empty_fields_len;
    session_data_len = static_session_data_len;

    fields = new Field *[2];
    fields[pos++] = genfield( 17,  6,  2, &data->O_CARRIER_ID ); /* Carrier Number */

    fields[pos++] = NULL;
};

int Delivery::process() {
    char *buf = tux.getbuf();
    memcpy(buf, dataptr, data_len);
    if (tux.atran(tran_type) < 0) {
                    return -1;
    }
    buf = tux.getbuf();
    memcpy(dataptr, buf, data_len);
    return 0;
}

int Delivery::validate() {
    if (!fields[0]->pos) {
                    pos=0;
                    message("Carrier ID is a required field");
                    return 0;
    }
    time((time_t *)&(data->s_queued_time));

    data->W_ID = user_data->warehouse;

    return 1;
}

int Delivery::respond() {
    if (data->status == TRAN_OK) {
                    position(status_x, status_y);
                    io.write("Execution Status: Delivery has been queued");
    } else {
                    display_status(data->status);
                    return -1;
    }
    return 0;
}

/*****************************************************************************
StockLevel
*****************************************************************************/
StockLevel::StockLevel(User_data *ud) : Screen(ud) {
    tran_type = STOCKLEVEL_SERVICE;
    dataptr = data = new StockLevel_data;
    data_len = sizeof(StockLevel_data);

    sprintf(static_session_data, "%s%4d%s%2d", POS(12,4), user_data->warehouse,
                                                                    POS(29,4),
user_data->district);

    status_x = 1;
    status_y = 10;

    int pos = 0;
    screen          = static_screen;
    empty_fields    = static_empty_fields;
    session_data    = static_session_data;
    screen_len      = static_screen_len;
    empty_fields_len = static_empty_fields_len;
    session_data_len = static_session_data_len;

    fields = new Field *[2];
    fields[pos++] = genfield( 24,  6,  2, &data->threshold ); /* Threshold */
    fields[pos++] = NULL;
};

int StockLevel::validate() {
    if (data->threshold <= 0) {
                    pos=0;
                    message("A positive non-zero threshold is required");
                    return 0;
    }
    data->W_ID = user_data->warehouse;
    data->D_ID = user_data->district;

    return 1;
}

int StockLevel::respond() {
    display_status(data->status);
    if (data->status != TRAN_OK)
                    return -1;

    position(12, 8);
    char buf[5];
    format_int(buf, 4, data->low_stock);
    io.write(buf, 4);

    return 0;
}

/*****************************************************************************
Payment perform
*****************************************************************************/
int Screen::process() {
    if (tran_type == NULL)
                    return 0;
    char *buf = tux.getbuf();
    memcpy(buf, dataptr, data_len);
    if (tux.tran(tran_type) < 0) {
                    return -1;
    }
    buf = tux.getbuf();
    memcpy(dataptr, buf, data_len);
    return 0;
}


/*****************************************************************************
Login
*****************************************************************************/
Login::Login(User_data *ud) : Screen(ud) {
    tran_type = NULL;
    status_x=1;
    status_y=24;

    int pos = 0;
    screen          = static_screen;
    screen_len      = static_screen_len;
    empty_fields    = static_empty_fields;
    empty_fields_len = static_empty_fields_len;

    fields = new Field *[3];
    fields[pos++] = genfield( 16,  5,  4, &(user_data->warehouse) ); //Warehouse
    fields[pos++] = genfield( 34,  5,  2, &(user_data->district)  ); //District
    fields[pos++] = NULL;
};
int Login::validate() {
```

```
    if (!fields[0]->pos) {
                pos=0;
                message("Warehouse ID is a required field");
                return 0;
    }
    if (!fields[1]->pos) {
                pos=1;
                message("District ID is a required field");
                return 0;
    }

    return 1;
}


/*************************************************************************
Menu
*************************************************************************/
Menu::Menu() : Screen(NULL) {
    tran_type = NULL;
    status_x=1;
    status_y=24;

    int pos = 0;
    screen          = static_screen;
    screen_len      = static_screen_len;
    empty_fields    = NULL;
    empty_fields_len = 0;


    fields = NULL;
};

/*************************************************************************
Static data
*************************************************************************/
char const * const blanks      = "                                        ";
char const * const underscores = "________________________________________";
char const * const backspaces  = "\b\b\b\b\b\b\b\b\b\b";

/*************************************************************************
Utility Functions
*************************************************************************/
int string_empty(char const *data) {
    return data[0] == 0;
}
int pos_zero(int const *val) {
    return *val == 0;
}
int pos_nonzeros(int const **val) {
    int const **ptr;
    for (ptr = val; *ptr; ptr++) {
                if (**ptr == 0)
                    return 0;
    }
    return 1;
}

int position(int x, int y, char *buf) {
    int pos = 0;
    buf[pos++] = ESCc;
    buf[pos++] = '[';
    if (y >= 10) buf[pos++] = (y / 10) + '0';
    buf[pos++] = (y % 10) + '0';
    buf[pos++] = ';';
    if (x >= 10) buf[pos++] = (x / 10) + '0';
    buf[pos++] = (x % 10) + '0';
    buf[pos++] = 'H';
    buf[pos++] = 0;
    return 0;
}

int position(int x, int y) {
    char buf[16];
    position(x, y, buf);
    io.write(buf);
    return 0;
}

int clear_eos() {
    io.write (ESC "[J");
    return 0;
}

int message(char const *text, int need_flush) {
    position(1,25);
    io.write(text);
    clear_eos();
    if (need_flush)
                io.flush();
    return 0;
}

int clear_eos(char *buf) {
    buf[0] = ESCc;
    buf[1] = '[';
    buf[2] = 'J';
    return 0;
}
```

```
/*************************************************************************/
/* screen_data.cpp                                          03/31/97 */
/*************************************************************************/
#include "screen.h"


char const NewOrder::static_screen[] =
    POS( 1, 3) CLEAR_EOS
    POS(36, 3) "New Order"
    POS( 1, 4) "Warehouse"
    POS(19, 4) "District:"
    POS(55, 4) "Date:"
    POS( 1, 5) "Customer:"

    POS(19, 5) "Name:"
    POS(44, 5) "Credit:"
    POS(57, 5) "Disc.:"
    POS( 1, 6) "Order Number:"
    POS(25, 6) "Number of Lines:"
    POS(52, 6) "W_Tax:"
    POS(67, 6) "D_Tax:"
    POS( 2, 8) "Supp_W Item_Num Item_Name"
    POS(44, 8) "Qty  Stock  B/G   Price   Amount"
;

char const NewOrder::static_empty_fields[] =
    POS(29, 4) "____" /* District */
    POS(12, 5) "____" /* Customer */

    POS( 3, 9) "____"
    POS(10, 9) "______"
    POS(45, 9) "__"
    POS( 3,10) "____"
    POS(10,10) "______"
    POS(45,10) "__"
    POS( 3,11) "____"
    POS(10,11) "______"
    POS(45,11) "__"
    POS( 3,12) "____"
    POS(10,12) "______"
    POS(45,12) "__"
    POS( 3,13) "____"
    POS(10,13) "______"
    POS(45,13) "__"
    POS( 3,14) "____"
    POS(10,14) "______"
    POS(45,14) "__"
    POS( 3,15) "____"
    POS(10,15) "______"
    POS(45,15) "__"
    POS( 3,16) "____"
    POS(10,16) "______"
    POS(45,16) "__"
    POS( 3,17) "____"
    POS(10,17) "______"
    POS(45,17) "__"
    POS( 3,18) "____"
    POS(10,18) "______"
    POS(45,18) "__"
    POS( 3,19) "____"
    POS(10,19) "______"
    POS(45,19) "__"
    POS( 3,20) "____"
    POS(10,20) "______"
    POS(45,20) "__"
    POS( 3,21) "____"
    POS(10,21) "______"
    POS(45,21) "__"
    POS( 3,22) "____"
    POS(10,22) "______"
    POS(45,22) "__"
    POS( 3,23) "____"
    POS(10,23) "______"
    POS(45,23) "__"
;

char NewOrder::static_session_data[] =
    POS(12,4) "####" /* Warehouse Id */
;

int NewOrder::static_screen_len = sizeof(NewOrder::static_screen) - 1;
int NewOrder::static_empty_fields_len = sizeof(NewOrder::static_empty_fields) - 1;
int NewOrder::static_session_data_len = sizeof(NewOrder::static_session_data) - 1;

/* Payment */
char const Payment::static_screen[] =
    POS( 1, 3) CLEAR_EOS
    POS( 38,3) "Payment"
    POS( 1,4) "Date:"
    POS( 1,6) "Warehouse:"
    POS( 42,6) "District:"
    POS( 1,11) "Customer:"
    POS( 17,11) "Cust-Warehouse:"
    POS( 39,11) "Cust-District:"
    POS( 1,12) "Name:"
    POS( 50,12) "Since:"
    POS( 50,13) "Credit:"
    POS( 50,14) "%Disc:"
```

```
    POS( 50,15) "Phone:"
    POS( 1,17) "Amount Paid:"
    POS( 37,17) "New Cust-Balance:"
    POS( 1,18) "Credit Limit:"
    POS( 1,20) "Cust-Data:"
;
char const Payment::static_empty_fields[] =
    POS( 52, 6) "__" /* District */
    POS( 11,11) "____" /* Customer # */
    POS( 33,11) "____" /* Cust-Warehouse */
    POS( 54,11) "__" /* Cust-District */
    POS( 29,12) "__________________" /* Name */
    POS( 23,17) "________" /* Amount Paid */
;

char Payment::static_session_data[] =
    POS( 12,6) "####" /* Warehouse */
;

int Payment::static_screen_len = sizeof(Payment::static_screen) - 1;
int Payment::static_empty_fields_len = sizeof(Payment::static_empty_fields) - 1;
int Payment::static_session_data_len = sizeof(Payment::static_session_data) - 1;

/* Order Status */
char const OrderStatus::static_screen[] =
    POS( 1, 3) CLEAR_EOS
    POS(35, 3) "Order-Status"
    POS( 1, 4) "Warehouse:"
    POS(19, 4) "District:"
    POS( 1, 5) "Customer:"
    POS(18, 5) "Name:"
    POS( 1, 6) "Cust-Balance:"
    POS( 1, 8) "Order-Number"
    POS(26, 8) "Entry-Date:"
    POS(60, 8) "Carrier-Number:"
    POS( 1, 9) "Supply-W"
    POS(14, 9) "Item-Num"
    POS(25, 9) "Qty"
    POS(33, 9)"Amount"
    POS(45, 9) "Delivery-Date"
;
char const OrderStatus::static_empty_fields[] =
    POS(29, 4) "__" /* District */
    POS(11, 5) "____" /* Customer ID */
    POS(44, 5) "__________________" /* Customer Name */
;

char OrderStatus::static_session_data[] =
    POS(12, 4) "####" /* Warehouse */
;
int OrderStatus::static_screen_len = sizeof(OrderStatus::static_screen) - 1;
int OrderStatus::static_empty_fields_len = sizeof(OrderStatus::static_empty_fields) - 1;
int OrderStatus::static_session_data_len = sizeof(OrderStatus::static_session_data) - 1;

/* Delivery */
char const Delivery::static_screen[] =
    POS(1,3) CLEAR_EOS
    POS( 38,3) "Delivery"
    POS( 1,4) "Warehouse:"
    POS( 1,6) "Carrier Number:"
;
char const Delivery::static_empty_fields[] =
    POS( 17,6) "__" /* Carrier Number */
;
char Delivery::static_session_data[] =
    POS( 12, 4) "####" /* Warehouse */
;
int Delivery::static_screen_len = sizeof(Delivery::static_screen) - 1;
int Delivery::static_empty_fields_len = sizeof(Delivery::static_empty_fields) - 1;
int Delivery::static_session_data_len = sizeof(Delivery::static_session_data) - 1;

/* Stock level */
char const StockLevel::static_screen[] =
    POS( 1, 3) CLEAR_EOS
    POS(35, 3) "Stock-Level"
    POS( 1, 4) "Warehouse:"
    POS(19, 4) "District:"
    POS( 1, 6) "Stock Level Threshold:"
    POS( 1, 8) "Low Stock:"
;

char const StockLevel::static_empty_fields[] =
    POS( 24,6) "__" /* Threshold */
;

char StockLevel::static_session_data[] =
    POS( 12,4) "####" /* Warehouse */
    POS( 29,4) "##"   /* District */
;
int StockLevel::static_screen_len = sizeof(StockLevel::static_screen) - 1;
int StockLevel::static_empty_fields_len = sizeof(StockLevel::static_empty_fields) - 1;
int StockLevel::static_session_data_len = sizeof(StockLevel::static_session_data) - 1;

/* Login */
char const Login::static_screen[] =
    POS( 1, 1) CLEAR_EOS
    POS(30, 3) "Please login."
    POS( 5, 5) "Warehouse:"
    POS(24, 5) "District:"
;
char const Login::static_empty_fields[] =
    POS( 16,5) "____" /* Warehouse */
    POS( 34,5) "__" /* District */
;
int Login::static_screen_len = sizeof(Login::static_screen) - 1;
int Login::static_empty_fields_len = sizeof(Login::static_empty_fields) - 1;

/* Menu */
char const Menu::static_screen[] =
    POS(1, 1) CLEAR_EOS
    "(1)New-Order (2)Payment (3)Order-Status (4)Delivery (5)StockLevel (9)Exit"
;

int Menu::static_screen_len = sizeof(Menu::static_screen) - 1;

/* end string */
char const Screen::end_str[] = "\033[H\n";
int Screen::end_str_len = sizeof(Screen::end_str) - 1;

/*****************************************************************************/
/* screen.h                                                  03/31/97 */
/*****************************************************************************/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <termios.h>
#include <time.h>

#include "field.h"
#include "inout.h"
#include "tpcc.h"

class User_data {
public:
    int warehouse;
    int district;
};

class Screen {
protected:
    static char const end_str[];
    static int        end_str_len;
          int         has_data;
    void       *dataptr;
    char       *tran_type;
    char const *screen;
    char const *empty_fields;
    char const *session_data;
    int         screen_len;
    int         session_data_len;
    int         empty_fields_len;
    int         pos;
    int         status_x, status_y;
    int         data_len;
public:
    User_data  *user_data;
    Field **fields;
    virtual char const *isa() { return "Screen"; };
    virtual int reset();
    virtual int present();
    virtual int present_empty_fields();
    virtual int process();
    virtual int user_input();
    virtual int validate()            { return 1; };
    virtual int respond()             { return 0; };
    int handle();
    int display_status(int status);
    Screen(User_data *ud) {
                    user_data = ud;
                    has_data = 0;
                    pos = 0;
                    fields = NULL;
                    screen = empty_fields = session_data = NULL;
                    screen_len = session_data_len = empty_fields_len = 0;
    };
    virtual ~Screen();
};

class Login : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    int validate();
    Login::Login(User_data *ud);
};

class NewOrder : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
```

```
    static int  static_empty_fields_len;
    static int  static_session_data_len;
    int start_field;
public:
    NewOrder_data *data;

    int reset();
    NewOrder::NewOrder(User_data *ud);
    int validate();
    int respond();
};

class Payment : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    Payment_data *data;
    int validate();
    int respond();
    Payment(User_data *ud);
};

class OrderStatus : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    OrderStatus_data *data;
    int validate();
    int respond();

    OrderStatus(User_data *ud);
};

class Delivery : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    Delivery_data *data;
    int validate();
    int process();
    int respond();

    Delivery(User_data *ud);
};

class StockLevel : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    StockLevel_data *data;
    int validate();
    int respond();

    StockLevel(User_data *ud);
};

class Menu : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int  static_screen_len;
    static int  static_empty_fields_len;
    static int  static_session_data_len;
public:
    Menu();
};
/*****************************************************************************/
/* field.cpp                                          03/31/97 */
/*****************************************************************************/
#include <stdio.h>
#include "field.h"
#include "inout.h"
#include "format.h"

extern InOut io;
extern char const * const blanks;
extern char const * const underscores;
extern char const * const backspaces;

Field *genfield(int x, int y, int len, int *ptr) {
    return new IntField(x, y, len, ptr);
}
Field *genfield(int x, int y, int len, short *ptr) {
    return new ShortField(x, y, len, ptr);
}
Field *genfield(int x, int y, int len, long *ptr) {
    return new LongField(x, y, len, ptr);
}
Field *genfield(int x, int y, int len, char *ptr) {
    return new TextField(x, y, len, ptr);
}
Field *genfield(int x, int y, int len, double *ptr) {
    return new MoneyField(x, y, len, ptr);
}
Field *genfield(int x, int y, int len, unsigned char *ptr) {
    return new Int8Field(x, y, len, ptr);
}

/*****************************************************************************
Field
*****************************************************************************/
Field::Field(int size, char *str)
    : len(size), pos(0), changed(0), need_redisplay(0)
{
    need_free_string = need_free = 0;
    if (str == NULL) {
                string = new char[len+1];
                need_free_string = 1;
    } else {
                string = str;
    }
    ok_func = NULL;
    ok_data = NULL;
    string[0] = 0;
}

Field::Field(int x, int y, int size, char *str)
    : x(x), y(y), len(size), pos(0), changed(0), need_redisplay(0)
{
    need_free_string = need_free = 0;
    if (str == NULL) {
                string = new char[len+1];
                need_free_string = 1;
    } else {
                string = str;
    }
    ok_func = NULL;
    ok_data = NULL;
    string[0] = 0;
}

int Field::reset() {
    pos=0;
    changed=0;
    return 0;
}
Field::~Field() {
    if (need_free_string)
                delete [] string;
}
int Field::finalize_field() {
    changed = 0;
    string[pos] = 0;
    return 0;
}
int Field::display_field(int use_underscores) {
    position(x,y);
    io.write(string);
    if (use_underscores) {
                io.write(underscores, len-pos);
    } else {
                io.write(blanks, len-pos);
    }
    return 0;
}
int Field::get_key() {
    char key;
    io.read(&key, 1);
    return key;
}
int Field::add_char(int key) {
    if (pos >= len || (!isprint(key) && key != ' ')) {
                io.write("\a", 1);
                return 1;
    }
    changed = 1;
    string[pos] = key;
    io.write(&string[pos++], 1);
    return 0;
}
int Field::backspace() {
    io.write("\b_\b", 3);
    changed = 1;
    pos--;
    return 0;
}
int Field::start_position () {
```

```
    position(x, y);
    return 0;
}
int Field::get_field (int need_pos) {
    int key;

    if (need_pos)
                position(x, y);
    if (pos != 0) {
                need_redisplay = 1;
                io.write(string, pos);
                io.write(underscores, len-pos);
                if (len-pos < 6)
                    io.write(backspaces, len-pos);
                else
                    position(x+pos, y);
    }

    io.mark();
    while (1) {
                key = get_key();
                switch(key) {
                case EOF:
                    return EOF;

                case '\r':   /* Carriage Return */
                case '\n':   /* Newline */
                    io.hold();
                    if (changed) {
                                finalize_field();
                    }
                    io.pop();
                    display_field(1);
                    return ENTER;
                    break;

                case '\t':   /* Tab */
                case '\006': /* Ctrl-F */
                case '\016': /* Ctrl-N */
                    if (changed) {
                                finalize_field();
                    }
                    io.pop();
                    display_field(1);
                    return NEXT_FIELD;
                    break;

                case '\002': /* Ctrl-B */
                case '\020': /* Ctrl-P */
                    if (changed) {
                                finalize_field();
                    }
                    io.pop();
                    display_field(1);
                    return PREV_FIELD;

                case '\b':   /* Backspace */
                case '\177': /* Del */
                    if (pos > 0) {
                                backspace();
                    } else
                                io.write("\a", 1);
                    break;

                case '\014': /* Ctrl-L */
                    io.pop();
                    return REDISPLAY;

                case '\030': /* Ctrl-X */
                case '\003': /* Ctrl-C */
                    io.unmark();
                    return ABORT;

                default:
                    add_char(key);
                }
    }
}

/*****************************************************************************
IntField
*****************************************************************************/
IntField::IntField(int x, int y, int size, int *val) : Field(x, y, size), value(val) {
    if (value==NULL) {
                value = new int;
                need_free=1;
    }
}
IntField::IntField(int size, int *val) : Field(size), value(val) {
    if (value==NULL) {
                value = new int;
                need_free=1;
    }
}
IntField::~IntField() {
    if (need_free)
                delete value;
}
int IntField::add_char(int key) {
    if (pos < len && isdigit(key)) {
                changed = 1;
                string[pos] = key;
                io.write(&string[pos++], 1);
                return 0;
    }
    io.write("\a", 1);
    return 1;
}
int IntField::display_field(int use_underscores) {
    int firstchar;
#if USE_ALLOCA
    char *buf = (char *)alloca(len+1);
#else
    char buf[len+1];
#endif
    memset(buf, 'x', len);
    if (pos)
                firstchar = format_int(buf, len+1, *value);
    else
                firstchar = len;
    position(x, y);
    if (use_underscores) {
                io.write(underscores, firstchar);
                io.write(buf+firstchar, len-firstchar);
    } else {
                io.write(buf, len);
    }
    return 0;
}
int IntField::finalize_field() {
    changed = 0;
    string[pos] = 0;
    if (value != NULL)
                *value = atoi(string);
    return 0;
}

/*****************************************************************************
ShortField
*****************************************************************************/
ShortField::ShortField(int x, int y, int size, short *val) : Field(x, y, size), value(val) {
    if (value==NULL) {
                value = new short;
                need_free=1;
    }
}
ShortField::ShortField(int size, short *val) : Field(size), value(val) {
    if (value==NULL) {
                value = new short;
                need_free=1;
    }
}
ShortField::~ShortField() {
    if (need_free)
                delete value;
}

int ShortField::add_char(int key) {
    if (pos < len && isdigit(key)) {
                changed = 1;
                string[pos] = key;
                io.write(&string[pos++], 1);
                return 0;
    }
    io.write("\a", 1);
    return 1;
}
int ShortField::display_field(int use_underscores) {
    int firstchar;
#if USE_ALLOCA
    char *buf = (char *)alloca(len+1);
#else
    char buf[len+1];
#endif
    if (pos)
                firstchar = format_short(buf, len+1, *value);
    else
                firstchar = len;
    position(x, y);
    if (use_underscores) {
                io.write(underscores, firstchar);
                io.write(buf+firstchar, len-firstchar);
    } else {
                io.write(buf, len);
    }
    return 0;
}
int ShortField::finalize_field() {
    changed = 0;
    string[pos] = 0;
    if (value != NULL)
                *value = atoi(string);
    return 0;
}

/*****************************************************************************
ShortField
```

```
*****************************************************************************/
Int8Field::Int8Field(int x, int y, int size, unsigned char *val) : Field(x, y, size), value(val) {
   if (value==NULL) {
                value = new unsigned char;
                need_free=1;
   }
}
Int8Field::Int8Field(int size, unsigned char *val) : Field(size), value(val) {
   if (value==NULL) {
                value = new unsigned char;
                need_free=1;
   }
}
Int8Field::~Int8Field() {
   if (need_free)
                delete value;
}

int Int8Field::add_char(int key) {
   if (pos < len && isdigit(key)) {
                changed = 1;
                string[pos] = key;
                io.write(&string[pos++], 1);
                return 0;
   }
   io.write("\a", 1);
   return 1;
}
int Int8Field::display_field(int use_underscores) {
   int firstchar;
#if USE_ALLOCA
   char *buf = (char *)alloca(len+1);
#else
   char buf[len+1];
#endif
   if (pos)
                firstchar = format_char(buf, len+1, *value);
   else
                firstchar = len;
   position(x, y);
   if (use_underscores) {
                io.write(underscores, firstchar);
                io.write(buf+firstchar, len-firstchar);
   } else {
                io.write(buf, len);
   }
   return 0;
}
int Int8Field::finalize_field() {
   changed = 0;
   string[pos] = 0;
   if (value != NULL)
                *value = atoi(string);
   return 0;
}

/*****************************************************************************
LongField
*****************************************************************************/
LongField::LongField(int x, int y, int size, long *val) : Field(x, y, size), value(val) {
   if (value==NULL) {
                value = new long;
                need_free=1;
   }
}
LongField::LongField(int size, long *val) : Field(size), value(val) {
   if (value==NULL) {
                value = new long;
                need_free=1;
   }
}
LongField::~LongField() {
   if (need_free)
                delete value;
}
int LongField::add_char(int key) {
   if (pos < len && isdigit(key)) {
                changed = 1;
                string[pos] = key;
                io.write(&string[pos++], 1);
                return 0;
   }
   io.write("\a", 1);
   return 1;
}
int LongField::display_field(int use_underscores) {
   int firstchar;
#if USE_ALLOCA
   char *buf = (char *)alloca(len+1);
#else
   char buf[len+1];
#endif
   if (pos)
                firstchar = format_long(buf, len+1, *value);
   else
                firstchar = len;
   position(x, y);
   if (use_underscores) {
                io.write(underscores, firstchar);
                io.write(buf+firstchar, len-firstchar);
   } else {
                io.write(buf, len);
   }
   return 0;
}
int LongField::finalize_field() {
   changed = 0;
   string[pos] = 0;
   if (value != NULL)
                *value = atoi(string);
   return 0;
}

/*****************************************************************************
MoneyField
*****************************************************************************/
MoneyField::MoneyField(int x, int y, int size, double *val) : Field(x, y, size), value(val) {
   seen_dollar = seen_sign = seen_dot = seen_digit = 0;
   if (value==NULL) {
                value = new double;
                need_free=1;
   }
}
MoneyField::MoneyField(int size, double *val) : Field(size), value(val) {
   seen_dollar = seen_sign = seen_dot = seen_digit = 0;
   if (value==NULL) {
                value = new double;
                need_free=1;
   }
}
MoneyField::~MoneyField() {
   if (need_free)
                delete value;
}
int MoneyField::add_char(int key) {
   do {
                if (pos >= len)
                   break;
                if (key == '$') {
                   if (!(pos == 0 || (pos == 1 && seen_sign))) break;
                   seen_dollar = 1;
                } else if (key == '-') {
                   if (!(pos == 0 || (pos == 1 && seen_dollar))) break;
                   seen_sign = 1;
                } else if (key == '.') {
                   if (seen_dot) break;
                   seen_dot = 1;
                } else if (!isdigit(key))
                   break;
                if (seen_dot) {
                   if (seen_dot >= 4)
                                break;
                   seen_dot++;
                }
                changed = 1;
                string[pos] = key;
                io.write(&string[pos++], 1);
                return 0;
   } while (0);
   io.write("\a", 1);
   return 1;
}
int MoneyField::backspace() {
   io.write("\b_\b", 3);
   changed = 1;
   pos--;
   if (seen_dot)
                seen_dot--;
   if (string[pos] == '-')
                seen_sign = 0;
   if (string[pos] == '$')
                seen_dollar = 0;
   if (string[pos] == '.')
                seen_dot = 0;
   return 0;
}
int MoneyField::display_field(int use_underscores) {
   int firstchar;
#if USE_ALLOCA
   char *buf = (char *)alloca(len+1);
#else
   char buf[len+1];
#endif
   if (pos)
                firstchar = format_money(buf, len+1, *value);
   else
                firstchar = len;
   position(x, y);
   if (use_underscores) {
                io.write(underscores, firstchar);
                io.write(buf+firstchar, len-firstchar);
   } else {
                io.write(buf, len);
   }
   return 0;
}
int MoneyField::finalize_field() {
   changed = 0;
```

```
    string[pos] = 0;
    if (value != NULL) {
                *value = atof(string + seen_dollar + seen_sign);
                if (seen_sign)
                    *value = -*value;
    }
    return 0;
}
int MoneyField::reset() {
    Field::reset();
    seen_dollar = seen_sign = seen_dot = seen_digit = 0;
    return 0;
}

/*****************************************************************************
TextField
*****************************************************************************/
TextField::TextField(int x, int y, int size, char *string) : Field(x, y, size, string) {
    value=TextField::string;
}
TextField::TextField(int size, char *string) : Field(size, string) {
    value=TextField::string;
}
int TextField::add_char(int key) {
    if (pos >= len || (!isalnum(key) && key != ' ' && key != '.')) {
                io.write("\a", 1);
                return 1;
    }
    changed = 1;
    string[pos] = key;
    io.write(&string[pos++], 1);
    return 0;
}

/*****************************************************************************/
/* field.h                                              03/31/97 */
/*****************************************************************************/
#if !defined(INCLUDE_FIELD_H)
#define INCLUDE_FIELD_H

class Field {
public:
    enum return_codes { INVALID, ENTER, NEXT_FIELD, PREV_FIELD, ABORT, REDISPLAY };
    int x, y;
    const int len;
    int pos;
    int changed;
    int need_redisplay;
    char *string;
    int (*ok_func)(void *data);
    int need_free;
    int need_free_string;
    void *ok_data;
    Field(int size, char *string=NULL);
    Field(int x, int y, int size, char *string=NULL);
    virtual ~Field();
    virtual int get_field (int need_pos=1);
    static  int get_key ();
    virtual int backspace();
    virtual int reset();
    virtual int start_position();
    virtual int add_char(int key);
    virtual int display_field(int use_underscores=0);
    virtual int finalize_field();

    class Error {
                enum { USER_ABORT };
    };
};

class Int8Field : public Field {
public:
    unsigned char *value;
    int add_char(int key);
    int display_field(int use_underscores=0);
    int finalize_field();

    Int8Field(int x, int y, int size, unsigned char *value=NULL);
    Int8Field(int size, unsigned char *value=NULL);
    virtual ~Int8Field();
};

class ShortField : public Field {
public:
    short *value;
    int add_char(int key);
    int display_field(int use_underscores=0);
    int finalize_field();

    ShortField(int x, int y, int size, short *value=NULL);
    ShortField(int size, short *value=NULL);
    virtual ~ShortField();
};


class IntField : public Field {
public:
    int *value;
    int add_char(int key);
    int display_field(int use_underscores=0);
    int finalize_field();

    IntField(int x, int y, int size, int *value=NULL);
    IntField(int size, int *value=NULL);
    virtual ~IntField();
};

class LongField : public Field {
public:
    long *value;
    int add_char(int key);
    int display_field(int use_underscores=0);
    int finalize_field();

    LongField(int x, int y, int size, long *value=NULL);
    LongField(int size, long *value=NULL);
    virtual ~LongField();
};

class MoneyField : public Field {
public:
    int seen_dollar, seen_sign, seen_dot, seen_digit;
    double *value;
    int add_char(int key);
    int reset();
    int backspace();
    int display_field(int use_underscores=0);
    int finalize_field();
    MoneyField(int x, int y, int size, double *value=NULL);
    MoneyField(int size, double *value=NULL);
    virtual ~MoneyField();
};

class TextField : public Field {
public:
    char *value;
    int add_char(int key);
    TextField(int x, int y, int size, char *value=NULL);
    TextField(int size, char *value=NULL);
};

Field *genfield(int x, int y, int len, int *ptr);
Field *genfield(int x, int y, int len, short *ptr);
Field *genfield(int x, int y, int len, long *ptr);
Field *genfield(int x, int y, int len, char *ptr);
Field *genfield(int x, int y, int len, unsigned char *ptr);
Field *genfield(int x, int y, int len, double *ptr);

#endif /* INCLUDE_FIELD_H */
/*****************************************************************************/
/* format.cpp                                           03/31/97 */
/*****************************************************************************/
#include <string.h>
#include <math.h>

int format_char(char *buf, int size, char val) {
    int neg, pos;
    pos = size;
    buf[--pos] = 0;
    if (val == 0 && pos > 0) {
                buf[--pos] = '0';
                neg = 0;
    } else {
                neg = (val < 0) ? 1 : 0;
                if (neg) val = -val;
                while (val && pos > 0) {
                    buf[--pos] = (val % 10) + '0';
                    val /= 10;
                }
    }
    /* Too long */
    if (!pos && (val || neg)) {
                memset (buf, '*', size);
                return -1;
    }
    if (neg)
                buf[--pos] = '-';
    if (pos)
                memset(buf, ' ', pos);
    return pos;
}
int format_short(char *buf, int size, short val) {
    int neg, pos;
    pos = size;
    buf[--pos] = 0;
    if (val == 0 && pos > 0) {
                buf[--pos] = '0';
                neg = 0;
    } else {
                neg = (val < 0) ? 1 : 0;
                if (neg) val = -val;
                while (val && pos > 0) {
                    buf[--pos] = (val % 10) + '0';
                    val /= 10;
                }
    }
    /* Too long */
    if (!pos && (val || neg)) {
```

```
                memset (buf, '*', size);
                return -1;
    }
    if (neg)
                buf[--pos] = '-';
    if (pos)
                memset(buf, ' ', pos);
    return pos;
}
int format_int(char *buf, int size, int val) {
    int neg, pos;
    pos = size;
    buf[--pos] = 0;
    if (val == 0 && pos > 0) {
                buf[--pos] = '0';
                neg = 0;
    } else {
                neg = (val < 0) ? 1 : 0;
                if (neg) val = -val;
                while (val && pos > 0) {
                    buf[--pos] = (val % 10) + '0';
                    val /= 10;
                }
    }
    /* Too long */
    if (!pos && (val || neg)) {
                memset (buf, '*', size);
                return -1;
    }
    if (neg)
                buf[--pos] = '-';
    if (pos)
                memset(buf, ' ', pos);
    return pos;
}
int format_long(char *buf, int size, long val) {
    int neg, pos;
    pos = size;
    buf[--pos] = 0;
    if (val == 0 && pos > 0) {
                buf[--pos] = '0';
                neg = 0;
    } else {
                neg = (val < 0) ? 1 : 0;
                if (neg) val = -val;
                while (val && pos > 0) {
                    buf[--pos] = (val % 10) + '0';
                    val /= 10;
                }
    }
    /* Too long */
    if (!pos && (val || neg)) {
                memset (buf, '*', size);
                return -1;
    }
    if (neg)
                buf[--pos] = '-';
    if (pos)
                memset(buf, ' ', pos);
    return pos;
}

int format_float(char *buf, int size, int dec, double val) {
    static double pow10[] = { 1, 10, 100, 1000, 10000, 100000, 1000000 };
    int neg, pos;
    pos = size;
    buf[--pos] = 0;
    val = rint(val * pow10[dec]);
    neg = (val < 0) ? 1 : 0;
    if (neg) val = -val;

    while (val >= 1 && pos > 0) {
                if (!dec--) {
                    buf[--pos] = '.';
                    continue;
                }
                buf[--pos] = (int)fmod(val, 10) + '0';
                val /= 10;
    }
    if (dec >= 0) {
                while (dec >= 0 && pos > 0) {
                    if (!dec--) {
                                buf[--pos] = '.';
                    } else {
                                buf[--pos] = '0';
                    }
                }
                if (pos > 0)
                    buf[--pos] = '0';
    }
    /* Too long */
    if (!pos && (val >= 1 || neg)) {
                memset (buf, '*', size);
                return -1;
    }
    if (neg)
                buf[--pos] = '-';
    if (pos)
                memset(buf, ' ', pos);
    return pos;
}

int format_money(char *buf, int size, double val) {
    int pos;
    pos = format_float(buf, size, 2, val);
    if (pos > 0)
                buf[--pos] = '$';
    return pos;
}

int format_date(char *buf, int size, char* val) {
    memcpy(buf, val, size);
    buf[size]=0;
    return 0;
}

int format_phone(char *buf, int size, char *phone) {
    buf[0] = phone[0];
    buf[1] = phone[1];
    buf[2] = phone[2];
    buf[3] = phone[3];
    buf[4] = phone[4];
    buf[5] = phone[5];
    buf[6] = '-';
    buf[7] = phone[6];
    buf[8] = phone[7];
    buf[9] = phone[8];
    buf[10] = '-';
    buf[11] = phone[9];
    buf[12] = phone[10];
    buf[13] = phone[11];
    buf[14] = '-';
    buf[15] = phone[12];
    buf[16] = phone[13];
    buf[17] = phone[14];
    buf[18] = phone[15];
    buf[19] = '\0';
    return size;
}

int format_zip(char *buf, int size, char *zip) {
    buf[0] = zip[0];
    buf[1] = zip[1];
    buf[2] = zip[2];
    buf[3] = zip[3];
    buf[4] = zip[4];
    buf[5] = '-';
    buf[6] = zip[5];
    buf[7] = zip[6];
    buf[8] = zip[7];
    buf[9] = zip[8];
    buf[10] = '\0';
    return size;
}
/*****************************************************************************/
/* format.h                                                  03/31/97 */
/*****************************************************************************/
#if !defined(INCLUDE_FORMAT_H)
#define INCLUDE_FORMAT_H

int format_char (char *buf, int size, char   val);
int format_int  (char *buf, int size, int    val);
int format_long (char *buf, int size, long   val);
int format_short(char *buf, int size, short  val);
int format_float(char *buf, int size, int    dec, double val);
int format_money(char *buf, int size, double val);
int format_date (char *buf, int size, char  *val);
int format_phone(char *buf, int size, char  *phone);
int format_zip  (char *buf, int size, char  *zip);

#endif /* INCLUDE_FORMAT_H */

/*****************************************************************************/
/* inout.cpp                                                 03/31/97 */
/*****************************************************************************/
#include <string.h>
#include <strings.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>

#include "screen.h"

#if 1
void InOut::write(const void *buf, size_t size) {
    debug("write('%*.*s', %d);\n", size, size, buf, size);
    output.queue(buf, size);
    if (!Hold && input.len() == 0) { /* Don't write anything until there is no input */
                flush();
    }
}

ssize_t InOut::read(void *buf, size_t size) {
    int rc;
    while (input.len() < size) {
                rc = ::read(in_fd, input.ptr(), input.free());
        debug("::read('%*.*s', %d) = %d;\n", rc, rc, input.ptr(), input.free(), rc);
```

```
    if (inlog) {
                fwrite(input.ptr(), rc, 1, inlog);
                fflush(inlog);
    }
            if (rc > 0) {
                input.queue(rc);
            } else if (rc <= 0) {
                fprintf (stderr, "Error reading data!\n");
                exit (1);
            }
  }
  memcpy(buf, input.ptr(), size);
  input.dequeue(size);
  debug("read('%*.*s', %d) = %d;\n", size, size, buf, size, size);
  return size;
}
#else
void InOut::write(const void *buf, size_t size) {
  debug("write('%s', %d);\n", buf, size);
  ::write(out_fd, buf, size);
}

ssize_t InOut::read(void *buf, size_t size) {
  int rc;
  rc = ::read(in_fd, buf, size);
  debug("read('%s', %d) = %d;\n", buf, size, rc);
  return rc;
}
#endif

void InOut::flush() {
  debug("flush();\n");
  Hold = 0;
  while (output.len()) {
    debug("::write('%*.*s', %d);\n", output.len(), output.len(), output.ptr(), output.len());
            int rc = ::write(out_fd, output.ptr(), output.len());
    if (outlog) {
                fwrite(output.ptr(), rc, 1, outlog);
                fflush(outlog);
    }
            if (rc > 0) {
                output.dequeue(rc);
            } else if (rc < 0) {
                fprintf (stderr, "Error writing data!\n");
                exit (1);
            }
  }
}

void InOut::write(const void *buf) {
  write(buf, strlen((const char *)buf));
}

InOut::InOut(int in, int out) : input(256), output(2048) {
  struct termios buf;

#ifdef DEBUG
  {
            char buf[256];
            sprintf(buf, "logs/debug.%d", getpid());
            debugfile = fopen(buf, "w");
            sprintf(buf, "logs/in.%d", getpid());
            inlog     = fopen(buf, "w");
            sprintf(buf, "logs/out.%d", getpid());
            outlog    = fopen(buf, "w");
  }
#endif

  Hold = 0;

  in_fd = in;
  if (out < 0)
            out_fd = in;
  else
    out_fd = out;
  if (tcgetattr(in_fd, &save_term) < 0)
            return;

  buf = save_term;

  buf.c_lflag &= ~(ECHO | ICANON); /* echo off, canonical mode off */

  buf.c_cc[VMIN] = 1;    /* Case B: 1 byte at a time, no timer */
  buf.c_cc[VTIME] = 0;

  if (tcsetattr(in_fd, TCSAFLUSH, &buf) < 0)
            return;
}

InOut::~InOut() {
  if (tcsetattr(in_fd, TCSAFLUSH, &save_term) < 0)
            return;
}

/*****************************************************************************/
/* inout.h                                          03/31/97 */
/*****************************************************************************/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <termios.h>
#include <stdarg.h>
#include <string.h>

#include "tpcc.h"

/* This is for a VT100 */
#if 1
#define ESC "\033"
#define ESCc '\033'
#else
#define ESCc '^'
#define ESC "^"
#endif

#define TRIGGER "\021"
#define TRIGGERc '\021'

#define POS(x,y) ESC "[" #y ";" #x "H"
#define CLEAR_EOS ESC "[J"

class InOut {
private:
  class Buffer {
  private:
            int BufSize;
            enum { NUMMARKS=8 };
            char *buffer;
            int marks[NUMMARKS];
  public:
            int Pos;
            int Start;
    int num_marks;
            Buffer(int size) {
              BufSize = size;
              buffer = new char [BufSize];
              Pos = Start = 0;
            }
            int  pos()      { return Pos; };
            void pos(int P)  { Pos = P; };
            int  start()    { return Start; };
            void start(int S) { Start = S; };
            int  len()      { return Pos-Start; };
            int  free()     { return BufSize-Pos-1; };
            void *ptr()     { return &buffer[Start]; };
            int  lastmark()  { if (num_marks) return marks[num_marks-1]; return 999; };

            void  mark()     {
              if (num_marks < NUMMARKS)
                        marks[num_marks++] = Pos;
              else {
                        fprintf(stderr, "Buffer mark overflow\n");
                        exit (1);
              }
            }
            void unmark() {
              if (num_marks <= 0)
                        return;
              num_marks--;
            }
            void pop() {
              if (num_marks <= 0)
                        return;
              if (marks[num_marks-1] >= Start) {
                        Pos=marks[--num_marks];
              } else {
                        num_marks=0;
              }
            }
            void queue(int size) {
              Pos += size;
            }
            void queue(const void *buf, int size) {
              /* If this is too big see if we can move what we have over */
              if (size+Pos >= BufSize) {
                        if (size + len() >= BufSize) {
                          fprintf(stderr, "Buffer overflow\n");
                          exit (1);
                        }
                        /* This requires memcpy to be "safe" */
                        memcpy(buffer, &buffer[Start], len());
                        Pos -= Start;

                        /* Fix up our marks*/
                        int count = 0;
                        for (int i = 0; i < num_marks; i++) {
                          if (marks[i] - Start >= 0)
                                    marks[count++] = marks[i] - Start;
                        }
                        num_marks = count;
                        Start = 0;
              }
              memcpy(&buffer[Pos], buf, size);
              Pos += size;
            }
            void dequeue(int size) {
              Start += size;
```

```
                if (Start >= Pos) {
                        /* Fix up our marks*/
                        int count = 0;
                        for (int i = 0; i < num_marks; i++) {
                           if (marks[i] - Start >= 0)
                                marks[count++] = marks[i] - Start;
                        }
                        num_marks = count;

                        Start = Pos = 0;
                }
        }
  };
  int in_fd, out_fd;
  int Hold;
  struct termios save_term;
  Buffer input;
  Buffer output;
  FILE *debugfile;
  FILE *inlog, *outlog;
public:
  ssize_t read(void *buf, size_t size);
  void write(const void *buf, size_t size);
  void write(const void *buf);
  void flush();
  void mark() { debug("mark()\n"); output.mark(); };
  void unmark() { debug("unmark()\n"); output.unmark(); };
  void pop()  { debug("pop()\n\n"); output.pop(); };
  void hold() { debug("hold()\n");  Hold = 1; };
#ifdef DEBUG
  void debug(char *fmt, ...) {
        va_list args;

    fprintf(debugfile, "Start=%2d, Pos=%2d, Marks=%2d(%03d): ", output.Start, output.Pos,
output.num_marks, output.lastmark());
        va_start(args,fmt);
        vfprintf (debugfile, fmt, args);
        va_end (args);
        ::fflush(debugfile);
  }
#else
  void debug(char *fmt, ...) {};
#endif
  InOut(int in, int out=-1);
  ~InOut();
};

extern InOut io;
extern char const * const blanks;
extern char const * const underscores;
extern char const * const backspaces;

int position(int x, int y, char *buf);
int position(int x, int y);
int clear_eos();
int clear_eos(char *buf);
int format_int(char *buf, int size, int val);
int format_float(char *buf, int size, int dec, double val);
int format_money(char *buf, int size, double val);
int message(char const *text, int need_flush=1);
int string_empty(char const *text);
int pos_zero(int const *val);

/*****************************************************************************/
/* tuxedo.cpp                                        03/31/97 */
/*****************************************************************************/

#include <stdlib.h>
#include "inout.h"
#include "tuxedo.h"

int Tuxedo::init = 0;

extern "C" {
#include "atmi.h"                               /* TUXEDO atmi library */
}


#ifndef FAKEDATABASE
void Tuxedo::cleanup() {
  if (init == 0) {
        if (tpterm() == -1) {
           char         str[128];
           sprintf(str, "tpterm: %s(%d)", tpstrerror(tperrno), tperrno);
           message(str);
           exit(1);
        }
  }
}

Tuxedo::Tuxedo() : TMinbufsize(TMINBUFSIZE), TMbuffer(NULL) {
  if (!init++) {
        const char *ptr;
        ptr = getenv("TUXCONFIG");
        if (ptr == NULL)
           if (putenv ("TUXCONFIG=/home/tpcc/install.tux/tuxconfig"))
                message ("putenv failed!");

        if (tpinit((TPINIT *)NULL) == -1) {
           char         str[128];
           sprintf(str, "tpinit: %s(%d)", tpstrerror(tperrno), tperrno);
           message(str);
           exit (1);
        }
        atexit(Tuxedo::cleanup);
  } else {
        message("Tuxedo::connect already performed!");
        return;
  }

  if ((TMbuffer = tpalloc("CARRAY", NULL, TMinbufsize)) == NULL) {
        char         str[128];
        sprintf(str, "tpalloc: %s(%d)", tpstrerror(tperrno), tperrno);
        message(str);
        exit(1);
  }

  return;
}

Tuxedo::~Tuxedo() {
  return;
}

int Tuxedo::tran (char *servname) {
  long TMoutbufsize;

  if (tpcall(servname,TMbuffer,TMinbufsize,&TMbuffer,&TMoutbufsize,0) == -1) {
        char         str[128];
        sprintf(str, "ERROR: tpcall: %s(%d)", tpstrerror(tperrno), tperrno);
        message(str);
        return -1;
  }

  if (TMoutbufsize < TMinbufsize) {
        char         str[128];
        sprintf(str, "ERROR: tpcall: TMoutbufsize < TMinbufsize");
        message(str);
        return -1;
  }

  return 0;
}

int Tuxedo::atran (char *servname) {
  if (tpacall(servname,TMbuffer,TMinbufsize, TPNOREPLY) == -1) {
        char         str[128];
        sprintf(str, "ERROR: tpacall: %s(%d)", tpstrerror(tperrno), tperrno);
        message(str);
        return -1;
  }

  return 0;
}
#else
int NewOrder_process(NewOrder_data *data) {
  int           i;

  data->s_W_ID = 11;
  data->s_D_ID = 22;
  data->s_C_ID = 3333;
  strcpy(data->s_C_LAST, "1234567890123456");
  strcpy(data->s_C_CREDIT, "BC");
  data->s_C_DISCOUNT = 0.1556;
  data->s_O_OL_CNT = 10;
  data->s_O_ID = 4444;
  strcpy(data->s_O_ENTRY_D, "1992-10-2 12:33:11");
  strcpy(data->s_status_line, "12345678901234567890123 4");
  data->s_total_amount = 12.98;
  data->s_transtatus = 0;
  data->s_W_TAX = 0.1234;
  data->s_D_TAX = 0.5678;

  for (i=0; i < data->s_O_OL_CNT; i++) {
        data->item[i].s_OL_SUPPLY_W_ID = i + 1;
        data->item[i].s_OL_I_ID = i + 1;
        strcpy(data->item[i].s_I_NAME, "12345678901234567890123 4");
        data->item[i].s_OL_QUANTITY = i + 1;
        data->item[i].s_S_QUANTITY = i + 1;
        data->item[i].s_brand_generic = 'B';
        data->item[i].s_I_PRICE = i + 1;
        data->item[i].s_OL_AMOUNT = i + 1;
  }
  return 0;
}

int Payment_process(Payment_data *data) {
  data->s_W_ID = 11;
  data->s_D_ID = 22;
  data->s_C_ID = 3333;
  data->s_C_W_ID = 44;
  data->s_C_D_ID = 55;
  data->s_H_AMOUNT = 9.55;
  strcpy(data->s_W_STREET_1, "12345678901234567890");
  strcpy(data->s_W_STREET_2, "12345678901234567890");
  strcpy(data->s_W_CITY, "1234567890123456789 0");
  strcpy(data->s_W_STATE, "PR");
  strcpy(data->s_W_ZIP, "123456789");
```

```
   strcpy(data->s_D_STREET_1, "12345678901234567890");
   strcpy(data->s_D_STREET_2, "12345678901234567890");
   strcpy(data->s_D_CITY, "12345678901234567890");
   strcpy(data->s_D_STATE, "PR");
   strcpy(data->s_D_ZIP, "123456789");
   strcpy(data->s_C_FIRST, "1234567890123456");
   strcpy(data->s_C_MIDDLE, "12");
   strcpy(data->s_C_LAST, "1234567890123456");
   strcpy(data->s_C_STREET_1, "12345678901234567890");
   strcpy(data->s_C_STREET_2, "12345678901234567890");
   strcpy(data->s_C_CITY, "12345678901234567890");
   strcpy(data->s_C_STATE, "PR");
   strcpy(data->s_C_ZIP, "123456789");
   strcpy(data->s_C_PHONE, "1234567890123456");
   strcpy(data->s_C_SINCE, "1992-23-22 21:11:11");
   strcpy(data->s_H_DATE, "1992-10-2 12:33:11");
   strcpy(data->s_C_CREDIT, "BC");
   data->s_C_CREDIT_LIM = 5000;
   data->s_C_DISCOUNT = 0.10;
   data->s_C_BALANCE = 122.10;
   strcpy(data->s_C_DATA,
"12345678901234567890123456789012345678901234567890123456789012345678901234567890");
   return 0;
}

int OrderStatus_process(OrderStatus_data *data) {
   int             i;

   data->s_W_ID = 11;
   data->s_D_ID = 22;
   data->s_C_ID = 3333;
   strcpy(data->s_C_FIRST, "1234567890123456");
   strcpy(data->s_C_MIDDLE, "12");
   strcpy(data->s_C_LAST, "1234567890123456");
   data->s_C_BALANCE = 122.10;
   data->s_O_ID = 44;
   strcpy(data->s_O_ENTRY_D,"1992-10-2 12:33:11");
   data->s_O_CARRIER_ID = 55;
   data->s_ol_cnt = 10;

   for (i=0; i < data->s_ol_cnt; i++) {
                data->item[i].s_OL_SUPPLY_W_ID = i + 1;
                data->item[i].s_OL_I_ID = i + 1;
                data->item[i].s_OL_QUANTITY = i + 1;
                data->item[i].s_OL_AMOUNT = i + 1;
                strcpy(data->item[i].s_OL_DELIVERY_D, "1992-10-2 12:33:11");
   }
   return 0;
}

int Delivery_process(Delivery_data *data) {
   strcpy(data->s_exec_status, "Delivery has been queued");
   return 0;
}

int StockLevel_process(StockLevel_data *data) {
   data->s_low_stock = 22;
   return 0;
}

int Tuxedo::atran (char *servname) {
   tran(servname);
   return 0;
}
int Tuxedo::tran (char *servname) {
   if (!strcmp(servname, "neword_sql")) {
                NewOrder_process   ((NewOrder_data *)   TMbuffer);
   } else if (!strcmp(servname, "payment_sql")) {
                Payment_process    ((Payment_data *)    TMbuffer);
   } else if (!strcmp(servname, "ordstat_sql")) {
                OrderStatus_process((OrderStatus_data *)TMbuffer);
   } else if (!strcmp(servname, "delivery_sql")) {
                Delivery_process   ((Delivery_data *)   TMbuffer);
   } else if (!strcmp(servname, "stocklev_sql")) {
                StockLevel_process ((StockLevel_data *) TMbuffer);
   } else {
                fprintf(stderr, "Illegal tuxedo transaction '%s'!\n", servname);
                exit(1);
   }
   return 0;
}

Tuxedo::Tuxedo() : TMinbufsize(TMINBUFSIZE), TMbuffer(NULL) {
   if ((TMbuffer = (char *)malloc(TMinbufsize)) == NULL) {
                char            str[128];
                sprintf(str, "tpalloc() = NULL: tperrno: %d", tperrno);
                message(str);
                exit(1);
   }

   return;
}

Tuxedo::~Tuxedo() {
   return;
}
#endif

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

```
/* tuxedo.h                                          03/31/97 */
/*****************************************************************************/

const int TMINBUFSIZE = 1536;
class Tuxedo {
   long  TMinbufsize;
   char *TMbuffer;
   static int  init;
public:
   static void cleanup();
   char *getbuf() { return TMbuffer; };
   int tran(char *servname);
   int atran(char *servname);
   Tuxedo();
   ~Tuxedo();
};
```

A.2 Client Transaction Code

```
/*****************************************************************************/
/* sybase_service.c                                  03/31/97 */
/*****************************************************************************/
/* #define SORT_LINES */

#include <stdio.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <oserror.h>
#include <atmi.h>
#include "tpcc.h"

#define USE_DEADLOCK

#define AMOUNT_IN_CENTS

#ifdef USE_DEADLOCK
#define EXTRA_DEADLOCK_CHECK || deadlock
int deadlock;
#else
#define EXTRA_DEADLOCK_CHECK
#endif

#define BLANKS "\n\n\n\n\n\n"
#define LINE   "--------------------------------------------------------------------\n"

#ifdef NOTUX
void tpreturn(int retval, int zero, char *data, int len, int x) {
}
#endif

DBPROCESS *dbproc;
int bad_transaction;

int err_handler( DBPROCESS *dbproc, int severity, int dberr, int oserr, char *dberrstr, char *oserrstr) {
   fprintf(stderr, "error %d, %s(%d), %s(%d)\n", severity, dberrstr, dberr, oserrstr, oserr);
   exit(2);
}

int msg_handler( DBPROCESS *dbproc, int msgno, int msgstate, int severity,
                                    char *msgtext, char *servername, char *procname, int line) {
   fprintf(stderr, "msg '%s', %d\n", msgtext, msgno);

   if (msgno == SQLSRV_ENVDB || msgno == SQLSRV_ENVLANG ||
                msgno == SQLSRV_ENVCHAR)
                return(SUCCEED);

   if (msgno == 560 || msgno == 515 || msgno == 3621) {
                bad_transaction = 1;
                return(SUCCEED);
   }

#if 0
   if (msgno == ABORT_ERROR)
                return(SUCCEED);
#endif

#ifdef USE_DEADLOCK
   if (msgno == 1205) {
                deadlock = 1;
                return(SUCCEED);
   }
#endif

   if (msgno==0) {
                return(SUCCEED);
   }

   bad_transaction = 1;
   return(SUCCEED);

   /* exit on any error */
}
```

```
#define MaxTries 5

void neword_sql(TPSVCINFO *tux) {
   int                  try;
   int rc;

#ifdef SORT_LINES
   sort_order_lines();
#endif

   bad_transaction = 0;

   for (try=0; try<MaxTries && !bad_transaction; try++) {
                if (!neword_body(tux->data) && !bad_transaction)
                   tpreturn(TPSUCCESS, 0, tux->data, tux->len,0);
                dbcancel(dbproc);
   }
   tpreturn(TPFAIL, 0, tux->data, tux->len,0);
}

void payment_sql(TPSVCINFO *tux) {
   int                  try, rc;
   bad_transaction = 0;

   for (try=0; try<MaxTries && !bad_transaction; try++) {
                if (!payment_body(tux->data) && !bad_transaction)
                   tpreturn(TPSUCCESS, 0, tux->data, tux->len,0);
                dbcancel(dbproc);
   }
   tpreturn(TPFAIL, 0, tux->data, tux->len,0);
}

void delivery_sql(TPSVCINFO *tux) {
   int                  try;
   int rc;
   bad_transaction = 0;

   for (try=0; try<MaxTries && !bad_transaction; try++) {
                if (!delivery_body(tux->data) && !bad_transaction)
                   tpreturn(TPSUCCESS, 0, tux->data, tux->len,0);
                dbcancel(dbproc);
   }
   tpreturn(TPFAIL, 0, tux->data, tux->len,0);
}

void ordstat_sql(TPSVCINFO *tux) {
   int                  try;
   int rc;
   bad_transaction = 0;

   for (try=0; try<MaxTries && !bad_transaction; try++) {
                if (!ordstat_body(tux->data) && !bad_transaction)
                   tpreturn(TPSUCCESS, 0, tux->data, tux->len,0);
                dbcancel(dbproc);
   }
   tpreturn(TPFAIL, 0, tux->data, tux->len,0);
}

void stocklev_sql(TPSVCINFO *tux) {
   int                  try;
   int rc;
   bad_transaction = 0;

   for (try=0; try<MaxTries && !bad_transaction; try++) {
                if (!stocklev_body(tux->data) && !bad_transaction)
                   tpreturn(TPSUCCESS, 0, tux->data, tux->len,0);
                dbcancel(dbproc);
   }
   tpreturn(TPFAIL, 0, tux->data, tux->len,0);
}

int neword_body (NewOrder_data *data) {
   int i;
   DBINT retcode;
   DBDATETIME tmpdate;
   struct itemstruct *item;
   char        i_name[25];
   DBSMALLINT   quantity;
   DBFLT8       i_price;
   char        brand_generic[2];

#ifdef USE_DEADLOCK
   deadlock = 0;
#endif

#ifdef DEBUG
   fprintf(stderr, BLANKS LINE "NewOrder Input\n");
   neword_debug(data);
#endif
   data->status = -1; /* Assume an error unless everything goes well */
   if (data->all_local)
                dbrpcinit(dbproc, "neworder_local", 0);
   else
                dbrpcinit(dbproc, "neworder_remote", 0);

   dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &(data->W_ID));
   dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, &(data->D_ID));
   dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, &(data->C_ID));
   dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, &(data->O_OL_CNT));
   item = data->item;
   for (i = 0; i < data->O_OL_CNT; i++) {
                dbrpcparam(dbproc, NULL, 0, SYBINT4,  -1, -1, &(item->OL_I_ID));
                if (!data->all_local)
                   dbrpcparam(dbproc, NULL, 0, SYBINT2,  -1, -1, &(item->OL_SUPPLY_W_ID));
                dbrpcparam(dbproc, NULL, 0, SYBINT1,  -1, -1, &(item->OL_QUANTITY));
                item++;
   }


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

   item = data->item;
   data->total_amount = 0;
   for (i = 0; i < data->O_OL_CNT; i++) {
                if (dbresults(dbproc) != SUCCEED EXTRA_DEADLOCK_CHECK) return TRUE;
   dbbind(dbproc, 1, NTBSTRINGBIND,  sizeof(i_name), i_name);
   dbbind(dbproc, 2, FLT8BIND,       0, &i_price);
   dbbind(dbproc, 3, SMALLBIND,      0, &quantity);
   dbbind(dbproc, 4, NTBSTRINGBIND,  sizeof(item->brand_generic), &brand_generic);

                if (dbnextrow(dbproc) != REG_ROW) return TRUE;

                strcpy(item->I_NAME, i_name);
                item->I_PRICE  = i_price;
                item->QUANTITY = quantity;
                strcpy(item->brand_generic, brand_generic);

#ifdef AMOUNT_IN_CENTS
                item->I_PRICE /= 100;
#endif
                item->OL_AMOUNT = item->I_PRICE * item->OL_QUANTITY;

                data->total_amount += item->I_PRICE * item->OL_QUANTITY;

                if(i_name[0] == '\0') {
                   bad_transaction = 0; /* This isn't a SQL failure, it's user error */
                   data->status = INVALID_ITEM;
                   dbcancel(dbproc);
                   return FALSE;
                }
                if (dbcanquery(dbproc) != SUCCEED EXTRA_DEADLOCK_CHECK) return TRUE;

                if (dbhasretstat(dbproc)) {
                   int retcode = dbretstatus(dbproc);
#ifdef USE_DEADLOCK
                   if (retcode == -3) deadlock = 1;
#endif
                   return TRUE;
                }
                item++;
   }
   if (dbresults(dbproc) != SUCCEED EXTRA_DEADLOCK_CHECK) return TRUE;
   dbbind(dbproc, 1, REALBIND,                    0,
&data->W_TAX);
   dbbind(dbproc, 2, REALBIND,          0,                    &data->D_TAX);
   dbbind(dbproc, 3, INTBIND,           0,                    &data->O_ID);
   dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(data->C_LAST),    data->C_LAST);
   dbbind(dbproc, 5, REALBIND,          0,                    &data->C_DISCOUNT);
   dbbind(dbproc, 6, NTBSTRINGBIND, sizeof(data->C_CREDIT),  data->C_CREDIT);
   dbbind(dbproc, 7, DATETIMEBIND,      0,                    &tmpdate);
   if (dbnextrow(dbproc)  != REG_ROW EXTRA_DEADLOCK_CHECK) return TRUE;
   if (dbcanquery(dbproc) != SUCCEED EXTRA_DEADLOCK_CHECK) return TRUE;

   data->total_amount *= (1 - data->C_DISCOUNT) *
                                                (1 + data->W_TAX + data->D_TAX);
   format_sybase_date(&tmpdate, &data->O_ENTRY_DATE);

   data->status = TRAN_OK; /* Everything is ok */
#ifdef DEBUG
   fprintf(stderr, LINE "NewOrder Output\n");
   neword_debug(data);
#endif
   return FALSE;
}

int payment_body (Payment_data *data) {
   DBDATETIME tmpdate, tmpdate2;

#ifdef DEBUG
   fprintf(stderr, BLANKS LINE "Payment Input\n");
   payment_debug(data);
#endif

   data->status = -1; /* Assume an error unless everything goes well */
#ifdef USE_DEADLOCK
   deadlock = 0;
#endif
   if (data->C_LAST[0])
                dbrpcinit(dbproc, "payment_byname", 0);
   else
                dbrpcinit(dbproc, "payment_byid", 0);

   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &data->W_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &data->C_W_ID);
   dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, &data->H_AMOUNT);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &data->D_ID);
```

```
  dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &data->C_D_ID);
  if (data->C_LAST[0])
                dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(data->C_LAST), data->C_LAST);
  else
                dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &data->C_ID);

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

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

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

  dbbind(dbproc, 14, NTBSTRINGBIND, sizeof(data->C_FIRST),     data->C_FIRST);
  dbbind(dbproc, 15, NTBSTRINGBIND, sizeof(data->C_MIDDLE),    data->C_MIDDLE);
  dbbind(dbproc, 16, NTBSTRINGBIND, sizeof(data->C_STREET_1),  data->C_STREET_1);
  dbbind(dbproc, 17, NTBSTRINGBIND, sizeof(data->C_STREET_2),  data->C_STREET_2);
  dbbind(dbproc, 18, NTBSTRINGBIND, sizeof(data->C_CITY),      data->C_CITY);
  dbbind(dbproc, 19, NTBSTRINGBIND, sizeof(data->C_STATE),     data->C_STATE);
  dbbind(dbproc, 20, NTBSTRINGBIND, sizeof(data->C_ZIP),       data->C_ZIP);
  dbbind(dbproc, 21, NTBSTRINGBIND, sizeof(data->C_PHONE),     data->C_PHONE);
  dbbind(dbproc, 22, DATETIMEBIND,  0,                    &tmpdate2);
  dbbind(dbproc, 23, NTBSTRINGBIND, sizeof(data->C_CREDIT),    data->C_CREDIT);
  dbbind(dbproc, 24, FLT8BIND,      0,                    &data->C_CREDIT_LIM);
  dbbind(dbproc, 25, REALBIND,      0,                    &data->C_DISCOUNT);
  dbbind(dbproc, 26, FLT8BIND,      0,                    &data->C_BALANCE);
  dbbind(dbproc, 27, NTBSTRINGBIND, sizeof(data->C_DATA),      data->C_DATA);
  if (dbnextrow(dbproc)  != REG_ROW EXTRA_DEADLOCK_CHECK) return TRUE;
  if (dbcanquery(dbproc) != SUCCEED EXTRA_DEADLOCK_CHECK) return TRUE;

  format_sybase_date(&tmpdate,  &data->H_DATE);
  format_sybase_date(&tmpdate2, &data->C_SINCE);

  data->status = TRAN_OK; /* Everything is ok */
#ifdef DEBUG
  fprintf(stderr, LINE "Payment Output\n");
  payment_debug(data);
#endif
  return FALSE;
}

int ordstat_body (OrderStatus_data *data) {
  DBSMALLINT    ol_supply_w_id;
  DBINT         ol_i_id;
  DBTINYINT     ol_quantity;
  DBFLT8        ol_amount;
  DBDATETIME    tmpdate;

  struct oitemstruct *item;

  data->status = -1; /* Assume an error unless everything goes well */
#ifdef DEBUG
  fprintf(stderr, BLANKS LINE "OrderStatus Input\n");
  ordstat_debug(data);
#endif

#ifdef USE_DEADLOCK
  deadlock = 0;
#endif
  if (data->C_LAST[0])
                dbrpcinit(dbproc, "order_status_byname", 0);
  else
                dbrpcinit(dbproc, "order_status_byid", 0);
  dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &data->W_ID);
  dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &data->D_ID);
  if (data->C_LAST[0])
                dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1, strlen(data->C_LAST), data->C_LAST);
  else
                dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, &data->C_ID);

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

  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,  &tmpdate);
  item = data->item;
  for (data->ol_cnt = 0; dbnextrow(dbproc) == REG_ROW
#ifdef USE_DEADLOCK
                && !deadlock
#endif
                 ; data->ol_cnt++) {
                item->OL_SUPPLY_W_ID = ol_supply_w_id;
                item->OL_I_ID        = ol_i_id;
                item->OL_QUANTITY    = ol_quantity;
                item->OL_AMOUNT      = ol_amount;
#ifdef AMOUNT_IN_CENTS
                item->OL_AMOUNT /= 100;
#endif
                format_sybase_date(&tmpdate, item->OL_DELIVERY_DATE);
                item++;
  }
 if (dbresults(dbproc) != SUCCEED     EXTRA_DEADLOCK_CHECK) return TRUE;

  dbbind(dbproc, 1, INTBIND,      0,                    &data->C_ID);
  dbbind(dbproc, 2, NTBSTRINGBIND, sizeof(data->C_LAST),    data->C_LAST);
  dbbind(dbproc, 3, NTBSTRINGBIND, sizeof(data->C_FIRST),   data->C_FIRST);
  dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(data->C_MIDDLE),  data->C_MIDDLE);
  dbbind(dbproc, 5, FLT8BIND,     0,                    &data->C_BALANCE);
  dbbind(dbproc, 6, INTBIND,      0,                    &data->O_ID);
  dbbind(dbproc, 7, DATETIMEBIND, 0,                    &tmpdate);
  dbbind(dbproc, 8, SMALLBIND,    0,                    &data->O_CARRIER_ID);

  if (dbnextrow(dbproc)  != REG_ROW EXTRA_DEADLOCK_CHECK) return TRUE;
  if (dbcanquery(dbproc) != SUCCEED EXTRA_DEADLOCK_CHECK) return TRUE;

  format_sybase_date(&tmpdate, data->O_ENTRY_DATE);

  data->status = TRAN_OK; /* Everything is ok */
#ifdef DEBUG
  fprintf(stderr, LINE "OrderStatus Output\n");
  ordstat_debug(data);
#endif

  return FALSE;
}

#define append_string(pos,x) { strcpy((pos), (x)); pos += sizeof(x) - 1; }
#define append_int(pos,size,val) { format_int((pos), (size), (val)); pos += size-1; }

int delivery_body (Delivery_data *data) {
  DBTINYINT             d_id = 1;
  int                   o_id;
  int                   sys_date;
  static int  delivery_count = 1;
  char buffer[1024];
  char *pos;

#ifdef DEBUG
  fprintf(stderr, BLANKS LINE "Delivery Input\n");
  delivery_debug(data);
#endif

  pos = buffer;
  append_string(pos, "----Transaction ");
  append_int   (pos, 8, delivery_count++);
  append_string(pos, " started----\nqueued-time: ");
  append_int   (pos, 10, data->queued_time);
  append_string(pos, "\nstart-time: ");
  time(&sys_date);
  append_int   (pos, 10, sys_date);
  append_string(pos, "\nW_ID: ");
  append_int   (pos, 10, data->W_ID);
  append_string(pos, ", CARRIER_ID: ");
  append_int   (pos, 3, data->O_CARRIER_ID);
  append_string(pos, "\n");

  data->status = -1; /* Assume an error unless everything goes well */
#ifdef USE_DEADLOCK
  deadlock = 0;
#endif
  dbrpcinit(dbproc, "delivery", 0);
  dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &data->W_ID);
  dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &data->O_CARRIER_ID);
  dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &d_id);
  if (dbrpcsend(dbproc) != SUCCEED) return TRUE;
  if (dbsqlok(dbproc) != SUCCEED) return TRUE;

  for (; d_id <= 10; d_id++) {
                if (dbresults(dbproc) != SUCCEED) return TRUE;
                dbbind(dbproc, 1, INTBIND,      0, &o_id);
                if (dbnextrow(dbproc) != REG_ROW) return TRUE;

                if (dbcanquery(dbproc) != SUCCEED) return TRUE;
                if (dbhasretstat(dbproc) && dbretstatus(dbproc) != 0) return TRUE;

                if (o_id == 0) { /* No new orders */
                  append_string(pos, "D_ID ");
                  append_int(pos, 3, d_id);
                  append_string(pos, " has no new orders.\n");
                } else {
                  append_string(pos, "D_ID ");
                  append_int(pos, 3, d_id);
                  append_string(pos, ", O_ID ");
                  append_int(pos, 8, o_id);
                  append_string(pos, "\n");
                }
  }

  time(&sys_date);
  append_string(pos, "end-time: ");
  append_int(pos, 11, sys_date);
  append_string(pos, "\n");
```

```
   write(1, buffer, (int)(pos-buffer));

   data->status = TRAN_OK; /* Everything is ok */

#ifdef DEBUG
   fprintf(stderr, LINE "Delivery Output\n");
   delivery_debug(data);
#endif
   return FALSE;
}

int stocklev_body (StockLevel_data *data) {

#ifdef DEBUG
   fprintf(stderr, BLANKS LINE "StockLevel Input\n");
   stocklev_debug(data);
#endif
   data->status = -1; /* Assume an error unless everything goes well */

   dbrpcinit(dbproc, "stock_level", 0);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &data->W_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, &data->D_ID);
   dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, &data->threshold);
   if (dbrpcsend(dbproc) != SUCCEED) return TRUE;
   if (dbsqlok(dbproc)   != SUCCEED) return TRUE;

   if (dbresults(dbproc)  != SUCCEED EXTRA_DEADLOCK_CHECK) return TRUE;
   dbbind(dbproc, 1, INTBIND,  0, &data->low_stock);
   if (dbnextrow(dbproc)  != REG_ROW EXTRA_DEADLOCK_CHECK) return TRUE;
   if (dbcanquery(dbproc) != SUCCEED EXTRA_DEADLOCK_CHECK) return TRUE;

   data->status = TRAN_OK; /* Everything is ok */

#ifdef DEBUG
   fprintf(stderr, LINE "StockLevel Output\n");
   stocklev_debug(data);
#endif
   return FALSE;
}

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

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

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

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

   login = dblogin();
   DBSETLUSER   (login, "sa");
   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");
   return 0;
}

int neword_debug(NewOrder_data *data) {
   int i;
   fprintf(stderr,
                "DBSMALLINT                W_ID = %d;\n"
                "DBTINYINT   D_ID = %d;\n"
                "DBINT          C_ID = %d;\n"
                "char           C_LAST[17] = '%s';\n"
                "char           C_CREDIT[3] = '%s';\n"
                "DBREAL      C_DISCOUNT = %f;\n"
                "DBTINYINT   O_OL_CNT = %d;\n"
                "DBINT          O_ID = %d;\n"
                "char           O_ENTRY_DATE[20] = '%s';\n"
                "char           statusline[25] = '%s';\n"
                "DBFLT8         total_amount = %f;\n"
                "DBTINYINT all_local = %d;\n"
                "short          status = %d;\n"
                "DBREAL      W_TAX = %f;\n"
                "DBREAL      D_TAX = %f;\n"
                "struct itemstruct {\n"
                ,
                data->W_ID,
                data->D_ID,
                data->C_ID,
                data->C_LAST,
                data->C_CREDIT,
                data->C_DISCOUNT,
                data->O_OL_CNT,
                data->O_ID,
                data->O_ENTRY_DATE,
                data->statusline,
                data->total_amount,
                data->all_local,
                data->status,
                data->W_TAX,
                data->D_TAX
                );
   for (i = 0; i < data->O_OL_CNT; i++) {
                if (i != 0)
                   fprintf(stderr, "\n");
                fprintf(stderr,
                   "DBSMALLINT   OL_SUPPLY_W_ID = %d;\n"
                   "DBINT        OL_I_ID = %d;\n"
                   "char         I_NAME[25] = '%s';\n"
                   "DBTINYINT    OL_QUANTITY = %d;\n"
                   "DBSMALLINT   QUANTITY = %d;\n"
                   "char         brand_generic[2] = '%s';\n"
                   "DBFLT8       I_PRICE = %f;\n"
                   "DBFLT8       OL_AMOUNT = %f;\n"
                   ,
                   data->item[i].OL_SUPPLY_W_ID,
                   data->item[i].OL_I_ID,
                   data->item[i].I_NAME,
                   data->item[i].OL_QUANTITY,
                   data->item[i].QUANTITY,
                   data->item[i].brand_generic,
                   data->item[i].I_PRICE,
                   data->item[i].OL_AMOUNT
                );
   }
   fprintf(stderr, "} item[15];\n");
   return 0;
}

int payment_debug(Payment_data *data) {
fprintf(stderr,
   "DBSMALLINT      W_ID = %d;\n"
   "DBTINYINT       D_ID = %d;\n"
   "DBINT           C_ID = %d;\n"
   "DBTINYINT       C_D_ID = %d;\n"
   "DBSMALLINT      C_W_ID = %d;\n"
   "short           status = %d;\n"
   "DBFLT8       H_AMOUNT = %f;\n"
   "char    H_DATE[20] = '%s';\n"
   "char    W_STREET_1[21] = '%s';\n"
   "char    W_STREET_2[21] = '%s';\n"
   "char    W_CITY[21] = '%s';\n"
   "char    W_STATE[3] = '%s';\n"
   "char    W_ZIP[10] = '%s';\n"
   "char    D_STREET_1[21] = '%s';\n"
   "char    D_STREET_2[21] = '%s';\n"
   "char    D_CITY[21] = '%s';\n"
   "char    D_STATE[3] = '%s';\n"
   "char    D_ZIP[10] = '%s';\n"
   "char    C_FIRST[17] = '%s';\n"
   "char    C_MIDDLE[3] = '%s';\n"
   "char    C_LAST[17] = '%s';\n"
   "char    C_STREET_1[21] = '%s';\n"
   "char    C_STREET_2[21] = '%s';\n"
   "char    C_CITY[21] = '%s';\n"
   "char    C_STATE[3] = '%s';\n"
   "char    C_ZIP[10] = '%s';\n"
   "char    C_PHONE[17] = '%s';\n"
   "char    C_SINCE[20] = '%s';\n"
   "char    C_CREDIT[3] = '%s';\n"
   "DBFLT8  C_CREDIT_LIM = %f;\n"
   "DBREAL  C_DISCOUNT = %f;\n"
   "DBFLT8  C_BALANCE = %f;\n"
   "char    C_DATA[200] = '%s';\n"
   ,
   data->W_ID,
   data->D_ID,
   data->C_ID,
   data->C_D_ID,
   data->C_W_ID,
   data->status,
   data->H_AMOUNT,
   data->H_DATE,
   data->W_STREET_1,
   data->W_STREET_2,
   data->W_CITY,
   data->W_STATE,
   data->W_ZIP,
```

```
    data->D_STREET_1,
    data->D_STREET_2,
    data->D_CITY,
    data->D_STATE,
    data->D_ZIP,
    data->C_FIRST,
    data->C_MIDDLE,
    data->C_LAST,
    data->C_STREET_1,
    data->C_STREET_2,
    data->C_CITY,
    data->C_STATE,
    data->C_ZIP,
    data->C_PHONE,
    data->C_SINCE,
    data->C_CREDIT,
    data->C_CREDIT_LIM,
    data->C_DISCOUNT,
    data->C_BALANCE,
    data->C_DATA
    );

    return 0;
}

int ordstat_debug(OrderStatus_data *data) {
    int i;
    fprintf(stderr,
    "DBSMALLINT     W_ID = %d;\n"
    "DBTINYINT      D_ID = %d;\n"
    "DBINT          C_ID = %d;\n"
    "char    C_FIRST[17] = '%s';\n"
    "char    C_MIDDLE[3] = '%s';\n"
    "char    C_LAST[17] = '%s';\n"
    "DBFLT8  C_BALANCE = %f;\n"
    "DBINT   O_ID = %d;\n"
    "char    O_ENTRY_DATE[20] = '%s';\n"
    "DBSMALLINT     O_CARRIER_ID = %d;\n"
    "short   ol_cnt = %d;\n"
    "short          status = %d;\n"
    ,
    data->W_ID,
    data->D_ID,
    data->C_ID,
    data->C_FIRST,
    data->C_MIDDLE,
    data->C_LAST,
    data->C_BALANCE,
    data->O_ID,
    data->O_ENTRY_DATE,
    data->O_CARRIER_ID,
    data->ol_cnt,
    data->status
    );
    for (i = 0; i < data->ol_cnt; i++) {
                if (i != 0)
                    fprintf(stderr, "\n");
                fprintf(stderr,
                "DBSMALLINT     OL_SUPPLY_W_ID = %d;\n"
                "DBINT          OL_I_ID = %d;\n"
                "DBTINYINT      OL_QUANTITY = %d;\n"
                "DBFLT8         OL_AMOUNT = %f;\n"
                "char           OL_DELIVERY_DATE[20] = '%s';\n"
                ,
                data->item[i].OL_SUPPLY_W_ID,
                data->item[i].OL_I_ID,
                data->item[i].OL_QUANTITY,
                data->item[i].OL_AMOUNT,
                data->item[i].OL_DELIVERY_DATE
                );
    }
};

int delivery_debug(Delivery_data *data) {
    fprintf(stderr,
    "DBSMALLINT     W_ID = %d;\n"
    "DBSMALLINT     O_CARRIER_ID = %d;\n"
    "int            queued_time = %d;\n"
    "short          status = %d;\n"
    "char    exec_status[50] = '%s';\n"
    ,
    data->W_ID,
    data->O_CARRIER_ID,
    data->queued_time,
    data->status,
    data->exec_status
    );
}

int stocklev_debug(StockLevel_data *data) {
    fprintf(stderr,
    "DBSMALLINT     W_ID = %d;\n"
    "DBTINYINT      D_ID = %d;\n"
    "DBSMALLINT     threshold = %d;\n"
    "DBINT          low_stock = %d;\n"
    "short          status = %d;\n"
    ,
    data->W_ID,
    data->D_ID,
    data->threshold,
    data->low_stock,
    data->status
    );
};

#ifdef NOTUX
int main (int argc, char **argv) {
    NewOrder_data no;
    OrderStatus_data os;
    tpsvrinit(argc, argv);

    memset(&no, 0, sizeof(no));
    no.all_local=1;
    no.W_ID=1;
    no.D_ID=1;
    no.C_ID=1;
    no.O_OL_CNT=1;
    no.item[0].OL_I_ID=999999;
    no.item[0].OL_SUPPLY_W_ID=1;
    no.item[0].OL_QUANTITY=1;

    os.W_ID=1;
    os.D_ID=1;
    os.C_ID=1;

    if (neword_body (&no))
                printf ("error\n");
    if (ordstat_body (&os))
                printf ("error\n");
    return 0;
}
#endif

/*****************************************************************************/
/* fmt_date.c                                           03/31/97 */
/*****************************************************************************/
#include <stdio.h>
#include <time.h>

typedef struct {
    unsigned int dtdays;
    unsigned int dttime;
} DATE;


#define Feb29 (31+29-1)
#define YEAR(yr) ( ((yr)-1900)*365 + ((yr)-1900-1)/4 )
void format_sybase_date(DATE *date, char *output ) {
    static int dur[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int quad, year, month, day;
    int cent;
    int hour, minute, sec;
    char *ptr;

    day = date->dtdays;
    sec = date->dttime/300;

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

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

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

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

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

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

    month += 1;
    day   += 1;
    year  += quad*4 + 1900;
    cent   = year/100;
    year  %= 100;
```

```
  ptr = output;
  *(ptr++) = (month >= 10)?month/10+'0':'0';
  *(ptr++) = month % 10 + '0';
  *(ptr++) = '-';
  *(ptr++) = (day >= 10)?day/10+'0':'0';
  *(ptr++) = day % 10 + '0';
  *(ptr++) = '-';
  *(ptr++) = (cent >= 10)?cent/10+'0':'0';
  *(ptr++) = cent % 10 + '0';
  *(ptr++) = (year >= 10)?year/10+'0':'0';
  *(ptr++) = year % 10 + '0';
  *(ptr++) = ' ';
  *(ptr++) = (hour >= 10)?hour/10+'0':'0';
  *(ptr++) = hour % 10 + '0';
  *(ptr++) = ':';
  *(ptr++) = (minute >= 10)?minute/10+'0':'0';
  *(ptr++) = minute % 10 + '0';
  *(ptr++) = ':';
  *(ptr++) = (sec >= 10)?sec/10+'0':'0';
  *(ptr++) = sec % 10 + '0';
  *(ptr++) = '\0';
}
```

A.3 Database Server Code

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

# Stored procedure for TPC-C 3.2 on ADAPTIVE SERVER ENTERPRISE 11.5 and later
# Copyright Sybase 1997
#
isql -Usa -P$PASSWORD <<EOF
use tpcc
go
if exists ( SELECT name FROM sysobjects WHERE name = 'neworder_local')
        DROP PROC neworder_local
go

CREATE PROC neworder_local (
        @w_id                   smallint,
        @d_id                   tinyint,
        @c_id                   int,
        @o_ol_cnt       tinyint,

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

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

        @i_price        real,
        @i_name                 char(24),       @i_data                 char(50),
        @s_quantity     smallint,
        @s_ytd                  int,                            @s_order_cnt int,
        @s_dist                 char(24),       @s_data                 char(50),

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

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

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

begin

        begin transaction NO

        OPEN c_no_wdc
        FETCH c_no_wdc INTO
                @w_tax, @d_tax, @o_id,
                @c_last, @c_discount, @c_credit
                ,@0, @1, @one, @ten, @commit_flag, @ol_number
                ,@o_entry_d

        /*
    ** Update will fail if matching row for warehouse, distict,
    ** customer not found. TPC-C V3.3 compliant.
    */

        UPDATE          district
                SET             d_next_o_id = @o_id + 1
                WHERE CURRENT OF c_no_wdc
        CLOSE c_no_wdc

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

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

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

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

                if (@@rowcount = 0)
                  begin
                    select @commit_flag = 0
```

```
                        select NULL, NULL, NULL, NULL
                        continue
                      end
    /*Otherwise if the item is found */
                      update stock
                        set s_ytd = s_ytd + @ol_qty,
                                @s_quantity = s_quantity - @ol_qty +
                            case when (s_quantity - @ol_qty < 10)
                            then 91 else 0 end,
                          s_quantity = s_quantity - @ol_qty +
                            case when (s_quantity - @ol_qty < 10)
                            then 91 else 0 end,
                                s_order_cnt = s_order_cnt + 1,
                          @s_data = s_data,
                          @s_dist = case @d_id
                            when 1 then s_dist_01
                            when 2 then s_dist_02
                            when 3 then s_dist_03
                            when 4 then s_dist_04
                            when 5 then s_dist_05
                            when 6 then s_dist_06
                            when 7 then s_dist_07
                            when 8 then s_dist_08
                            when 9 then s_dist_09
                            when 10 then s_dist_10
                            end
                              where s_w_id = @w_id and
                                s_i_id = @i_id
                    if (@@rowcount = 0)
                      begin
                        select @commit_flag = 0
                        select NULL, NULL, NULL, NULL
                        continue
                          end
                     /*Otherwise if the Stock is found */
/* Loader used Jan 01 1800 as NULL date */
                /*
                ** Insert of NULL value will cause insert to fail.
                ** TPC-C V3.3 compliant.
                */
                 INSERT INTO order_line (
                    ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                    ol_supply_w_id, ol_delivery_d, ol_quantity,
                    ol_amount, ol_dist_info)
                  VALUES (
                    @o_id, @d_id, @w_id, @ol_number, @i_id,
                    @w_id, "18000101", @ol_qty,
                    @ol_qty * @i_price, @s_dist)


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

              end /* while */

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

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

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

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

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

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

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

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

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

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

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

begin

              begin transaction NO

              OPEN c_no_wdc
              FETCH c_no_wdc INTO
                              @w_tax, @d_tax, @o_id,
                              @c_last, @c_discount, @c_credit
                              ,@0, @1, @one, @ten, @commit_flag, @ol_number
                              ,@o_entry_d

    /*
    ** Update will fail if matching row for warehouse, distict,
    ** customer not found. TPC-C V3.3 compliant.
    */

              UPDATE          district
                              SET             d_next_o_id = @o_id + 1
                              WHERE CURRENT OF c_no_wdc
              CLOSE c_no_wdc

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

```
                when 14 then @ol_qty15
                else @ol_qty
                end
                        ,@s_w_id = case @ol_number
            when 1 then @s_w_id2
                when 2 then @s_w_id3
                when 3 then @s_w_id4
                when 4 then @s_w_id5
                when 5 then @s_w_id6
                when 6 then @s_w_id7
                when 7 then @s_w_id8
                when 8 then @s_w_id9
                when 9 then @s_w_id10
                when 10 then @s_w_id11
                when 11 then @s_w_id12
                when 12 then @s_w_id13
                when 13 then @s_w_id14
                when 14 then @s_w_id15
                else @s_w_id
                end

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

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

        if (@@rowcount = 0)
          begin
            select @commit_flag = 0
            select NULL, NULL, NULL, NULL
            continue
          end
/* Loader used Jan 01 1800 as NULL date */
                /*
                ** Insert of NULL value causes insert to fail.
                ** TPC-C V3.3 compliant
                */

                 INSERT INTO order_line (
                    ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                    ol_supply_w_id, ol_delivery_d, ol_quantity,
                    ol_amount, ol_dist_info)
                 VALUES (
                    @o_id, @d_id, @w_id, @ol_number, @i_id,
                    @w_id, "18000101", @ol_qty,
                    @ol_qty * @i_price, @s_dist)

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

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

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

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

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

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

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

declare @screen_data        char(200),      @today      datetime

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

BEGIN TRANSACTION PID
            OPEN c_pay_wd
            FETCH c_pay_wd INTO
                        @w_id_retrieved,@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 will fail if w_id_retrieved is invalid.
            ** TPC-C V3.3 complaint.
            **/

            UPDATE district
                        SET     d_ytd  = @d_ytd + @h_amount
                        WHERE CURRENT OF c_pay_wd
            UPDATE warehouse
                        SET w_ytd = @w_ytd + @h_amount
                        WHERE CURRENT OF c_pay_wd
            CLOSE c_pay_wd

            /* Customer data */
            UPDATE customer SET
                         @c_first = c_first
                        , @c_middle = c_middle
                        , @c_last = c_last
                        , @c_street_1 = c_street_1
                        , @c_street_2 = c_street_2
                        , @c_city = c_city
                        , @c_state = c_state
                        , @c_zip = c_zip
                        , @c_phone = c_phone
                        , @c_credit = c_credit
                        , @c_credit_lim = c_credit_lim
                        , @c_discount = c_discount
                        , c_balance = c_balance - @h_amount
                        , @c_balance = c_balance - @h_amount
```

```
                        , c_ytd_payment = c_ytd_payment + @h_amount
                        , c_payment_cnt = c_payment_cnt + 1
                        , @c_since = c_since
                        , @data1 = c_data1
                        , @data2 = c_data2
                        , @today = getdate()
                where
                        c_id = @c_id
                        and c_w_id = @c_w_id
                        and c_d_id = @c_d_id


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

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

                /* Create the history record */

                /**
                ** Insert will fail if w_id is invalid.
                ** TPC-C V 3.3 compliant
                **/

                INSERT INTO history (
                        h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                        h_date, h_amount, h_data)
                VALUES (
                        @c_id, @c_d_id, @c_w_id, @d_id, @w_id_retrieved,
                        @today, @h_amount, (@w_name + "    " + @d_name))

COMMIT TRANSACTION PID

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

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

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

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

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

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

declare @screen_data            char(200),      @today          datetime

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

declare c_find CURSOR FOR
                SELECT  c_id
                FROM customer (index c_non1 prefetch 2 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

                OPEN c_pay_wd
                FETCH c_pay_wd INTO
                                @w_id_retrieved, @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 will fail if w_id_retrieved is invalid.
        ** TPC-C V3.3 complaint.
        **/

                UPDATE district
                                SET     d_ytd  = @d_ytd + @h_amount
                                WHERE CURRENT OF c_pay_wd
                UPDATE warehouse
                                SET w_ytd = @w_ytd + @h_amount
                                WHERE CURRENT OF c_pay_wd
                CLOSE c_pay_wd

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

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

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

                /* Create the history record */
                /**
                **  Insert will fail if w_id is invalid.
                **  TPC-C V3.3 compliant.
                **/

                INSERT INTO history (
                            h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                            h_date, h_amount, h_data)
                VALUES (
                            @c_id, @c_d_id, @c_w_id, @d_id, @w_id_retrieved,
                            @today, @h_amount, (@w_name + "    " + @d_name))

COMMIT TRANSACTION PNM

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

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

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

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

BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
                SELECT      @o_id = o_id, @o_carrier_id = o_carrier_id,
                            @o_entry_d = o_entry_d
                FROM        orders (index o_clu prefetch 16 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
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
                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 OSNM
go
if exists (select * from sysobjects where name = 'delivery')
                drop proc delivery
go
CREATE PROC delivery
                @w_id                       smallint,
                @o_carrier_id smallint,
```

```
                @d_id                   tinyint = 1
as

declare         @no_o_id    int,                    @o_c_id             smallint,
                @ol_total   float,                  @ol_amount  float,
                @junk_id    smallint,
                @ten                    tinyint

declare c_del_no CURSOR FOR
                SELECT      no_o_id
                FROM        new_order (index no_clu) HOLDLOCK
                WHERE       no_d_id = @d_id
                AND         no_w_id = @w_id
                FOR UPDATE
                /*
                ** The only purpose of the index hint in the above is to ensure
                ** that the clustered index is used.  As it turns out, our optimizer
                ** chooses the clustered index anyway -- with or without the hint.
                */

begin
   select @ten = 10

   while (@d_id <= @ten) begin

                BEGIN TRANSACTION DEL
                OPEN c_del_no
                FETCH c_del_no INTO @no_o_id

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

                DELETE      FROM new_order
                WHERE       CURRENT OF c_del_no
                CLOSE       c_del_no

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

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

                COMMIT TRANSACTION DEL

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

CREATE PROC stock_level
                @w_id       smallint,
                @d_id       tinyint,
                @threshold smallint
as

                select  count(distinct(s_i_id))
                FROM        district,
                            order_line (index ol_clu prefetch 2 mru),
                            stock (index s_clu prefetch 2 lru)
                WHERE       d_w_id          =           @w_id
                AND         d_id            =           @d_id
                AND         ol_w_id         =           @w_id
                AND         ol_d_id         =           @d_id
                AND         ol_o_id         between (d_next_o_id - 20) and (d_next_o_id - 1)
                AND         s_w_id          =           ol_w_id
                AND         s_i_id          =           ol_i_id
                AND     s_quantity < @threshold

go
EOF
```

Appendix B: Tunable Parameters

B.1 Database Parameters

DBMS Parameters and Tunables

```
#*******************************************************************************
#
#                       Configuration File for the Sybase Adaptive Server Enterprise
#
#                       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 = 2
        memory alignment boundary = DEFAULT
        global async prefetch limit = 0

[Named Cache:c_customer]
        cache size = 16M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement

[2K I/O Buffer Pool]
        pool size = 16M
        wash size = 6144 K
        local async prefetch limit = 0

[Named Cache:c_customer_index]
        cache size = 180M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement

[2K I/O Buffer Pool]
        pool size = 180M
        wash size = 512 K
        local async prefetch limit = 0

[Named Cache:c_log]
        cache size = 24M
        cache status = log only
        cache replacement policy = relaxed LRU replacement

[2K I/O Buffer Pool]
        pool size = 2M
        wash size = 1024 K
        local async prefetch limit = 0

[4K I/O Buffer Pool]
        pool size = 22M
        wash size = 1024 K
        local async prefetch limit = 0

[Named Cache:c_no_ol]
        cache size = 162M
        cache status = mixed cache
        cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
        pool size = 162M
        wash size = 8192 K
        local async prefetch limit = 0

[Named Cache:c_ol_index]
        cache size = 47M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
        pool size = 47M
        wash size = 4096 K
        local async prefetch limit = 0

[Named Cache:c_orders]
        cache size = 130M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
        pool size = 118M
        wash size = 2048 K
        local async prefetch limit = 0

[16K I/O Buffer Pool]
        pool size = 12M
        wash size = 1024 K
        local async prefetch limit = 0

[Named Cache:c_stock]
        cache size = 1800M
        cache status = mixed cache
        cache replacement policy = DEFAULT

[2K I/O Buffer Pool]
        pool size = 1800M
        wash size = 40960 K
        local async prefetch limit = 0

[Named Cache:c_stock_index]
        cache size = 88M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement

[2K I/O Buffer Pool]
        pool size = 88M
        wash size = 512 K
        local async prefetch limit = 0

[Named Cache:c_sysindexes]
        cache size = 512 K
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement

[2K I/O Buffer Pool]
        pool size = 512 K
        wash size = 256 K
        local async prefetch limit = 0

[Named Cache:c_tinyhot]
        cache size = 40.6M
        cache status = mixed cache
        cache status = HK ignore cache
        cache replacement policy = relaxed LRU replacement

[2K I/O Buffer Pool]
        pool size = 28.6M
        wash size = 512 K
        local async prefetch limit = 0

[16K I/O Buffer Pool]
        pool size = 12M
        wash size = 512 K
        local async prefetch limit = 0

[Named Cache:default data cache]
        cache size = 13M
        cache status = default data cache
        cache status = HK ignore cache
        cache replacement policy = relaxed LRU replacement

[2K I/O Buffer Pool]
        pool size = 9M
        wash size = 2048 K
        local async prefetch limit = 0

[16K I/O Buffer Pool]
        pool size = 4M
        wash size = 2048 K
        local async prefetch limit = 0

[Meta-Data Caches]
        number of open databases = DEFAULT
        number of open objects = DEFAULT
        open object spinlock ratio = DEFAULT
        number of open indexes = DEFAULT
        open index hash spinlock ratio = DEFAULT
        open index spinlock ratio = DEFAULT

[Disk I/O]
        allow sql server async i/o = DEFAULT
        disk i/o structures = 8192
        page utilization percent = DEFAULT
        number of devices = 135
        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

[Parallel Query]
        number of worker processes = DEFAULT
        memory per worker process = DEFAULT
        max parallel degree = DEFAULT
        max scan parallel degree = DEFAULT

[Physical Resources]

[Physical Memory]
        total memory = 1441792
        additional network memory = 1474560
        lock shared memory = DEFAULT
        shared memory starting address = DEFAULT
        max SQL text monitored = DEFAULT

[Processors]
        max online engines = 4
        min online engines = DEFAULT

[SQL Server Administration]
        default database size = DEFAULT
        identity burning set factor = DEFAULT
        allow nested triggers = DEFAULT
        allow updates to system tables = 1
        print deadlock information = DEFAULT
        default fill factor percent = DEFAULT
        number of mailboxes = DEFAULT
        number of messages = DEFAULT
        number of alarms = DEFAULT
        number of pre-allocated extents = DEFAULT
        event buffers per engine = DEFAULT
        cpu accounting flush interval = DEFAULT
        i/o accounting flush interval = DEFAULT
        sql server clock tick length = DEFAULT
        runnable process search count = 2000
        i/o polling process count = DEFAULT
        time slice = DEFAULT
        deadlock retries = DEFAULT
        cpu grace time = DEFAULT
        number of sort buffers = DEFAULT
        size of auto identity column = DEFAULT
        identity grab size = DEFAULT
        lock promotion HWM = DEFAULT
        lock promotion LWM = DEFAULT
        lock promotion PCT = DEFAULT
        housekeeper free write percent = 0
        partition groups = DEFAULT
        partition spinlock ratio = DEFAULT
        allow resource limits = DEFAULT
        number of aux scan descriptors = DEFAULT
        SQL Perfmon Integration = DEFAULT
        allow backward scans = DEFAULT

[User Environment]
        number of user connections = 120
        stack size = 40960
        stack guard size = DEFAULT
        permission cache entries = DEFAULT
        user log cache size = 4096
        user log cache spinlock ratio = DEFAULT

[Lock Manager]
        number of locks = 10000
        deadlock checking period = DEFAULT
        freelock transfer block size = DEFAULT
        max engine freelocks = 25
        address lock spinlock ratio = DEFAULT
        page lock spinlock ratio = DEFAULT
        table lock spinlock ratio = 1

[Security Related]
        systemwide password expiration = DEFAULT
        audit queue size = DEFAULT
        evaluated configuration = DEFAULT
        curread change w/ open cursors = DEFAULT
        allow procedure grouping = DEFAULT
        select on syscomments.text = DEFAULT
        auditing = DEFAULT
        current audit table = DEFAULT
        suspend audit when device full = DEFAULT
        max roles enabled per user = DEFAULT
        unified login required = DEFAULT
        use security services = DEFAULT
        msg confidentiality reqd = DEFAULT
        msg integrity reqd = DEFAULT
        msg replay detection reqd = DEFAULT
        msg origin checks reqd = DEFAULT
        msg out-of-seq checks reqd = DEFAULT
        secure default login = DEFAULT
        map to DOLLAR SIGN in login = DEFAULT
        map to POUND SIGN in login = DEFAULT
        map to UNDERSCORE in login = DEFAULT

[Extended Stored Procedure]
        esp unload dll = DEFAULT
        esp execution priority = DEFAULT
        esp execution stacksize = DEFAULT
        xp_cmdshell context = DEFAULT
        start mail session = DEFAULT

[Error Log]
        event logging = DEFAULT
        log audit logon success = DEFAULT
        log audit logon failure = DEFAULT
        event log computer name = DEFAULT

[Rep Agent Thread Administration]
        enable rep agent threads = DEFAULT

[Omni Administration]
        enable omni services = DEFAULT
        omni connect timeout = DEFAULT
        omni bulk insert batch size = DEFAULT
        max omni remote connections = DEFAULT
        max omni remote servers = DEFAULT
        omni packet size = DEFAULT
        omni cursor rows = DEFAULT
        omni idle timeout = DEFAULT
        omni rpc handling = DEFAULT
```

B.2 Transaction Monitor Parameters

```
        TUXEDO CONFIGURATION

*RESOURCES
IPCKEY 44003
UID 10001
GID 1
MASTER client1
PERM 0660
MODEL SHM
LDBAL Y
MAXACCESSERS 5000
MAXSERVERS 1000
MAXSERVICES 1000
SCANUNIT 60
SANITYSCAN 5
BLOCKTIME 3
BBLQUERY 60

*MACHINES
client1 LMID=client1
 TUXDIR="/home/tuxedo"
 APPDIR="/home/sybase/tpcc_cstux_syb_01/app"
 TUXCONFIG="/home/sybase/install.tux/tuxconfig"
 ULOGPFX="/home/sybase/install.tux/ULOG"

*GROUPS
DEFAULT: LMID=client1
base   GRPNO=1
svr1   GRPNO=11
svr2   GRPNO=12
svr3   GRPNO=13
svr4   GRPNO=14
svr5   GRPNO=15

*SERVERS

DEFAULT:    SRVGRP=base
DEFAULT:    REPLYQ=Y
DEFAULT:    MAXGEN=2
DEFAULT:    RESTART=Y
DEFAULT:    CLOPT="-A"
#
sybase_service SRVID=900 MIN=25 RQADDR="900"
#
sybase_service SRVID=110 MIN=5 RQADDR="110" SRVGRP=svr1
sybase_service SRVID=120 MIN=5 RQADDR="120" SRVGRP=svr2

*SERVICES
neword_sql
payment_sql
stocklev_sql
ordstat_sql
delivery_sql
```

B.3 AIX Parameters

```
        OS PARAMETERS

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    40000   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     3145728 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
```

Appendix C: Database Setup

C.1 Sybase Table Space Definition

```
# db_scale = 700

DEVICE      master          /dev/rlvsybmstr 50
            db=tpcc size=50
            segment=default
            segment=system
DEVICE_END

DEVICE  tpcc_log4           /dev/rlvsyblog4                 1830
            db=tpcc size=1830 log
DEVICE_END

DEVICE  tpcc_log3           /dev/rlvsyblog3                 1830
            db=tpcc size=1830 log
DEVICE_END

DEVICE  tpcc_log2           /dev/rlvsyblog2                 1830
            db=tpcc size=1830 log
DEVICE_END

DEVICE  tpcc_log1           /dev/rlvsyblog1                 1830
            db=tpcc size=1830 log
DEVICE_END

# Stripe these devices over multiple disks with 16K chunks
#
DEVICE      w_d_i_no        /dev/rlvwdino   115
            db=tpcc         size=115
            segment=Scache
#           segment=Swarehouse segment=Sdistrict segment=Sitem segment=Snew_order
DEVICE_END

# Space requirement is 0.9 MB/warehouse + growth
DEVICE  orders1     /dev/rlvorders1    50
            db=tpcc size=50
            segment=Sorders
DEVICE_END

DEVICE  orders2     /dev/rlvorders2    50
            db=tpcc size=50
            segment=Sorders
DEVICE_END

DEVICE  orders3     /dev/rlvorders3    50
            db=tpcc size=50
            segment=Sorders
DEVICE_END

DEVICE  orders4     /dev/rlvorders4    50
            db=tpcc size=50
            segment=Sorders
DEVICE_END

DEVICE  orders5     /dev/rlvorders5    50
            db=tpcc size=50
            segment=Sorders
DEVICE_END

DEVICE  orders6     /dev/rlvorders6    50
            db=tpcc size=50
            segment=Sorders
DEVICE_END

DEVICE  orders7     /dev/rlvorders7    50
            db=tpcc size=50
            segment=Sorders
DEVICE_END

DEVICE  orders8     /dev/rlvorders8    50
            db=tpcc size=50
            segment=Sorders
DEVICE_END

DEVICE  orders9     /dev/rlvorders9    50
            db=tpcc size=50
            segment=Sorders
DEVICE_END

DEVICE  orders10    /dev/rlvorders10   131
            db=tpcc size=131
            segment=Sorders
DEVICE_END

DEVICE  orders11    /dev/rlvorders11   95
            db=tpcc size=95
            segment=Sorders
DEVICE_END

# Non clustered index for customer
# Space requirement is 1.5 MB/warehouse

DEVICE  c_idx1      /dev/rlvcust_indx1  338
      db=tpcc size=338
      segment=Scustomer_index
DEVICE_END

DEVICE  c_idx2      /dev/rlvcust_indx2  338
      db=tpcc size=338
      segment=Scustomer_index
DEVICE_END

DEVICE  c_idx3      /dev/rlvcust_indx3  338
      db=tpcc size=338
      segment=Scustomer_index
DEVICE_END

DEVICE  c_idx4      /dev/rlvcust_indx4  280
      db=tpcc size=280
      segment=Scustomer_index
DEVICE_END

# Order_line Table, Space requirement is 18.9 MB/warehouse
# + growth.

DEVICE  order_line1 /dev/rlvorder_line1    620
            db=tpcc size=620
            segment=Sorder_line
DEVICE_END

DEVICE  order_line2 /dev/rlvorder_line2    620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line3 /dev/rlvorder_line3    620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line4 /dev/rlvorder_line4    620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line5 /dev/rlvorder_line5    620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line6 /dev/rlvorder_line6    620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line7 /dev/rlvorder_line7    620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line8 /dev/rlvorder_line8    620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line9 /dev/rlvorder_line9    620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line10 /dev/rlvorder_line10  620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line11 /dev/rlvorder_line11  620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line12 /dev/rlvorder_line12  620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line13 /dev/rlvorder_line13  620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line14 /dev/rlvorder_line14  620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line15 /dev/rlvorder_line15  620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END
```

```
DEVICE  order_line16  /dev/rlvorder_line16   620
      db=tpcc size=620
      segment=Sorder_line
DEVICE_END

DEVICE  order_line17  /dev/rlvorder_line17   1530
      db=tpcc size=1530
      segment=Sorder_line
DEVICE_END

DEVICE  order_line18  /dev/rlvorder_line18   1530
      db=tpcc size=1530
      segment=Sorder_line
DEVICE_END

DEVICE  order_line19  /dev/rlvorder_line19   1115
      db=tpcc size=1115
      segment=Sorder_line
DEVICE_END

DEVICE  order_line18  /dev/rlvorder_line18   1115
      db=tpcc size=1115
      segment=Sorder_line
DEVICE_END

# customer table, space requirement is 20.5 MB/warehouse
# Use 1 disk for every 20 warehouse

DEVICE        customer1      /dev/rlvcust1   606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE        customer2      /dev/rlvcust2   606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE        customer3      /dev/rlvcust3   606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer4      /dev/rlvcust4          606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer5      /dev/rlvcust5   606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer6      /dev/rlvcust6   606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer7      /dev/rlvcust7   606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer8      /dev/rlvcust8   606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer9      /dev/rlvcust9   606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer10     /dev/rlvcust10  606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer11     /dev/rlvcust11  606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer12     /dev/rlvcust12  606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer13     /dev/rlvcust13  606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer14     /dev/rlvcust14  606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer15     /dev/rlvcust15  606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer16     /dev/rlvcust16  606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer17     /dev/rlvcust17  606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer18     /dev/rlvcust18  606 db=tpcc size=606 segment=Scustomer DEVICE_END
DEVICE  customer19     /dev/rlvcust19  1509 db=tpcc size=1509 segment=Scustomer DEVICE_END
DEVICE  customer20     /dev/rlvcust20  1509 db=tpcc size=1509 segment=Scustomer DEVICE_END
DEVICE  customer21     /dev/rlvcust21  639 db=tpcc size=639 segment=Scustomer DEVICE_END

# Stock Table, space requirement is 34.24 MB/warehouse
# Use 1 disk for every 10 warehouse

DEVICE  stock1      /dev/rlvstock1   510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock2      /dev/rlvstock2   510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock3      /dev/rlvstock3   510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock4      /dev/rlvstock4   510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock5      /dev/rlvstock5   510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock6      /dev/rlvstock6   510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock7      /dev/rlvstock7   510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock8      /dev/rlvstock8   510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock9      /dev/rlvstock9   510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock10     /dev/rlvstock10  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock11     /dev/rlvstock11  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock12     /dev/rlvstock12  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock13     /dev/rlvstock13  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock14     /dev/rlvstock14  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock15     /dev/rlvstock15  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock16     /dev/rlvstock16  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock17     /dev/rlvstock17  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock18     /dev/rlvstock18  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock19     /dev/rlvstock19  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock20     /dev/rlvstock20  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock21     /dev/rlvstock21  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock22     /dev/rlvstock22  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock23     /dev/rlvstock23  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock24     /dev/rlvstock24  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock25     /dev/rlvstock25  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock26     /dev/rlvstock26  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock27     /dev/rlvstock27  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock28     /dev/rlvstock28  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock29     /dev/rlvstock29  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock30     /dev/rlvstock30  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock31     /dev/rlvstock31  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock32     /dev/rlvstock32  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock33     /dev/rlvstock33  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock34     /dev/rlvstock34  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock35     /dev/rlvstock35  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock36     /dev/rlvstock36  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock37     /dev/rlvstock37  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock38     /dev/rlvstock38  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock39     /dev/rlvstock39  510 db=tpcc size=510 segment=Sstock DEVICE_END
DEVICE  stock40     /dev/rlvstock40  1750 db=tpcc size=1750 segment=Sstock DEVICE_END
DEVICE  stock41     /dev/rlvstock41  1444 db=tpcc size=1444 segment=Sstock DEVICE_END
DEVICE  stock42     /dev/rlvstock42  1444 db=tpcc size=1444 segment=Sstock DEVICE_END

DEVICE  history     /dev/rlvhist    1200
      db=tpcc size=1200
      segment=Shistory
DEVICE_END

DEVICE  history2     /dev/rlvhist2    200
      db=tpcc size=200
      segment=Shistory
DEVICE_END
```

C.2 Table Creation and Index Build

```
#!/bin/sh -f

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

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

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

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

if exists ( select name from sysobjects where name = 'history' )
        drop table history
go
create table history (
        h_c_id                  int,
        h_c_d_id        tinyint,
        h_c_w_id        smallint,
```

```
                h_d_id                  tinyint,
                h_w_id                  smallint,
                h_date                  datetime,
                h_amount        float,
                h_data                  char(24)
) on Shistory
go
alter table history partition 20
go

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

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

if exists ( select name from sysobjects where name = 'order_line' )
                drop table order_line
go
create table order_line (
                ol_o_id                 int,
                ol_d_id                 tinyint,
                ol_w_id                 smallint,
                ol_number       tinyint,
                ol_i_id                 int,
                ol_supply_w_id          smallint,
                ol_delivery_d   datetime,       /*- Updated by D */
                ol_quantity     smallint,
                ol_amount       float,
                ol_dist_info    char(24)
) on Sorder_line
go
create unique clustered index ol_clu
                on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
                on Sorder_line
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go

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

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

checkpoint
go
EOF


#!/bin/sh -f

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

create unique clustered index w_clu
                on warehouse(w_id)
                with fillfactor = 1
                on Scache
go

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

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

checkpoint
go
EOF
```

C.3 Data Generation Code

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

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

int load_item;
int load_warehouse;
int load_district;
int load_history;
int load_orders;
int load_new_order;
int load_order_line;
int load_customer;
int load_stock;
ID  w1, w2;
ID  warehouse;
int batch_size = 1000;
char   password[10];

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

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

        getargs(argn, argv);
        Randomize();

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


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

Warehouse

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


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


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

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

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

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

                warehouse_load();

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

NTVOID
begin_warehouse_load()
{
        int             i = 1;

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

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

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

NTVOID
end_warehouse_load()
{
        bulk_close(bulk_w);
}
```

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

District

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



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

int bulk_d;

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

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

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

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

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

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



NTVOID
begin_district_load()
{
        int             i = 1;

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

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



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

Item

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

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

int bulk_i;
NTVOID
LoadItems()
{
        int perm[MAXITEMS+1];
#ifndef _NTINTEL
        int i, r, t;
#endif

        printf("Loading items\n");

        begin_item_load();

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

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

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

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

         item_load();
         }

        end_item_load();
        return;
}



NTVOID
begin_item_load()
{
        int             i = 1;

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

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



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

NTVOID
end_item_load()
{
        bulk_close(bulk_i);
}



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

History

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



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

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

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

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

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

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

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


NTVOID
begin_history_load()
{
        int             i = 1;

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

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

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

NTVOID
end_history_load()
{
        bulk_close(bulk_h);
}



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

Customer

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

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

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

int bulk_c;

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

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

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

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

LastName(i<1000?i:NURandomNumber(255,NURAND_C,0,999),c_last);
                        MakeAlphaString(8, 16, c_first);
                        MakeAddress(c_street_1,c_street_2,c_city,c_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();
                }
        }
}
NTVOID
begin_customer_load()
{
        int             i = 1;

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

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


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

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

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



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

Order, Order line, New order

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

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

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

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


int  o_bulk;
int ol_bulk;
int no_bulk;


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

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

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

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

```
{
        ID w_id;
        ID d_id;

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

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

        RandomPermutation(cust, ORD_PER_DIST);


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

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

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

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

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

        }
}



NTVOID
OrderLine(ol)
        ID ol;
{

        ol_o_id = o_id;
        ol_d_id = o_d_id;
        ol_w_id = o_w_id;
        ol_number = ol;
        ol_i_id = RandomNumber(1, MAXITEMS);
        ol_supply_w_id = o_w_id;
    if (o_id <= 2100) ol_delivery_d = o_entry_d;
    else        { ol_delivery_d.x[0] = 0;
                  ol_delivery_d.x[1] = 0; }
        ol_quantity = 5;
        if (o_id <= 2100) ol_amount = 0;
        else                    ol_amount = RandomNumber(1, 999999);
        MakeAlphaString(24, 24, ol_dist_info);
        order_line_load();
}

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

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



NTVOID
begin_order_load()
{
        int             i = 1;

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

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



NTVOID
begin_order_line_load()
{
        int             i = 1;

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

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

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

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

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

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

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

Stock

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


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

int bulk_s;

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

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

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

        int w, i, j;

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

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

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

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

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



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

NTVOID
end_stock_load()
{
        bulk_close(bulk_s);
}


NTVOID
test(){}

NTVOID
getargs(argc, argv)

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

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


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

        }
```

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

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

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

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

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

}


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

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


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

        length = RandomNumber(min, max);

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

        return length;
}

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

        length = 4;
        for (i=0;  i<length;  i++)
        num[i] = digit[RandomNumber(0,9)];
        num[length] = '\0';

        return length;
}



int MakeAlphaString(min, max, str)
        int min;
        int max;
        TEXT str[];
{
        static char character[] =
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
        int length;
        int i;

        length = RandomNumber(min, max);

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

        return length;
}

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

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

        pos = RandomNumber(0,len-8);

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



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

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

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



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



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

```
}


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

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

                return r;
}

#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED

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


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

#define         NURAND_C 123

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

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

typedef struct timeval TIME;




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

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

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

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

#endif /* TPCC_INCLUDED */

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

Appendix D: Screen Layouts

D.1 New-Order Screen

```
(1)New-Order (2)Payment (3)Order-Status (4)Delivery (5)Stock-Level (9)Exit

                                         New Order
Warehouse: 9999   District: 99                        Date: DD-MM-YYY hhmm:ss
Customer:   9999   Name: XXXXXXXXXXXXXXXX     Credit: XX   %Disc: 99.99
Order Number: 99999999  Number of Lines: 99         W_tax: 99.99  D_tax: 99.99

 Supp_W  Item_Id  Item Name                              Qty  Stock  B/G Price       Amount
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
  9999     999999   XXXXXXXXXXXXXXXXXXXXXXXX  99    999    X   $999.99  $9999.99
Execution Status: XXXXXXXXXXXXXXXXXXXXXXXX                              Total:  $99999.99
```

D.2 Payment Screen

```
(1)New-Order (2)Payment (3)Order-Status (4)Delivery (5)Stock-Level (9)Exit

                                          Payment
Date: DD-MM-YYYY hh:mm:ss

Warehouse: 9999                                   District: 99
XXXXXXXXXXXXXXXXXX                    XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX                    XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX XX XXXXX-XXXX         XXXXXXXXXXXXXXXXXXX XX
XXXXX-XXXX

Customer: 9999  Cust-Warehouse: 9999  Cust-District: 99
Name:   XXXXXXXXXXXXXXXX XX XXXXXXXXXXXXXXXX      Since:  DD-MM-YYYY
          XXXXXXXXXXXXXXXXXXX                        Credit: XX
                        XXXXXXXXXXXXXXXXXXX                %Disc:  99.99
          XXXXXXXXXXXXXXXXXX XX XXXXX-XXXX        Phone:
XXXXXX-XXX-XXX-XXXX

Amount Paid:            $9999.99                          New Cust-Balance: $-9999999999.99
Credit Limit:   $9999999999.99


Cust-Data:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

D.3 Order-Status Screen

```
(1)New-Order (2)Payment (3)Order-Status (4)Delivery (5)Stock-Level (9)Exit

                                      Order-Status
Warehouse: 9999   District: 99
Customer: 9999    Name: XXXXXXXXXXXXXXXX XX XXXXXXXXXXXXXXXX
Cust-Balance: $-99999.99

Order-Number: 99999999   Entry-Date: DD-MM-YYYY hh:mm:ss   Carrier-Number: 99
Supply-W     Item-Id    Qty     Amount         Delivery-Date
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
  9999         999999     99    $99999.99      DD-MM-YYYY
```

D.4 Delivery Screen

```
(1)New-Order (2)Payment (3)Order-Status (4)Delivery (5)Stock-Level (9)Exit

                                          Delivery
Warehouse: 9999

Carrier Number: 99

Execution Status: XXXXXXXXXXXXXXXXXXXXXXXXX
```

D.5 Stock-Level Screen

```
(1)New-Order (2)Payment (3)Order-Status (4)Delivery (5)Stock-Level (9)Exit

                                          Stock-Level
Warehouse: 9999   District: 99

Stock Level Threshold: 99

low stock: 999
```

Appendix E: RTE Scripts

E.1 RTE Parameters

```
MEASUREMENT="2"
MASTER master
SUT   = "sut1"
SLAVES driver1, driver2, driver3, driver4, driver5, driver6

CLIENT_REAL   = "client1 client2 client3"
CLIENT client1e sybase e26ssyb
CLIENT client2e sybase e26ssyb
CLIENT client3e sybase e26ssyb

TELNET telnet 23
SOCKET socket 199703
SOCKET_NETWORK socket1 199703 driver1, driver2
SOCKET_NETWORK socket2 199703 driver4, driver5
SOCKET_NETWORK socket3 199703 driver3, driver6

OUTPUTNAME="../runs/wild"

CPU=4
NUM_CKPT=2
CKPT_SLEEP=12:30

RAMPUP=35:00
RUNTIME=30:00
RAMPDOWN=15:00
RAMPDOWN_WAIT=2:00
INTERVAL=2:00   /* Interval to calculate mix from */

LOGIN_MAX_LOAD       = 4
LOGIN_BEGIN          = 1
NOBEGIN              = 0
KEYSTROKE_PACKET_SIZE = 0

MAX_CONCURRENT_SPAWN  = 10
SPAWN_COUNT          = 5

MIN_PORT             = 8088
MAX_PORT              = 8089

/* User variables.  Think, Emulex Delay, %desired, %min, %max */
NEWORDER = "12.1, 0, 0"
PAYMENT  = "12.1, 0, 0, 43.05, 40.5, 45.0 "
ORDSTAT  = "10.1, 0, 0,  4.05,  3.9,  4.5 "
DELIVERY = "05.1, 0, 0,  4.05,  3.9,  4.5 "
STOCKLEV = "05.1, 0, 0,  4.05,  3.9,  4.5 "

#if 1
START client1e socket1 2200
START client2e socket2 2200
START client3e socket3 2200
#endif

#define TES_FLAG_TRACE         0x00000010
#define TES_FLAG_KEYSTROKE_TIME 0x00000200
#define TES_FLAG_LOCAL_LOG      0x00000400
#define TES_FLAG_LOCAL_TRACE    0x00000800

#if 0
/* SETFLAG ALL TES_FLAG_TRACE */
SETFLAG ALL TES_FLAG_LOCAL_TRACE
#endif

#if 0
SETFLAG client1x telnet 1 TES_FLAG_KEYSTROKE_TIME
#endif
```

E.2 Master Script

```
/*****************************************************************************
*/
/* user_master.C                                Audit: 05/30/96 */
/*****************************************************************************
*/

static char *rcsid="$Id: user_master.C,v 1.8 1996/11/27 19:58:49 channui Exp channui $";

#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 <string.h>
#include <stdlib.h>
#include <unistd.h>
#include "data/rte.h"
#include "data/Stats.h"
#include "data/misc.h"
#include "user_tpcc.h"

struct header_s {
   int slave;
   int num;
   int type;
   int num_timestamps;
   int user_data_length;
   int data_type;
};


char *get_variable(char *name);
int get_variable(char *name, int *number);
int send_global_data(void);
int make_ratios (double *buffer);
extern int ramp_up_complete;
extern int interval_start_time, interval_stop_time;
extern "C" int strcasecmp(char *s1, char *s2);
extern "C" int strncasecmp(char *s1, char *s2, int n);

struct UserSpawnData {
   int Warehouse;
   int District;
};

/* user_master.C */
int user_statistics_print(void);
int user_spawn(int min, int max, int number, int *length, char *buffer);
int user_finished(int length, char *buffer);

extern SlaveStatus slave_status[MAX_SLAVES];

extern Stats status[MAX_TRAN_TYPE][MAX_TIMES];
extern WINDOW *statistics_win;
extern UserGlobal *shmglobal;

/* 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;

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++) {
                /* 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;
}

extern int login_begin;
int login_max_load;

const int MAX_WAREHOUSES=20000;
/* 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];
int user_spawn(int min, int max, int number, int *length, char *buffer) {
    int i, min_index;
    int adj_wh = num_warehouses;                                    // adjusted warehouse number
    UserSpawnData *ptr = (UserSpawnData *)buffer;
    *length = sizeof(*ptr);

                if (min ==0 && max == 0) {
                                min++;
                                min_index =  0 ;
                } else {
                                adj_wh = max;   // inclusive range of wh-s
                                min = min * 10;
                                min_index = min;
                }
    for (i = min ; i < (adj_wh)*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]++;
  return RTE_OK;
}

int user_finished(int length, char *buffer) {
    UserSpawnData *ptr = (UserSpawnData *)buffer;
    int temp = (ptr->Warehouse-1)*10+ptr->District-1;
    warehouses[temp]--;
  return RTE_OK;
}


double limit(double min, double max, double val) {
    if (val < min)
                return min;
    if (val > max)
                return max;
    return val;
}


int make_ratios (double *buffer) {
    int neword   = status[NEWORDER][0].samples();
    int payment  = status[PAYMENT] [0].samples();
    int ordstat  = status[ORDSTAT] [0].samples();
    int delivery = status[DELIVERY][0].samples();
    int stocklev = status[STOCKLEV][0].samples();
    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;
}

int user_global_update(int *length, char *buffer) {
    UserGlobal *shmglobal = (UserGlobal *)buffer;
    static double last[6];
    static last_test_state = 0;
    static int users_last=-1;
    double ratios[6];
    double current[6];
    int i, different = 0;
    int desired = 0;
    int host_busy;

    *length = sizeof(*shmglobal);

    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 */
    if (ramp_up_complete) {
                current[NEWORDER] = 100.0;
                for (i = 2; i < 6; i++) {
                    if (ratio_desired[i] > ratios[i]) {
                                current[i] = ratio_max[i];
                    } else {
                                current[i] = 2*ratio_desired[i] - ratios[i];
                                if (current[i] < ratio_min[i])
                                    current[i] = ratio_min[i];
                    }
                    current[NEWORDER] -= current[i];
                }
    } else {
                for (i = 1; i < 6; i++) {
                    current[i] = ratio_desired[i];
                }
    }

    /* Add up all the users */
   shmglobal->total_users = 0;
    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;

    host_busy = 0;
    for (i = 1; i <= 5; i++) {
                if ( status[i][current_status].average()/1000.0 > login_max_load )
                    host_busy = 1;
    }
    if (host_busy != shmglobal->host_busy) {
                shmglobal->host_busy = host_busy;
                different = 1;
    }

    for (i = 2; i < 6; i++) {
                if (current[i] != last[i])
                    different = 1;
    }

    if (last_test_state != shmglobal->test_state) {
                different = 1;
                last_test_state = shmglobal->test_state;
    }
```

```
    // Don't send if it's the same as last time
    if ( !different && shmglobal->total_users == users_last ) {
                return RTE_ERROR;
    }

    users_last = shmglobal->total_users;
    for (i = 1; i < 6; i++) {
                shmglobal->chances[i] = last[i] = current[i];
    }

    return RTE_OK;
}


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;
}

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;
}


int user_init() {
    double  dbuffer[32];
    int   rc, i;
    char *ptr;

    if (get_variable("KEYSTROKE_SLEEP", &shmglobal->keystroke_sleep) != RTE_OK) {
                shmglobal->keystroke_sleep = 0;
    }
    if (get_variable("LOGIN_TIMEOUT", &shmglobal->login_timeout) != RTE_OK) {
                shmglobal->login_timeout = 120; /* 2 minutes */
    }
    if (get_variable("KEYSTROKE_PACKET_SIZE", &shmglobal->keystroke_packet_size) !=
RTE_OK) {
                shmglobal->keystroke_packet_size = 0;
    }
    shmglobal->login_timeout *= 1000;
    if (get_variable("LOGIN_MAX_LOAD", &login_max_load) != RTE_OK) {
                login_max_load = 1;
    }
    if (get_variable("WAREHOUSES", &num_warehouses) != RTE_OK) {
                num_warehouses = -1;
    }
    if (get_variable("LASTC", &shmglobal->lastc) != RTE_OK) {
                shmglobal->lastc = 193; /* 2 minutes */
    }
    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);
    }

    if (!(ptr = get_variable("NEWORDER"))) {
                iprint_error ("Error. NEWORDER variable not found\n");
                exit (1);
    }
    if (parse_array(ptr, 3, dbuffer)!=3) {
                iprint_error ("Error. NEWORDER should be think, emulex_menu,
emulex_response");
                exit (1);
    }
    shmglobal->think          [NEWORDER] = dbuffer[0];
    shmglobal->emulex_menu    [NEWORDER] = dbuffer[1];
    shmglobal->emulex_response[NEWORDER] = dbuffer[2];
    shmglobal->test_state = 0;

    for (i = 2; i < 6; i++) {
                if (!(ptr = get_variable(ratio_names[i])) ||
                    (parse_array(ptr, 6, dbuffer)!=6)) {
                    iprint(__FILE__, __LINE__, IPRINT_ERROR,
                        "Error. %s should be think, emulex_menu, emulex_response, desired,
min, max",
                                ratio_names[i]);
                    exit (1);
                }
                shmglobal->think[i]          = dbuffer[0];
                shmglobal->emulex_menu[i]    = dbuffer[1];
                shmglobal->emulex_response[i] = dbuffer[2];
                ratio_desired[i]             = dbuffer[3];
                ratio_min[i]                 = dbuffer[4];
                ratio_max[i]                 = dbuffer[5];
                ratio_range[i]               = ratio_max[i]-ratio_min[i];
    }

    return RTE_OK;
}

int user_extra_data(header_s *header) {
    int i;
    int 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;
}

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, "pause")) {
                shmglobal->test_state = 1;
    } else if (!strcasecmp (ptr, "warmup")) {
                shmglobal->test_state = 2;
    } else if (!strcasecmp (ptr, "notest")) {
                shmglobal->test_state = 0;
    } else 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;
}
```

E.3 User Script

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

```
/* user_slave.C                                    Audit: 05/30/96 */
/*****************************************************************************
*/

static char *rcsid="$Id: user_slave.C,v 1.9 1996/11/27 19:53:38 channui Exp channui $";

/*****************************************************************************/
/***                TPCC FILE FOR ALL USERS                        ***/
/*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/time.h>
#include "rte_slave.h"
#include "user_tpcc.h"

/* This MUST match the corresponding one in client's inout.h file! */
#define TRIGGER "\021"
#define EXPECT_TIMEOUT 600000

extern SHM_Slave *shm;
extern TableEntrySlave *shmentry;
extern DriverStatus    *status;
extern echo_trace(char *);
extern echo_trace();
extern char *expect_save;

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";

int             WHSEID;                         /* warehouse number for each users */

/*****************************************************************************/
/* The "uniform()" function has range of the absolute value of the       */
/* difference between the min. and the max values upto 2147483647.      */
/*****************************************************************************/
/*-------------------------------------------------------*/
/* NURand                          */
/*-------------------------------------------------------*/
/* 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 */
/*-------------------------------------------------------*/
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;
}

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

#if 0
   random_num = NURand(255, 0, 999, shmglobal->lastc);
#else
   random_num = NURand(255, 0, 999, LASTC);
#endif
   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);
}

typedef struct gen_tran_s {
   int invalid;
   void *data;
   long len;
   long keywait;
   long type;
   char *menu;
   char *request;
} gen_tran_t;

int generic_transaction( gen_tran_t *data ) {
   char buffer[2048];
   int rc;
   set_typing_delay(0);
#ifndef NOSLEEP
   if (shmglobal->test_state == 0)
               transaction_sleep_do();
#endif

#ifdef EXPECT_TIMEOUT
   int timeout = EXPECT_TIMEOUT;
#else
   int timeout = 0;
#endif

   // Start the transaction (MENU)
   transaction_start(data->type, data->len, data->data);

   transmit(data->menu);
   echo_trace ("Waiting for Menu (DELIVERY)");
   if (expect(TRIGGER, timeout) == ERROR) {
               iprint (IPRINT_ERROR, "Slave %d: Failed to receive %s screen\n",
                     shmentry->num, data->menu);
               return (ERROR);
   }
#ifndef NOSLEEP
   usleep(shmglobal->emulex_menu[data->type]*1000000.0+0.9);
#endif

   // Send our request (KEYING)
   transaction_mark(WHERE_NOW);
   echo_trace ("Keying");

#ifndef NOSLEEP
   sleep(data->keywait); // Keying delay
#endif
   transmit(data->request);

   // Wait for response (RESPONSE)
   transaction_mark(WHERE_NOW);
   echo_trace ("Wait for Response");
   if (expect(TRIGGER, timeout) == ERROR) {
               iprint (IPRINT_ERROR, "Slave %d: Failed to receive %s response\n",
                           shmentry->num, data->menu);
               return (ERROR);
   }
#ifndef NOSLEEP
   usleep(shmglobal->emulex_response[data->type]*1000000.0+0.9);
#endif

   // Look for errors and set our think time (THINK)
   transaction_mark(WHERE_NOW);
   if (expect_after_match ("ERROR: ")) {
               data->invalid = 1;
               iprint (IPRINT_ERROR, "Slave %d: %s found '%s'\n",
                     shmentry->num, data->menu, "ERROR:");
               return RTE_ERROR;
   }
   echo_trace ("Thinking");
   transaction_sleep_set(neg_exp_4(shmglobal->think[data->type])*1000.0);
   return (RTE_OK);
}

/*****************************************************************************/
/***                Delivery Transaction                        ***/
/*****************************************************************************/
int
Delivery()
{
   static struct delivery_struct delivery, delivery_new;
   int            rc;
   char          *ptr;
   char               buffer[256];
   gen_tran_t      tran;

   tran.invalid = 0;
   tran.data    = &delivery;
   tran.len     = sizeof(delivery);
   tran.keywait = 2;
```

```
    tran.type    = DELIVERY;
    tran.menu    = "4";
    tran.request = buffer;

    // Set up all data for new transactions
    delivery_new.carrier = uniform(1, 10);      // carrier # 1 to 10

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\n", delivery_new.carrier);

    // Go do the transaction
    rc = generic_transaction(&tran);
    delivery         = delivery_new;
    delivery.invalid = tran.invalid;

    return (rc);
}

/*****************************************************************************/
/***                  New Order Transaction                              ***/
/*****************************************************************************/
int NewOrder() {
    static struct neword_struct neword, neword_new;
    int             i, rc, whses, low_whse=1;
    char            buffer[2048];
    char            *ptr;
    const char      *ptr2;
    gen_tran_t      tran;

    tran.invalid = 0;
    tran.data    = &neword;
    tran.len     = sizeof(neword);
    tran.keywait = 18;
    tran.type    = NEWORDER;
    tran.menu    = "1";
    tran.request = buffer;

    neword_new.rollback=0;

    /*** SECTION TO DETERMINE ROLLBACK TRANSACTION FOR 1% OF NEW
ORDERS ***/
    neword_new.did = uniform(1, 10);                        // district number
    neword_new.cid = NURand(1023, 1, 3000, CUSTC);    // customer # 1 to 3000
    neword_new.nloop = uniform(5, 15);        // number of items to order (5-15)
    neword_new.olremote=0;  // find total number of remote order-lines

    whses = shmglobal->max_warehouses;

    for (i = 0; i < neword_new.nloop; i++) {
                // Warehouse Number
                neword_new.item[i].olswid = WHSEID;
                if (whses > 1 && (uniform(0.0, 100.0) < 1.0)) {
                   /* for 1% of items (if * uniform()==0) */
                   /* Generate a uniform whse number that's different from WHSEID */
                   neword_new.item[i].olswid =
                                (long) uniform((long) low_whse, (long)whses-1);
                   if (neword_new.item[i].olswid >= WHSEID)
                                neword_new.item[i].olswid++;
                   neword_new.olremote++;  // find total number of remote order-lines
                }
                // Item number 1-100000
                neword_new.item[i].oliid = NURand(8191, 1, 100000, ITEMC);
                // Quantity 1-10
                neword_new.item[i].olquantity = uniform(1, 10);
    }                                                       /* end of for n_loop */
     if (uniform(1, 5000) <= 50)
                neword_new.item[neword_new.nloop-1].oliid = 999999;

    neword_new.oremote = (neword_new.olremote > 0);

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\t%d", neword_new.did, neword_new.cid);
    for (i = 0; i < neword_new.nloop; i++) {
                ptr += sprintf(ptr, "\t%d\t%d\t%d",
                                        neword_new.item[i].olswid,
                                        neword_new.item[i].oliid,
                                        neword_new.item[i].olquantity);
    }
    ptr += sprintf(ptr, "\n");

    // Go do the transaction
    rc = generic_transaction(&tran);
    neword         = neword_new;
    neword.invalid = tran.invalid;

    // Check for a rollback
    if (expect_after_match (ROLLBACK_MESSAGE)) {
                neword.rollback=1;
                echo_trace ("Found rollback!\n");
    }

    // Grab the orderID from the
    if (!(ptr2 = expect_after_match("\033[6;15H"))) {
                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(ptr2+8);
    }

    if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME) {
                log_data(RTE_ITEM_KEYSTROKE_TIMES,
keystroke_length*sizeof(int),keystroke_times);
    }

    return (rc);
}


/*****************************************************************************/
/***                  Order Status Transaction                           ***/
/*****************************************************************************/
int OrderStatus() {
    static struct ordstat_struct ordstat, ordstat_new;
    char            buffer[2048];
    int             rc;
    char            *ptr;
    gen_tran_t      tran;

    tran.invalid = 0;
    tran.data    = &ordstat;
    tran.len     = sizeof(ordstat);
    tran.keywait = 2;
    tran.type    = ORDSTAT;
    tran.menu    = "3";
    tran.request = buffer;

    // Set up all data for new transactions
    ordstat_new.did = uniform(1, 10);             /* district number 1 to 10 */
    if (uniform(1, 100) <= 60) {/* for 60% of transactions */
                char *tmp = getname();
                strcpy(ordstat_new.clast, tmp);                 /* by customer last name */
                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);
                   return RTE_ERROR;
                }
                ordstat_new.byname = 1;
                ordstat_new.cid = 0;
    } else {
                ordstat_new.cid = NURand(1023, 1, 3000, CUSTC);          /* cust. # 1 to 3000
*/
                ordstat_new.byname = 0;
                ordstat_new.clast[0] = (char) NULL;
    }

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\t", ordstat_new.did);
    if (ordstat_new.byname) {
                ptr += sprintf(ptr, "\t%s\n", ordstat_new.clast);
    } else {
                ptr += sprintf(ptr, "%d\n", ordstat_new.cid);
    }

    // Go do the transaction
    rc = generic_transaction(&tran);
    ordstat         = ordstat_new;
    ordstat.invalid = tran.invalid;

    return (rc);
}

/*****************************************************************************/
/***                  Payment Transaction                                ***/
/*****************************************************************************/
int
Payment()
{
    static struct payment_struct payment, payment_new;
    int             dollars, cents, rc, whses, low_whse = 1;
    char            buffer[2048];
    char            *ptr;
    gen_tran_t      tran;

    tran.invalid = 0;
    tran.data    = &payment;
    tran.len     = sizeof(payment);
    tran.keywait = 3;
```

```
    tran.type    = PAYMENT;
    tran.menu    = "2";
    tran.request = buffer;

    payment_new.did = uniform(1, 10);           /* district number 1 to 10 */
    if (uniform(1, 100) <= 60) {/* for 60% of transactions */
                strncpy(payment_new.clast, getname(), 17);    // by customer last name
                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);
                   return RTE_ERROR;
                }
                payment_new.byname = 1;
                payment_new.cid = 0;
    } else {
                payment_new.cid = NURand(1023, 1, 3000, CUSTC);         /* cust. # 1 to 3000
*/
                payment_new.byname = 0;
                payment_new.clast[0] = (char) NULL;
    }

    whses = shmglobal->max_warehouses;

    if (whses < 2 || uniform(1, 100) <= 85) {/* for 85 % of transactions */
                payment_new.cwid = WHSEID;
                payment_new.cdid = payment_new.did;
                payment_new.remote = 0;
    } else {                                    /* for 15 % of transactions */
                payment_new.cwid = (long) uniform((long)low_whse, (long) whses-1);
                if (payment_new.cwid >= WHSEID)
                   payment_new.cwid++;

                payment_new.remote = 1;
                payment_new.cdid = uniform(1, 10);              /* district 1 to 10 */
    }

    dollars = uniform(1, 5000); /* dollar amt = 1 to 5000 */
    if (dollars == 5000)
                cents = 0;
    else
                cents = uniform(0, 99);

    payment_new.amount = ((double) dollars) + ((double) cents) / 100.0;

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\t", payment_new.did);
    if (payment_new.byname) {
                ptr += sprintf(ptr, "\t%s\t", payment_new.clast);
    } else {
                ptr += sprintf(ptr, "%d\t\t", payment_new.cid);
    }
    ptr += sprintf(ptr, "%d\t%d\t", payment_new.cwid, payment_new.cdid);
    ptr += sprintf(ptr, "%d.%02.2d\n", dollars, cents);

    // Go do the transaction
    rc = generic_transaction(&tran);
    payment         = payment_new;
    payment.invalid = tran.invalid;

    return (rc);
}

/*****************************************************************************/
/***                 Stock Level Transaction                           ***/
/*****************************************************************************/
int
StockLevel()
{
    static struct stocklev_struct stocklevel, stocklevel_new;
    char            buffer[2048];
    int             rc;
    char            *ptr;
    gen_tran_t      tran;

    tran.invalid = 0;
    tran.data    = &stocklevel;
    tran.len     = sizeof(stocklevel);
    tran.keywait = 2;
    tran.type    = STOCKLEV;
    tran.menu    = "5";
    tran.request = buffer;

    stocklevel_new.invalid = 0;
    stocklevel_new.threshold = uniform(10, 20);                 /* uniform no. between 10 and
                                                                                * 20
*/
    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\n", stocklevel_new.threshold);

    // Go do the transaction
    rc = generic_transaction(&tran);
    stocklevel         = stocklevel_new;
    stocklevel.invalid = tran.invalid;

    return (rc);
}


/*****************************************************************************/
/***  MAIN()                                                               ***/
/*****************************************************************************/
int
user_transaction()
{
    char            logout[32];
    double          ntask;
    int             resp;
    static int task = 0;

    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;
    }

#if 1
    switch (shmglobal->test_state) {
    case 0: // Normal
                break;
    case 1: // pause
                sleep (1);
                return RTE_OK;
    case 2: // warmup
                switch(task++) {
                case 0: return Delivery();
                case 1: return OrderStatus();
                case 2: return Payment();
                case 3: return StockLevel();
                case 4: task = 0; return NewOrder();
                }
    }

/*****************************************************************************/
    /***  CHOOSE ONE OF THE TRANSACTIONS                                   ***/

/*****************************************************************************/
    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();
#else

    // use a card deck with no replacement to fulfill the requirements
    {
                int deck[100], count=-1, i, size=1, tmp;
                // lock deck
                if (count < 0) {
                   // deck is empty fill it up
                   count = 0;
                   for (i = 0; i < 43 * size; i++) {
                                deck[count++] = Payment;
                   }
                   for (i = 0; i < 4 * size; i++) {
                                deck[count++] = StockLevel;
                   }
                   for (i = 0; i < 4 * size; i++) {
                                deck[count++] = OrderStatus;
                   }
                   for (i = 0; i < 4 * size; i++) {
                                deck[count++] = Delivery;
                   }
                   for (; count < 100 * size; i++) {
```

```
                        deck[count++] = NewOrder;
                }
                // randomize the deck
                for (i = 0; i < 100 * size; i++) {
                        int tmp;
                        int pick = uniform(i+1, 100);
                        tmp = deck[i];
                        deck[i] = deck[pick];
                        deck[pick] = tmp;
                }
        }
        tmp = deck[count--];
        // unlock deck
        switch(tmp) {
        case Delivery:    return Delivery();
        case OrderStatus: return OrderStatus();
        case Payment:     return Payment();
        case StockLevel:  return StockLevel();
        case NewOrder:    return NewOrder();
        }
    }
#endif

#if 0
    if (resp != RTE_OK) {                       /* logoff if response is not correct */
                strcpy(logout, "9\n");          /* 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     = ", shmentry->num);
    for (i = 0; i < MAX_TRAN_TYPE; i++)
                iprint(IPRINT_TRACE, "%6.2f ", shmglobal->chances[i]);
    iprint(IPRINT_TRACE, "\nSlave %d: think       = ", shmentry->num);
    for (i = 0; i < MAX_TRAN_TYPE; i++)
                iprint(IPRINT_TRACE, "%6.2f ", shmglobal->think[i]);
    iprint(IPRINT_TRACE, "\n");
    #endif
    return RTE_OK;
}

int user_login(char *user, char *password, void *data) {
    UserLocal *localdata = (UserLocal *)data;
    int rc;
    int timeout_value = shmglobal->login_timeout;
    char buffer[32];
    set_typing_delay(0);

    rc = expect   (TRIGGER, timeout_value);
    if (rc == RTE_ERROR) {
                iprint (IPRINT_ERROR, "Slave %d: didn't find Warehouse prompt\n",
shmentry->num);
    }
    sprintf(buffer, "%d\t%d\n", localdata->Warehouse, localdata->District);
    transmit(buffer);

    rc = expect (TRIGGER, timeout_value);
    if (rc != RTE_OK) {
        iprint (IPRINT_ERROR, "Slave %d: Failed logging in\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 () {
    transaction_sleep_do();
    transaction_start(0, 0, NULL); // Just something to clear out the buffer...
    return RTE_OK;
}

int user_spawn_ok() {
    int rc, hb;
    hb = ((UserGlobal *)(shm->global_data))->host_busy;
    rc = hb?RTE_ERROR:RTE_OK;
    return rc;
}

/*****************************************************************************
*/
/* user_tpcc.h                                  Audit: 05/30/96 */
/*****************************************************************************
*/

/* $Id: user_tpcc.h,v 1.6 1996/11/27 19:53:51 channui Exp $ */

#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.                   ***/
/*****************************************************************************/
/* Change for 3.1 */
#define         LASTC         193
#define         CUSTC         319
#define         ITEMC         3849

/*****************************************************************************/
/***   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;
    int lastc;
    int test_state;
    int host_busy;
    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
```

DICKENS data systems

your source for total IBM business solutions

February 9, 1998

Mr. Dee Prewit
IBM Corporation
11400 Burnet Road
Austin, TX 78758

Dear Mr. Dee Prewit,

Thank you for the opportunity to quote the following RS/6000 F50 166 MHz server and RS/6000 clients. Attached is the hardware systems, software, and maintenance pricing for all items that we having been discussing.

Let's review the quote at your earliest convenience. The attached quotation is valid for ninety days from February 9, 1998.

Thank you for your time, and I look forward to speaking with you soon.

Sincerely,

Dickens Data Systems

Peter Wells
RS/6000 Product Manager
800-448-6177 Ext. 7658
770-625-7525 Fax
peterw@dickens.com

cc: Robyn Feinberg, Dickens Data Systems

IBM Business Partner

1175 Northmeadow Parkway • Suite 150 • Roswell, GA 30076
770/625/7500 • fax 770/625/7525 • 800/448/6177 • www.dickens.com

Product	Description	Qty	Purchase Price	Maintenance (5 Years)
	Server Hardware			
7025-F50	RS/6000 Server Model F50	1	19,900	11,952
	1604e, 128mb Memory, 4,5GB Disk,CDROM,			
	2 Intg SCSI2 F/W Adptr, Intg Ethernet Adapter			
4303	604e 2way Proc Select, 2-256kb L2	1	4,000	3,120
4309	604e 2way Proc Card, 2-256kb L2	1	8,000	6,240
4106	256MB DIMM Memory Select	1	3,200	0
4110	256MB DIMM Memory Modules	11	70,400	0
4093	Memory Expansion Feature 2nd Card	1	1,038	0
2911	9.1GB SCSI-2 F/W Hot Swap Disk	8	30,800	0
3019	9.1GB Hot Swap SCSI Disk Select	1	1,800	0
6519	SCSI Hot Swap 6-Pack Kits	2	750	0
2444	16-bit Integrated SCSI Adapter Cable	1	66	0
2985	PCI Ethernet Adapter	2	390	0
6208	PCI SCSI-2 F/W Adapter	1	360	0
6218	PCI SSA 4port RAID Adapter	3	9,000	0
7015-R00	System Rack Model R00	2	6,220	2,976
7133-020	SSA Disk Subsystem w/ 4 4.5GB Disk	7	135,450	67,200
3401	4.5 GB Disk Drive Module	80	168,000	0
5010	SSA Cables	14	560	0
3153-BG3	IBM ASCII Terminal, Keyboard	4	2,308	2,640
	Client Hardware			
7043-140	RS/6000 Model 43P-140, 233 MHz	3	24,000	10,080
	2.1g b Disk, Intg SCSI-2 FW, Intg Enet			
4106	128MB DIMM Memory	3	3,840	0
4115	128MB DIMM Memory Expansion	12	30,720	0
2934	Async Terminal/Printer Cable	4	180	0
2985	Ethernet Adapter, PCI	9	1,755	0
	Hardware Subtotal		522,737	104,208
	Software			
5765-C34	AIX 4.2.1 F50 + Support	1	240	0
5756-030	AIX 4.1.5 Unlimited Users	3	10,980	0
	Software Subtotal		11,220	0
	SYSTEM TOTALS			
	RS/6000		533,957	104,208
	Sybase Adaptive Server Enterprise 11.5 incl Client	1	35,995	28,000
	Tuxedo EPT Ver 6.2	3	9,000	6,750
	Ethernet Hubs +10% Spares	303	65,325	0
	System Total		**644,277**	**138,958**
	System Total with Support			**783,235**
	Dickens Data Systems' Solution excluding tax/ship			**510,535**

The above quotation is valid for ninety days from February 9, 1998

BEA SYSTEMS, INC.
140 Allen Road, Rm.133
Libery Corner, NJ 07938
Phone:(908)580-3026
FAX: (908)580-3049
From: Berny Wright

PROPOSAL

Feb. 11, 1998

For:
Mr. Dee Prewit
IBM Corporation
11400 Burnet Road
MS 9221
Austin, TX 78758

ITEM	TYPE	DESCRIPTION	QTY	UNIT PRICE	TOTAL
1	License	BEA TUXEDO 6.2 Core Functionality Services	1	$3,000	$3,000
2	Service	Annual Maintenance Standard 5x8 Service	1	$450	$450
		Total		$3,450	$3,450

Note: This quotation is v valid for 90 days from quotatio n and is subject to the BEA Systems Inc. terms and conditions.

BNright

Senior Sales Executive-Berny Wright

FAXSR:_At_01/22/'98 14:30:08,Page 1

DataComm Warehouse, Inc.

1690 Oak Street, Lakewood, NJ 08701 USA

1-800-328-2261 US SALES - TECHNICAL SUPPORT
1-732-363-4823 FAX NUMBER

DATE: January 22, 1998

NAME: Dee Prewitt

COMPANY: IBM

FAX #: 512 838 1258

Thank you for your interest in DataComm Warehouse
Below is the information you requested.

Part #	QTY	Description	Unit Cost	Price	
DEH1487	363	ALLIED TELYSN 24 PORT 10BASE-T HUB W/AUI + BNC AT-3024TR	215.95	$78,389.85	
		Total		$78,389.85	

Additional Information:
Thank you. Any questions please feel free to call me.

If I can be of further assistance, please contact me at:
Phone. 1-800-328-2261 FAX 1-732-363-4823 EMAIL: grossm@mwhse.com

This quotation is valid for 90 days.
Thank you

Mark Gross 20278
ACCOUNT MANAGER EXTENSION

IF THIS TRANSMISSION IS INCOMPLETE OR ILLEGIBLE, PLEASE CALL 1-908-370-3801

JAN 22 '98 13:34 PAGE.02

To: Dee Prewit/Austin/IBM

cc:

From: Don Johnson/Austin/IBM

Subject: Re: IBM/SYBASE Partnership

FYI...

To: Don Schiro/Poughkeepsie/IBM@IBMUS

cc: Don Johnson/Austin/IBM@IBMUS

From: Brian Sweeney/Atlanta/IBM@ IBMUS

Subject: Re: IBM/SYBASE Partnership

Is there anything missing here?

Brian J. Sweeney

VP, RS/6000 Enterprise Servers

IBM Server Group

4111 Northside Parkway, H09B1, Atlanta, GA 30327

(404) 238-7016 or T/L 888-7016 FAX: T/L 888-2051

bjsween@us.ibm.com

---------- Forwarded by BrianSweeney/Atlanta/IBMon 02/02/98 03:13 PM----------

Please respond tomregan@sybase.com@ internet

To: Brian Sweeney/Atlanta/IBM@ibmus

cc: Mike Borman/Somers/IBM@IBMUS, David Gelardi/Poughkeepsie/IBM@ibmus,Don Schiro/Poughkeepsie/IBM@ibmus,Don Johnson/Austin/IBM@ibmus, rnathan@sybase.com@ internet, Jenny King/SanFrancisco/IBM@IBMUS, mforster@sybase.com@ internet

Subject: Re: IBM/SYBASE Partnership

Brian,

I wanted to confirm for you the pricing on the F50. Here is the standard

format we use when we submitTPCC numbers:

Workplace Pricing

Pdt Pricing $34,995

Open Client $ 1000

Support Pricing

16% x 35K x 5yrs $28,000

Total Price $64,000

Brian, I appreciate your support in this effort and look forward to meeting

you on Feb 17th at the IBM/Sybase event in San Francisco

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

Sponsor: Subra Balan
IBM RS/6000 Performance
11400 Burnet Road
Austin, Texas 78758

May 12, 1997

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

Platform: RISC System/6000 Workgroup Server F50 c/s
Operating system: AIX 4.2.1
Database Manager: Sybase Adaptive Server Enterprise 11.5
Transaction Manager: Tuxedo Version 6.2

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: RISC System/6000 Workgroup Server F50 c/s				
4 x PowerPC 604e (166 MHz)	3072 MB (256 KB L2 Cache per processor)	82 x 2.2 GB 26 x 4.5 GB 9 x 9.1 GB	1.54 Seconds	8142.40
Three Clients: RISC System/6000 Workgroup Server 43P-140 (Specification for each)				
1 x PowerPC 604e (233 MHz)	640 MB	1 x 2.2 GB	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for Revision 3.3 of the benchmark. The following verification items were given special attention:

- The transactions were correctly implemented
- The database records were the proper size
- The database was properly scaled and populated
- The ACID properties were met
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.

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

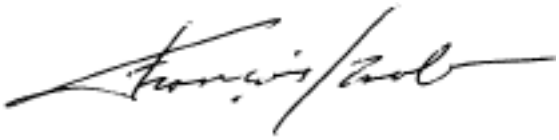
- At least 90% of all delivery transactions met the 80 Second completion time limit
- All 90% response times were under the specified maximums
- The measurement interval was representative of steady state conditions
- The reported measurement interval was 30 minutes.
- One checkpoint was taken during the measurement interval
- Measurement repeatability was verified
- The 180 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

The (82) 2.2 GB disks used in the tested configuration were substituted with 4.5 GB disks in the priced configuration. Based on the specifications of the disks and on measurement data collected, it is my opinion that this substitution would have no negative effect on the reported performance.

The 3-Com 3C905-TX-IBM Fast EtherLink cards used in the tested configuration (1 for each client), were substituted with PCI Ethernet Adapters in the priced configuration. Based on the specifications of the Ethernet cards and on measurement data collected, it is my opinion that this substitution would have no negative effect on the reported performance.

Respectfully Yours,



François Raab
President

RISC System/6000 Workgroup Server F50

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

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

Test Sponsor: John Fowler
IBM RS/6000 Performance
11400 Burnet Road
Austin, Texas 78758

February 11, 1998

In May of 1997, I verified the performance of the RISC System/6000 Workgroup Server F50 C/S executing the TPC Benchmark™ C under AIX 4.2.1, using Sybase Adaptive Server Enterprise 11.5, and Tuxedo Version 6.2.

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: RISC System/6000 Workgroup Server F50 c/s				
4 x PowerPC 604e (166 MHz)	3072 MB (256 KB L2 Cache per processor)	82 x 2.2 GB 26 x 4.5 GB 9 x 9.1 GB	1.54 Seconds	8142.40
Three Clients: RISC System/6000 Workgroup Server 43P-140 (specification for each)				
1 x PowerPC 604e (233 MHz)	640 MB	1 x 2.2 GB	n/a	n/a

I recently verified the repricing of this configuration. The five year cost for the RISC System/6000 Workgroup Server F50 is $510,643. In my opinion the new pricing meets the requirements of the TPC-C standard Revision 3.3.2.

Respectfully Yours,

François Raab
President

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