TPC Benchmark™ C
Full Disclosure Report

IBM PC Server 704 c/s

using

Oracle7 and SunSoft Solaris V2.5.1

IBM Corporation
PC Server Performance Laboratory

December 23, 1996

First Edition - December 1996

The information contained in this document is distributed on an AS IS basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is the customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used.

This publication was produced in the United States. IBM may not offer the products, services, or features discussed in this document in other countries, and the information is subject to change without notice. Consult your local IBM representative for information on products and services available in your area.

Trademarks

IBM is a registered trademark of International Business Machines Corporation.

The following terms used in this publication are trademarks of other companies as follows: TPC Benchmark, trademark of Transaction Processing Performance Council; Intel and Pentium Pro are trademarks or registered trademarks of Intel Corporation; SunSoft and Solaris are trademarks or registered trademarks of Sun Microsystems; Oracle and ORACLE7 are trademarks of Oracle Corporation. ORACLE, SQL*DBA, SQL*Loader, SQL*Net, and SQL*Plus are registered trademarks of Oracle Corporation. Oracle7, Pro+C and PL/SQL are trademarks of Oracle Corporation.

Other company, product, or service names, which may be denoted by two asterisks (**), may be trademarks or service marks of others.

Notes

[1] MHz denotes internal clock speed of the microprocessor; other factors also affect application performance.
[2] GB is the abbreviation for gigabyte; 1GB equals one billion bytes.

Table of Contents

Executive Summary

General Items 11

1 Clause 1: Related Items 15

2 Clause 2: Related Items 16

3 Clause 3: Related Items 18

4 Clause 4: Related Items 24

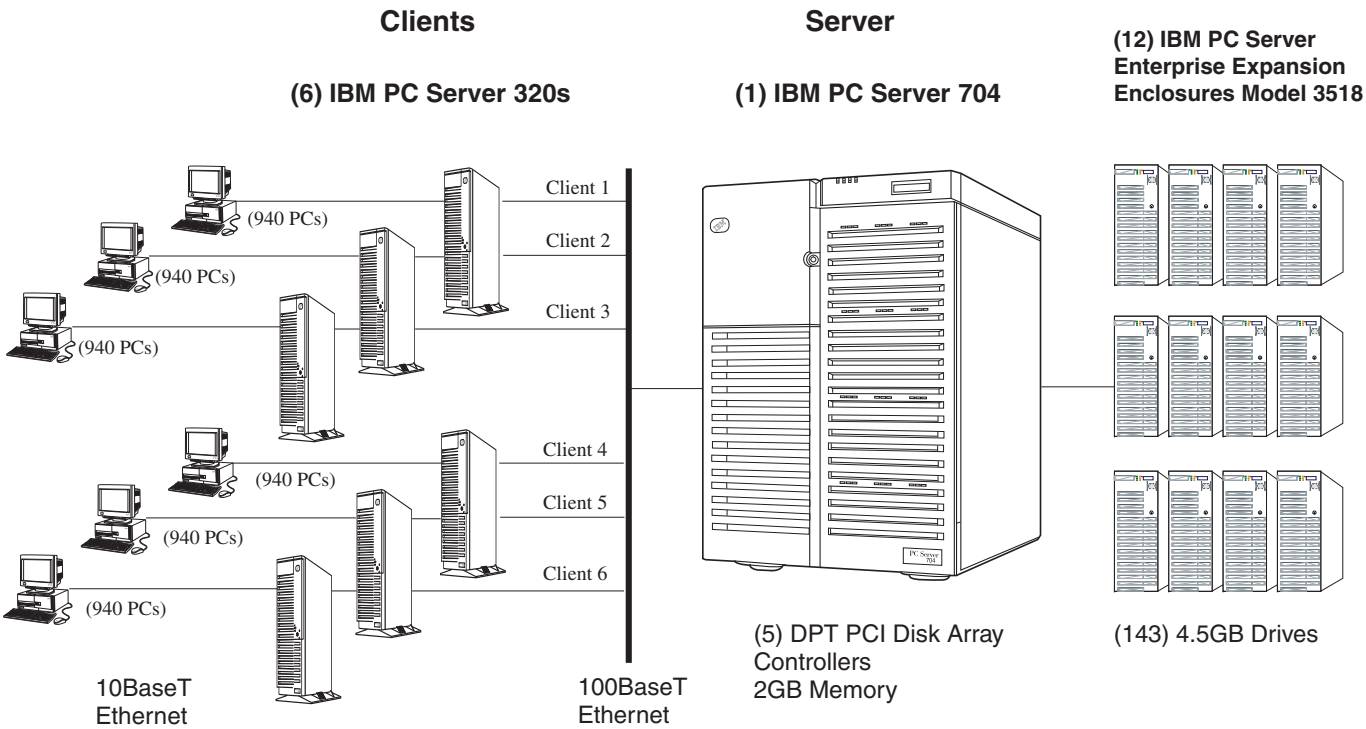
IBM PC Server 704

TPC-C-C Rev 3.2.2

Report Date: Dec. 23, 1996

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$587,986	**6679.50tpmC**	**$88.00/tpmC**	**Hardware/OS: Now** **Oracle7: 30 May 1997**

Processors	Database Manager	Operating System	Other Software	Number of Users
4 - 200MHz Pentium® Pro	**Oracle7 v. 7.3.3**	**SunSoft Solaris 2.5.1**	**Tuxedo /T V4.2.2** **ProWorks C++ V4.01**	**5640**



System Components	Qty	Server	Qty	Clients
Processors	**4**	**200MHz Pentium Pro**	**6**	**IBM PC Server 320**
		512KB Level 2 Cache	**6**	**133MHz Pentium**
Memory	**1**	**2048MB**	**6**	**256MB**
Disk Controllers	**5**	**DPT PCI SCSI Disk Controller**	**6**	**SCSI-2 F/W PCI Adapter**
Disk Drives	**143**	**IBM 4.5GB SCSI-2 F/W Hot-Swap Drives**	**6**	**2.25GB SCSI-2 Drive**
Total GB of Storage		**643.5**		
Tape Drives	**1**	**IBM 4/10GB 4mm DAT**		
Terminals	**1**	**15-inch Display**	**6**	**15-inch Display**
Terminal Connectivity			**240**	**AccuLAN 10Mbps 24-Port Ethernet Hubs**

Description	Part Number	Third Party	Brand	Pricing	Unit Price	Qty	Extended Price	5 Yr. Maint. Price
Server Hardware								
IBM PC Server 704	8650-4MO			1	13,696	1	$13,696	4,800
2 Pentium Pro 200MHz Processor Boards							0	0
1 Pentium Pro 200MHz Processor							0	0
128MB ECC Memory							0	0
PC Server 704 SMP 200MHz Processor Upgrade	94G6678			1	2,544	3	7,632	0
128MB Memory Option	94G6682			1	3,989	16	63,824	0
DPT PCI SCSI Disk Controller	PM3334/W			3	946	5	4,730	0
DPT Dual-Channel Expansion Module	SX4030/2W			3	452	5	2,260	0
IBM PC Server Expansion Enclosure	3518001			1	2,016	12	24,192	11,520
IBM PC Server Hot-Swap Backplane III	70G9855			1	176	12	2,112	0
3-Meter Cable Option	94G5567			1	64	12	768	0
Backplane to Backplane Cable	94G4070			1	43	12	516	0
IBM SCSI-2 Fast/Wide Hot-Swap Drive Tray III	70G9860			1	85	143	12,155	0
SCSI-2 Fast/Wide 4.5GB Hot-Swap Drive	94G6489			1	1,322	143	189,046	0
IBM PC Server 500 Power Supply Upgrade	70G9739			1	170	12	2,040	0
IBM 100/10 PCI Ethernet Adapter	25H4374			1	159	1	159	0
IBM G-50 LV-N 15-Inch Color Monitor	6543301			1	382	1	382	236
4/10GB 4mm DAT Tape Drive	74G8631			1	1,299	1	1,299	594
							$324,811	$17,150
Server Software								
Solaris Workgroup Server Version 2.5.1	251CDBWGS	SunSoft		1	769	1	769	4,200
Oracle7 Workgroup Server v. 7.3.3		Oracle		2	29,233	1	29,233	29,233
						Subtotal	$30,002	$33,433
Client Hardware								
IBM PC Server 320	8640-0DV			1	4,382	6	$26,292	$8,100
1 Pentium/133MHz, 16MB Memory				1			0	0
1 2.25GB SCSI-2 Disk Drive				1			0	0
SCSI-2 Fast/Wide PCI Adapter				1			0	0
Quad-Speed CD-ROM Drive				1			0	0
Select 32MB SIMMs	92G7205			1	372	48	17,856	0
IBM 100/10 PCI Ethernet Adapter	25H4374			1	159	12	1,908	0
IBM G50 LV-N Color Monitor (15-inch)	6543301			1	382	6	2,292	1,416
						Subtotal	$48,348	$9,516
Client Software								
Solaris Workgroup Server Version 2.5.1	251CDBWGS	SunSoft		1	769	6	4,614	$25,200
ProWorks C++ 4.01	PWX-C++4.0-S1	SunSoft		1	902	1	902	$2,846
Tuxedo /T Version 4.2.2				4	1,200	6	7,200	5,400
						Subtotal	$12,716	$33,446
Terminal Connectivity								
AccuLAN 10Mbps 24-Port Ethernet Hub (10% spares)	AL-HUB24L			1	282	264	74,448	0
AsanteFast 100Mbps 12-port Stackable Hub (2 spares)	990035200			1	1,372	3	4,116	0
						Subtotal	$78,564	$0
						Total	$494,441	$93,545

Notes: 1 - The Gibraltar Computer Group; 2 - Oracle Corp.; 3 - DPT; 4 - BEA Systems

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 stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.

Five-Year Cost of Ownership:	$587,986
tpmC Rating:	6679.50
$ / tpmC:	$88.00

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification, Revision 3.2.2, dated August 27, 1996, for measurements on the IBM PC Server 704 system with an Intel** Pentium** Pro 200MHz processor.

The software used on the IBM PC Server 704 system includes the SunSoft** Solaris** Version 2.5.1 operating system and Oracle7** database manager and the Tuxedo /T Version 4.2.2 transaction manager.

Two standard metrics, transactions per minute-C (tpmC) and price per tpmC ($/tpmC), are reported as required by the TPC Benchmark C Standard Specification. The independent auditor's report, prepared by Richard Gimarc of Performance Metrics, Inc., is contained in Section 9 of this report.

Standard System Summary

Company Name	System Name	Database Software	Operating System Software
IBM Corporation Oracle Corporation SunSoft	IBM PC Server 704	Oracle7 v 7.3.3	SunSoft Solaris 2.5.1
Hardware Available: Now Oracle7 v. 7.3.3 Available: May 30, 1997 Solaris 2.5.1 Available: Now			

Total System Cost	TPC-C Throughput	Price/Performance
Hardware, Software and 5-Year Maintenance	System's Sustained Maximum Throughput Expressed As the tpmC	Total System Cost/tpmC
$587,986	6,679.50	$88.00/tpmC

Numerical Quantities Summary for the IBM PC Server 704

MQTh, computed Maximum Qualified Throughput:			6679.50 tpmC
% throughput difference, reported and reproducibility runs:			0.1 %
Response Times (in seconds)	**90th Percentile**	**Average**	**Maximum**
New-Order	2.60	1.54	8.37
Payment	1.00	0.80	289.17
Order-Status	1.60	1.06	7.08
Delivery (Interactive)	2.00	1.10	9.55
Delivery (Deferred)	6.00	2.33	11.00
Stock Level	8.00	4.08	126.34
Menu	0.40	0.33	0.60
Response time delay added for emulated components is NONE.			
Transaction Mix, in percent of total transactions		**Total Occurrences**	**Percent**
New-Order		146,949	44.83 %
Payment		141,019	43.02 %
Order-Status		13,211	4.03 %
Delivery		13,422	4.09 %
Stock-Level		13,221	4.03 %
Keying/Think Times (in seconds)	**Minimum**	**Average**	**Maximum**
New-Order	18.00 / 0.00	18.00 / 12.20	18.10 / 122.00
Payment	3.00 / 0.00	3.00 / 12.20	3.10 / 122.00
Order Status	2.00 / 0.00	2.00 / 10.20	2.10 / 102.00
Delivery	2.00 / 0.00	2.00 / 5.20	2.10 / 51.10
Stock-Level	2.00 / 0.00	2.00 / 5.20	2.10 / 48.30
Test Duration			
Ramp-up time			27 mimutes
Measurement interval			22 minutes
Number of checkpoints in measurement interval			1
Checkpoint interval			22 minutes
Number of transactions (all types) completed in measurement interval			327,822

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 3.2.2 on December 11, 1996.

This is the full disclosure report of the benchmark results as required by the TPC Benchmark C Standard Specification to document the measurements made with the IBM PC Server 704 system.

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

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

This benchmark defines four online 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 substitute for a specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

The performance metric reported by TPC-C is a “business throughput” measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of 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 (tmpC). 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.

General Items

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 IBM PC Server 704 application code used for the TPC Benchmark C transactions. Appendix D contains the terminal functions and layouts.

Benchmark Sponsors

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

This benchmark was sponsored by International Business Machines Corporation, Oracle Corporation and SunSoft Engineering, a division of Sun Microsystems.

Parameter Settings

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

- *Database tuning options*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and application configuration parameters.*
- *Compilation and linkage options and run-time optimizations used to create/install applications, OS, and/or databases.*

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

Appendix B contains the system, database and application parameters that were changed from their default values and used in the TPC Benchmark C tests.

Configuration Diagrams

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

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

The configuration diagrams for the tested and priced systems are provided on the following pages.

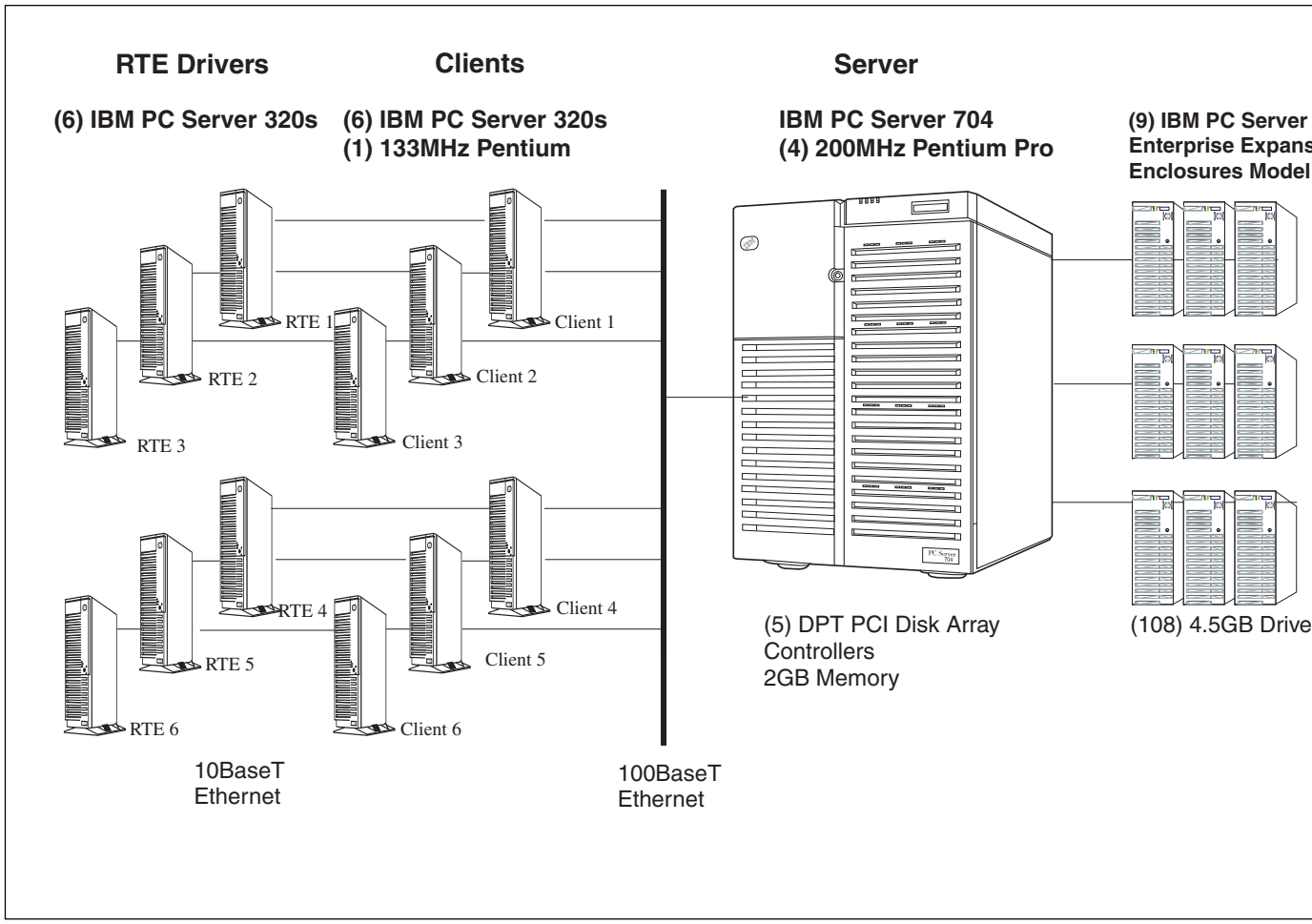
The Remote Terminal Emulator (RTE) used in the benchmarked configuration is a SunSoft internally developed tool. The RTE was used to emulate the workstations and the Ethernet hubs.

The benchmarked configuration also used IBM PC Server 320 systems as clients, which executed the terminal I/O and submitted transactions to the Tuxedo /T transaction manager running on the clients.

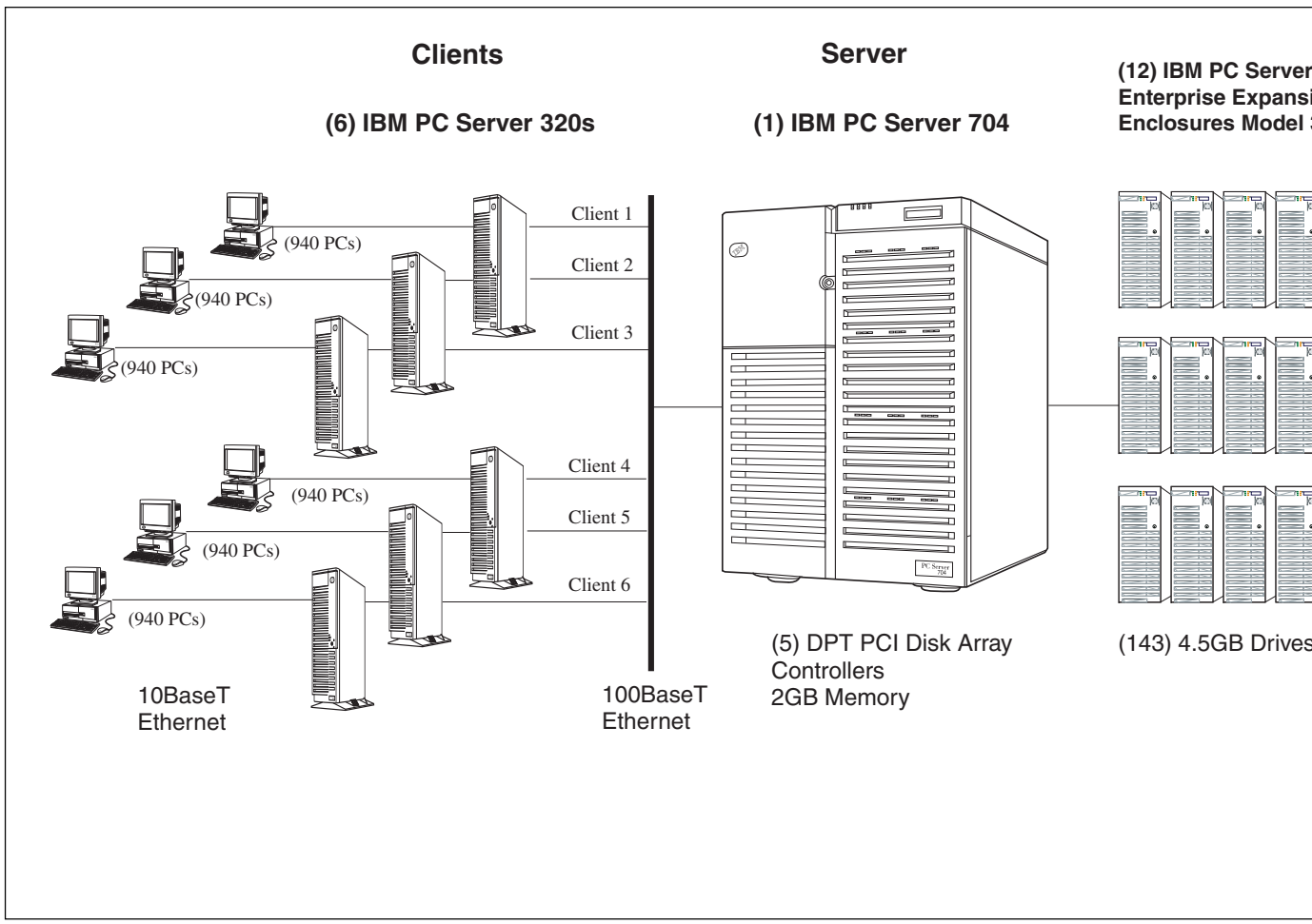
Oracle7 is the DBMS executing on the server.

Appendix D contains a listing of the RTE scripts and inputs used in the benchmark testing.

IBM PC Server 704 Benchmarked Configuration



IBM PC Server 704 Priced Configuration



1 Clause 1: Related Items

Table Definitions

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

Appendix C contains the table creation, index creation, and the data generation programs used to load the database.

Physical Organization of the Database

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

Physical space was allocated to Oracle7 on the server disks as detailed in Figure 4-2. The size of the space segments on each disk was calculated to provide even distribution of data across the disk.

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. Furthermore, any restriction in the SUT database implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows.

There were no restrictions on insert/delete operations to any of the tables. The space required for an additional 5 percent of the initial table cardinality was allocated to Oracle7 and priced as static space.

Horizontal or Vertical Partitioning

While there are few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark (see Clause 1.6), any such partitioning must be disclosed.

Partitioning was not used for any of the measurements reported in this full disclosure report.

Replication Tables, Duplication or Additions

Replication tables, if used, must be disclosed (see Clause 1.4.6). Additional and/or duplicated attributes in any table must be disclosed, along with a statement on the impact on performance (see Clause 1.4.7).

No replications, duplications or additional attributes were used in this benchmark.

2 Clause 2: Related Items

Random Number Generation

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

The Random Number Generator used was the one that appeared in the article titled “Random Number Generators: Good Ones Are Hard to Find” in the communications of the ACM, October 1988, Vol. 31, No. 10. The properties of this random number generator are well-known and are documented in the article as producing a uniformly distributed pseudo-random sequence. To generate a random number, the driver programs first used a seed based on the host address, current time and the process-id of the respective session. This guarantees that each emulated user on all the RTE machines is mathematically independent of others.

Input/Output Screen Layout

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

The screen layouts correspond exactly to the layouts presented in Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3 and 2.8.3 of the TPC Benchmark C Standard Specification, Revision 3.2.2.

Verification of 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 disclosed. Although not specifically priced, the type and model of the terminals used must be disclosed and commercially available (including supporting software and maintenance).

The auditor verified terminal features by direct experimentation.

Presentation Manager or Intelligent Terminal

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

The terminals emulated in the priced configuration are IBM PC desktop computer systems. All processing of the input/output screens was handled by the IBM PC Server 320 clients. All data manipulation was handled by the IBM PC Server 704.

Transaction Statistics

Table 2-1 lists the numerical quantities required by Clauses 8.1.3.5 to 8.1.3.11.

Table 2-1. Transaction Statistics for the IBM PC Server 704

New Order	
Percentage of home order lines in New-Order	98.997 %
Percentage of remote order lines in New-Order	1.003 %
Percentage of New-Order transactions rolled back	1.026 %
Number of Items per order	9.996
Payment	
Percentage of home Payment transactions	84.917 %
Percentage of remote Payment transactions	15.083 %
Non-Primary Key Access	
Percentage of Payment transactions using C_LAST	60.281 %
Percentage of Order-Status transactions using C_LAST	59.814 %
Delivery	
Percentage of Delivery transactions skipped	0 %
Transaction Mix	
New-Order	44.83 %
Payment	43.02 %
Order-Status	4.03 %
Stock-Level	4.09 %
Delivery	4.03 %

Queuing Mechanism of Delivery

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

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

3 Clause 3: Related Items

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

The ACID Properties

It is the intent of this section to informally define the ACID properties and to specify a series of tests that must be performed to demonstrate that these properties are met. Comment: These tests are intended to demonstrate that the ACID principles are supported by the SUT and enabled during the performance measurement interval. They are not intended to be an exhaustive quality assurance test.

Atomicity Requirements

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

All ACID tests were conducted according to specification. The Atomicity, Consistency, Isolation and Durability tests were performed on the IBM PC Server 704.

Atomicity of Completed Transactions

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

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

1. The balance was retrieved from the CUSTOMER table for a random Customer, District and Warehouse, giving BALANCE_1.
2. The Payment transaction was executed for the Customer, District and Warehouse used in step 1.
3. The balance was retrieved again for the Customer used in step 1 and step 2, giving BALANCE_2. It was verified that BALANCE_1 was greater than BALANCE_2 by AMT.

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 Payment application code was changed to execute a rollback of the transaction instead of performing the commit.
2. Using the balance, BALANCE_2, from the CUSTOMER table retrieved for the completed transaction, the Payment transaction was executed for the Customer, District and Warehouse used in step 1 of section 3.1.1, using a payment amount (AMT) of 410.00. The transaction rolled back due to the change in the application code from step 1.
3. The balance was retrieved again for the Customer used for step 2, giving BALANCE_3. It was verified that BALANCE_2 was equal to BALANCE_3.

Consistency Requirements

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

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 database was always in a consistent state both initially and after a measurement.

```
Select D_W_ID, 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;
```

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 that have no outstanding new orders (i.e., the number of rows is zero).

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

```
Select D_W_ID, D_ID, D_NEXT_O_ID - 1, MAX(O_ID)
        from DISTRICT, ORDERS
        where D_ID = O_D_ID and D_W_ID = O_W_ID
        group by D_W_ID, D_ID, D_NEXT_O_ID
        order by D_W_ID, D_ID;

Select D_W_ID, D_ID, D_NEXT_O_ID - 1, MAX(NO_O_ID)
        from DISTRICT, NEW_ORDER
        where D_ID = NO_D_ID and D_W_ID = NO_W_ID
        group by D_W_ID, D_ID, D_NEXT_O_ID
        order by D_W_ID, D_ID;
```

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 that have no outstanding new orders (i.e., the number of rows is zero).

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

```
Select COUNT(*), MAX(NO_O_ID) - MIN(NO_O_ID) + 1
        from NEW_ORDER, DISTRICT
        where NO_W_ID = D_W_ID and NO_D_ID = D_ID
        group by NO_W_ID, NO_D_ID;
```

Consistency Condition 4

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

- *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 were executed before and after a measurement to show that the database was always in a consistent state both initially and after a measurement.

- Select O_ID,O_D_ID,SUM(O_OL_CNT) from ORDERS group by O_W_ID and O_D_ID
- Select OL_W_ID,OL_D_ID,COUNT(*) from ORDER_LINE group by OL_W_ID and OL_D_ID

Consistency Tests

Consistency is the property of the application that requires any execution of a database transaction to take the database from one consistent state to another, assuming that the database is initially in a consistent state. Verify that the database 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 consistency conditions defined in 3.3.2.1 through 3.3.2.4 were tested using a shell script to issue queries to the database. All queries showed that the database was in a consistent state.

After executing transactions at full load for approximately 60 minutes, the shell script was executed again. All queries showed that the database was still in a consistent state.

Isolation Requirements

Isolation can be defined in terms of phenomena that can occur during the execution of concurrent database transactions... . Sufficient conditions must be enabled at either the system or the application level to ensure that the required isolation defined in Clause 3.4.1 is obtained.

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. An Order status transaction T0 was executed for a randomly selected customer, and the order returned was recorded. Transaction T0 was committed.
2. A new-order transaction T1 was started for the same customer used in T0. T1 was stopped immediately prior to commit.
3. An order-status transaction T2 was started for the same customer used in T1. Transaction T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 completed and was committed.
5. An order-status transaction T3 was started for the same customer used in T1. T3 returned the order inserted by T1.

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

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 transaction:

1. An Order status transaction T0 was executed for a randomly selected customer, and the order returned was recorded. Transaction T0 was committed.
2. A new-order transaction T1 with an invalid item was started for the same customer used in T0. Transaction T1 was stopped prior to rollback
3. An order-status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. Transaction T2 returned the same order that T0 had returned.
4. T1 was rolled back.
5. An order-status transaction T3 was started for the same customer used in T1. T3 returned the same order that T0 returned.

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. The D_NEXT_O_ID of a randomly selected district was retrieved.
2. A new-order transaction T1 was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to commit.
3. Another new-order transaction was started for the same customer used in T1. Transaction T2 waited.
4. T1 completed. T2 completed and was committed.
5. The order number returned by T1 was the same as the D_NEXT_O_ID retrieved in step 1. The order number returned by T2 was one greater than the order number returned by T1.
6. The D_NEXT_O_ID of the same district was retrieved again. It had been incremented by 2 (it was 1 greater than the order number returned by T2).

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

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

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. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C_BALANCE of the customer found in step 1 is retrieved.A delivery transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the rollback of the database transaction corresponding to the district used in step 1.
3. A payment transaction T2 was started for the same customer found in step 1. Transaction T2 waited.
4. T1 was allowed to roll back. T2 completed and was committed.
5. The C_BALANCE of the customer found in step 1 was retrieved again. The C_BALANCE reflected the results of only transaction T2.

Isolation Test 7

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

The following steps were performed to successfully conduct this test:

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

Case A was followed in this isolation test.

Isolation Test 8

This test demonstrates isolation for phantom protection between a delivery and a new-order transaction.

The following steps were performed to successfully conduct this test:

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

Isolation Test 9

This test demonstrates isolation for phantom protection between an order-status and a new-order transaction.

The following steps were performed to successfully conduct this test:

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

Durability Requirements

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

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

This test was conducted on a fully scaled database. The following steps were successfully performed to pass the Durability test of failure of a disk unit with database tables:

1. The contents of the disk containing the orders 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 12 minutes.
4. The disk containing the orders table was corrupted using the Unix DD command. Errors were observed in the Tuxedo logs and the RTE logs.
5. The run was aborted by the RTE and the database was aborted.
6. The orders disk was restored from the backup copy.
7. Oracle7 was restarted and its transaction log was used roll forward the transactions that had completed but weren't written to disk before the failure.
8. Step 2 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.

This test, which was conducted on a fully scaled database, combines the system and logs tests. The following steps were successfully performed:

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 10 minutes.
3. The mirror disk containing the Oracle7 transaction log data was powered off. Since this disk was mirrored, Oracle7 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 off the Power, thereby removing power from the system and the memory.
6. The system was powered on again and rebooted.
7. Step 1 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.
8. Consistency condition 3 was verified.

4 Clause 4: Related Items

Cardinality of Database 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.

Table 4-1 provides the TPC-C Benchmark C defined tables and the number of rows for each table as they were built initially.

Table 4-1. Initial Cardinality of Tables

Table Name	Rows
Warehouse	564
District	5,640
Item	100,000
History	16,920,000
Orders	16,920,000
Customer	16,920,000
New_order	5,076,000
Order_line	169,168,006
Stock	56,400,000

Distribution of Tables and Logs

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

Figure 4-2 depicts the database configuration of the tested system to meet the 8-hour steady state requirement.

The formatted data capacity of the IBM 4.5GB drives was 4.3GB (where GB equals 1024*1024*1024 bytes). All disk controllers used in the test were Distributed Processing Technology (DPT) 3334 three-channel controllers. The cacheing capability of the disk controllers was disabled for all the drives in the configuration; hence, no UPS was required.

A total of five DPT 3334 three-channel controllers were used. Nine of the available 15 channels were used.

A total of 108 IBM 4.5GB SCSI-2 Fast/Wide drives were used in the configuration. The 108 drives were housed in the nine IBM PC Server Enterprise Expansion Enclosures.

A total of 11 logical volumes were used in the configuration.

Controller #0: one of three SCSI channels used

- c0t0: System Boot Disk, Swap
- c0t1: Mirrored Log Disk
- c0t3: A RAID-0 stripe of 9 IBM drives

Controllers #1, #2, #4, #5:

- Two of three SCSI channels used
- Two RAIDs of 12 drives each were created on the controllers:
  - RAID t0 consists of Channel 0 targets 0-5, Channel 2 targets 0-5
  - RAID t8 consists of Channel 0 targets 8-13, Channel 2 targets 8-13

Controller #3 is the on-board Adaptec controller, which was used to drive the CD-ROM and tape drives.

Symbolic links were used to map database files to raw partitions. Figure 4-2 shows the mapping sorted by controller and slice number.

Figure 4-2. Data Distribution for the Benchmarked Configuration

Symbolic Name	Raw Device Name	Size (in MB)
/dev/orac/log1	/dev/rdsk/c0t1d0s0	2024
/dev/orac/log2	/dev/rdsk/c0t1d0s1	2024
/dev/orac/iord2	/dev/rdsk/c0t3d0s0	1700
/dev/orac/icust2	/dev/rdsk/c0t3d0s1	1600
/dev/orac/iordl3	/dev/rdsk/c0t3d0s3	1700
/dev/orac/iord1	/dev/rdsk/c0t3d0s4	500
/dev/orac/stk2	/dev/rdsk/c1t0d0s0	1700
/dev/orac/stk10	/dev/rdsk/c1t0d0s1	1700
/dev/orac/cust2	/dev/rdsk/c1t0d0s3	1700
/dev/orac/cust10	/dev/rdsk/c1t0d0s4	1700
/dev/orac/ordl5	/dev/rdsk/c1t0d0s6	2000
/dev/orac/ware1	/dev/rdsk/c1t0d0s7	20
/dev/orac/icust1	/dev/rdsk/c1t0d0s9	1000
/dev/orac/stk6	/dev/rdsk/c1t8d0s0	1700
/dev/orac/stk14	/dev/rdsk/c1t8d0s1	1700
/dev/orac/cust6	/dev/rdsk/c1t8d0s3	1700

/dev/orac/iordl2	/dev/rdsk/c1t8d0s4	1700
/dev/orac/ordl1	/dev/rdsk/c1t8d0s5	2000
/dev/orac/ordl9	/dev/rdsk/c1t8d0s6	2000
/dev/orac/temp1	/dev/rdsk/c1t8d0s7	1999
/dev/orac/ordl12	/dev/rdsk/c1t8d0s9	2000
/dev/orac/stk3	/dev/rdsk/c2t0d0s0	1700
/dev/orac/stk11	/dev/rdsk/c2t0d0s1	1700
/dev/orac/cust3	/dev/rdsk/c2t0d0s3	1700
/dev/orac/cust11	/dev/rdsk/c2t0d0s4	1700
/dev/orac/iord1	/dev/rdsk/c2t0d0s5	1400
/dev/orac/ordl6	/dev/rdsk/c2t0d0s6	2000
/dev/orac/hist1	/dev/rdsk/c2t0d0s7	1900
/dev/orac/istk1	/dev/rdsk/c2t0d0s9	1700
/dev/orac/stk7	/dev/rdsk/c2t8d0s0	1700
/dev/orac/stk15	/dev/rdsk/c2t8d0s1	1700
/dev/orac/cust7	/dev/rdsk/c2t8d0s3	1700
/dev/orac/ordl10	/dev/rdsk/c2t8d0s4	2000
/dev/orac/ordl2	/dev/rdsk/c2t8d0s5	2000
/dev/orac/temp2	/dev/rdsk/c2t8d0s7	1999
/dev/orac/stk1	/dev/rdsk/c4t0d0s0	1700
/dev/orac/stk9	/dev/rdsk/c4t0d0s1	1700
/dev/orac/cust1	/dev/rdsk/c4t0d0s3	1700
/dev/orac/cust9	/dev/rdsk/c4t0d0s4	1700
/dev/orac/ordl4	/dev/rdsk/c4t0d0s5	2000
/dev/orac/sys1	/dev/rdsk/c4t0d0s6	300
/dev/orac/temp4	/dev/rdsk/c4t0d0s7	1999
/dev/orac/stk5	/dev/rdsk/c4t8d0s0	1700
/dev/orac/stk13	/dev/rdsk/c4t8d0s1	1700
/dev/orac/cust5	/dev/rdsk/c4t8d0s3	1700
/dev/orac/ordl8	/dev/rdsk/c4t8d0s4	2000
/dev/orac/ord1	/dev/rdsk/c4t8d0s5	1600
/dev/orac/nord1	/dev/rdsk/c4t8d0s6	500
/dev/orac/temp3	/dev/rdsk/c4t8d0s10	1999
/dev/orac/stk4	/dev/rdsk/c5t0d0s0	1700
/dev/orac/stk12	/dev/rdsk/c5t0d0s1	1700
/dev/orac/cust4	/dev/rdsk/c5t0d0s3	1700
/dev/orac/cust12	/dev/rdsk/c5t0d0s4	1700
/dev/orac/ordl7	/dev/rdsk/c5t0d0s5	2000
/dev/orac/item1	/dev/rdsk/c5t0d0s6	20
/dev/orac/stk8	/dev/rdsk/c5t8d0s0	1700
/dev/orac/stk16	/dev/rdsk/c5t8d0s1	1700

/dev/orac/cust8	/dev/rdsk/c5t8d0s3	1700
/dev/orac/iordl4	/dev/rdsk/c5t8d0s4	1700
/dev/orac/ordl3	/dev/rdsk/c5t8d0s5	2000
/dev/orac/roll1	/dev/rdsk/c5t8d0s6	400
/dev/orac/iordl1	/dev/rdsk/c5t8d0s7	1700
/dev/orac/ordl11	/dev/rdsk/c5t8d0s9	2000

Database Model Implemented

A statement must be provided that describes:

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

Oracle7 is a relational database. Embedded SQL was used with C language to implement the TPC-C benchmark.

Partitions/Replications Mapping

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

Neither horizontal nor vertical partitioning was implemented for these TPC-C tests.

180-Day Space Requirement

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

tpmC: 6,679.50

Warehouses: 564

SEGMENT	TYPE	TSPACE	BLOCKS	FIVE_PCT	DAILY_GROW	TOTAL
CUSTOMER	TABLE	CUST	8,460,006	423,000		8,883,006
DISTRICT	TABLE	WARE	5,644	282	0	5,926
HISTORY	TABLE	HIST	477,176	0	90,420	567,596
ICUSTOMER	INDEX	ICUST1	197,870	9,894	0	207,764
ICUSTOMER2	INDEX	ICUST2	449,374	22,469	0	471,843
IDISTRICT	INDEX	WARE	1,000	50	0	1,050
IITEM	INDEX	ITEMS	1,000	50	0	1,050
INEW_ORDER	INDEX	INORD	62,801	3,140	0	65,941
IORDERS	INDEX	IORD1	195,036	9,752	0	204,788
IORDERS2	INDEX	IORD2	330,839	16,542	0	347,381
IORDER_LINE	INDEX	IORDL	2,272,444	113,622	0	2,386,066
ISTOCK	INDEX	ISTK	586,522	29,326	0	615,848
ITEM	TABLE	ITEMS	6,667	333	0	7,000
IWAREHOUSE	INDEX	WARE	100	5	0	105
NEW_ORDER	TABLE	NORD	44,832	2,242	0	47,074
ORDERS	TABLE	ORD	343,866	0	65,159	409,025
ORDER_LINE	TABLE	ORDL	6,317,527	0	1,197,104	7,514,631
ROLL_SEG	SYS	ROLL	204,288	0	0	204,288
STOCK	TABLE	STOCKS	11,280,002	564,000	0	11,844,002
SYSTEM	SYS	SYSTEM	153,088	0	0	153,088
WAREHOUSE	TABLE	WARE	564	28	0	592
TOTAL			31,390,646	1,194,735	1,352,683	33,938,064

Dynamic space:	7,138,569	
Static space	25,446,812	
Free space	1,352,683	
Daily growth:	1,352,683	
Daily spread:	0	(Oracle7 may be configured so the daily spread equals zero)
180-day space (blks)	268,929,752	
Block size (bytes)	2,048	
180-day (MB)	**525,253**	

Space Usage

Usage	Size
180-day space	525,253
Logs (with/mirroring)	80,438
Root, swap	3,500
Total:	**609,191**

Allocated Disks

Number	Capacity (MB)	Total (MB)
143	4,300	614,900

Log Computation

Log space was calculated by monitoring a full benchmark run. The system was run under full load and included a full checkpoint. I/O rates to the log volume were monitored using the Solaris iostat utility across the entire measurement interval.

Bytes/I/O	53,101	Data extracted from monitored test run
Writes/Second	27.2	
Bytes/Minute	86,660,832	Data was collected during measurement interval only
tpmC:	6588.36	Actual measured result from log space test run
Log Bytes/tpmC	13,153.63	
8-hour log (MB):	**40,219**	

5 Clause 5: Related Items

Response Times

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

Table 5-1 provides the response times for each of the transaction types and the menu for the measured system.

Table 5-1. IBM PC Server 704 Response Times in Seconds

Transaction Type	90th Percentile	Average	Maximum
New Order	2.60	1.54	8.37
Payment	1.00	0.80	289.17
Order Status	1.60	1.06	7.08
Delivery (interactive)	2.00	1.10	9.55
Delivery (deferred)	6.00	2.33	11.00
Stock Level	8.00	4.08	126.34
Menu	0.40	0.33	0.60

The TPC-C requirements for the average response time and the 90th percentile were met.

Keying/Think Times

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

Table 5-2 lists the keying/think times for the measured system.

Table 5-2. IBM PC Server 704 Keying/Think Times

Transaction Type	Minimum	Average	Maximum
New Order	18.00 / 0.00	18.00 / 12.20	18.10 / 122.00
Payment	3.00 / 0.00	3.00 / 12.20	3.10 / 122.00
Order Status	2.00 / 0.00	2.00 / 10.20	2.10 / 102.00
Delivery	2.00 / 0.00	2.00 / 5.20	2.10 / 51.10
Stock Level	2.00 / 0.00	2.00 / 5.20	2.10 / 48.30

Response Time Frequency Distribution

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

Figure 5-1. New-Order Transaction - Response Time Frequency Distribution

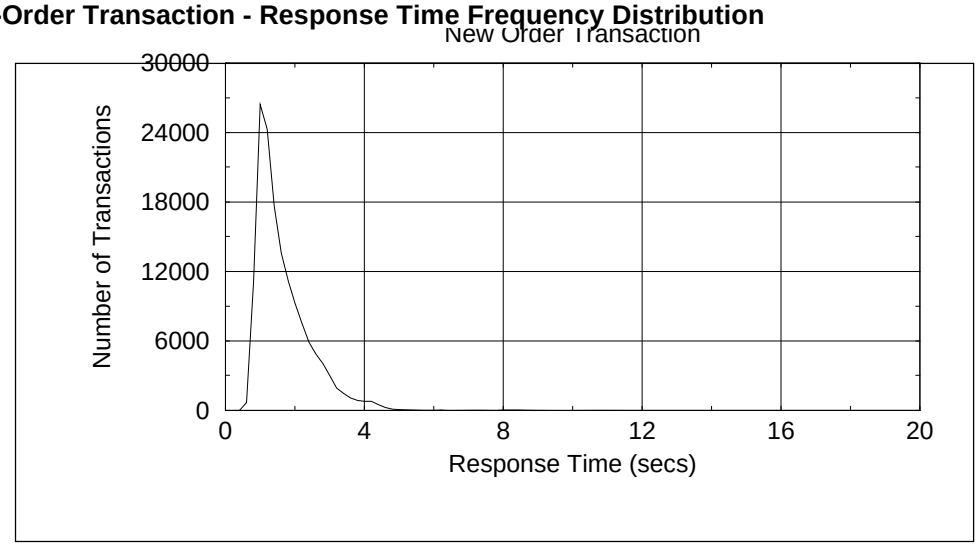


Figure 5-2. Payment Transaction - Response Time Frequency Distribution

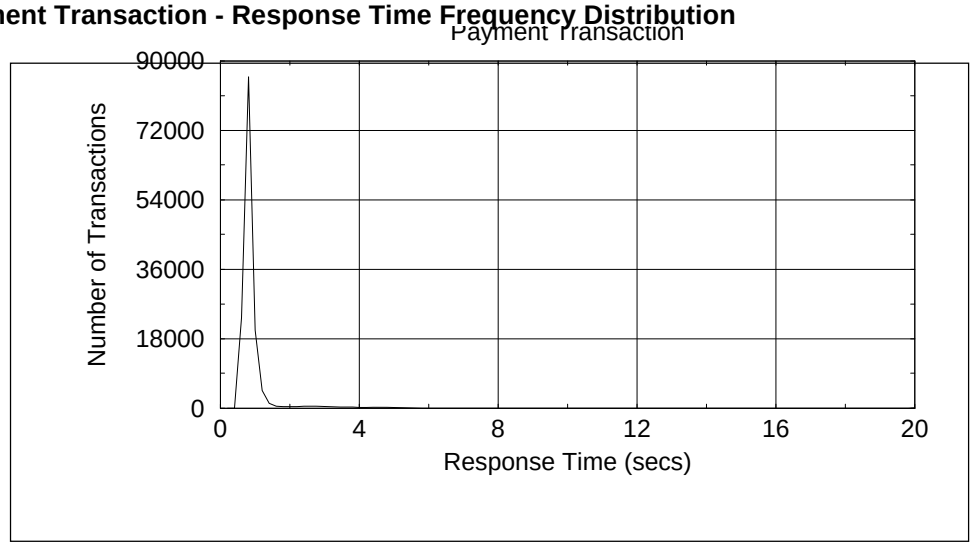


Figure 5-3. Order-Status Transaction - Response Time Frequency Distribution

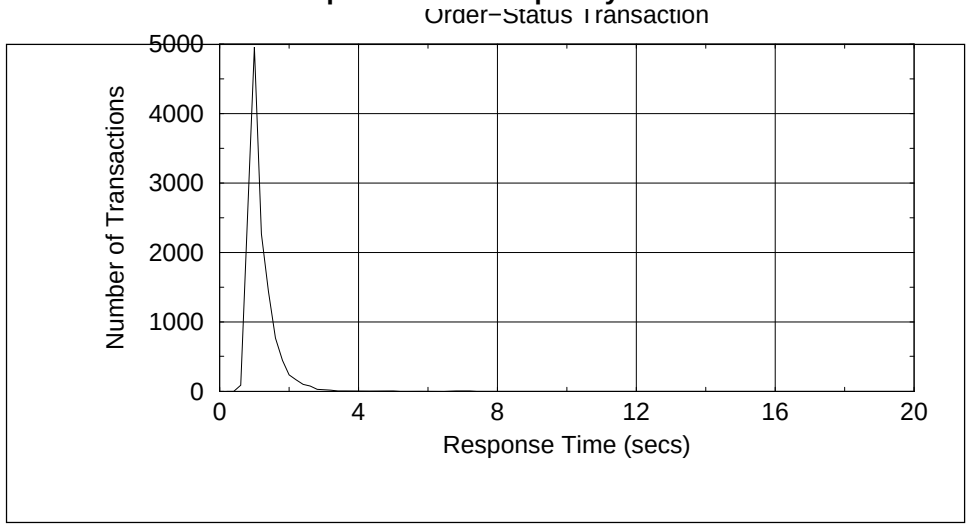


Figure 5-4. Delivery Transaction - Response Time Frequency Distribution

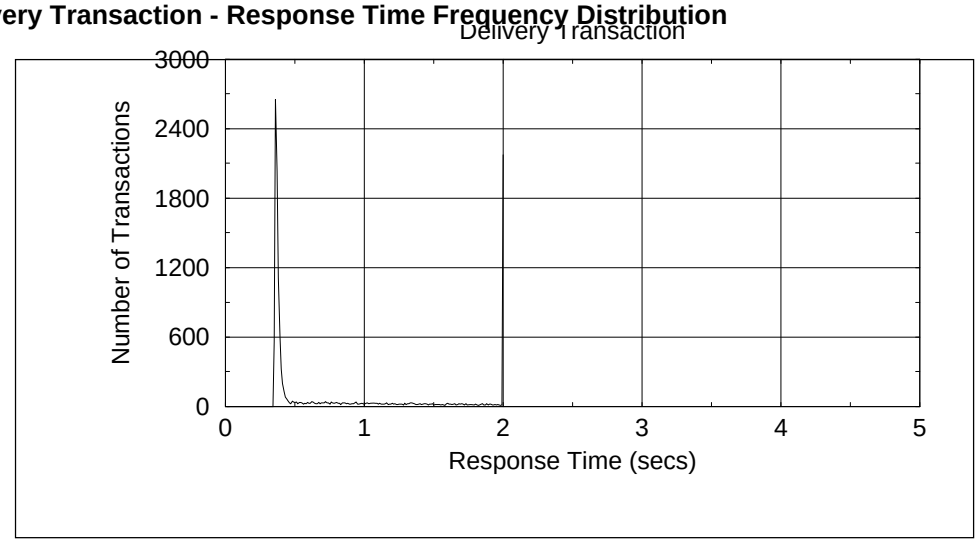
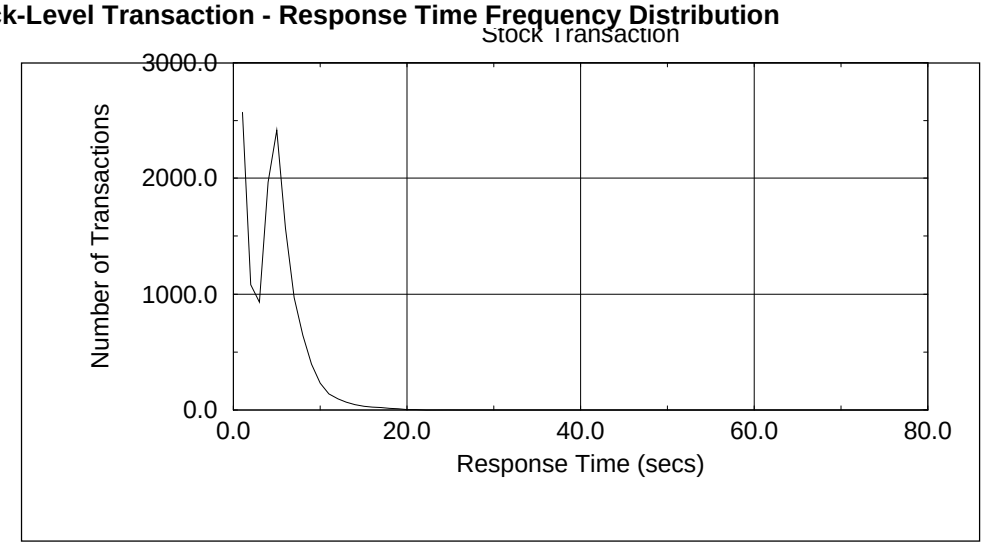


Figure 5-5. Stock-Level Transaction - Response Time Frequency Distribution



Performance Curve for Response Time vs. Throughput

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

Figure 5-6. New-Order Response Time vs. Throughput

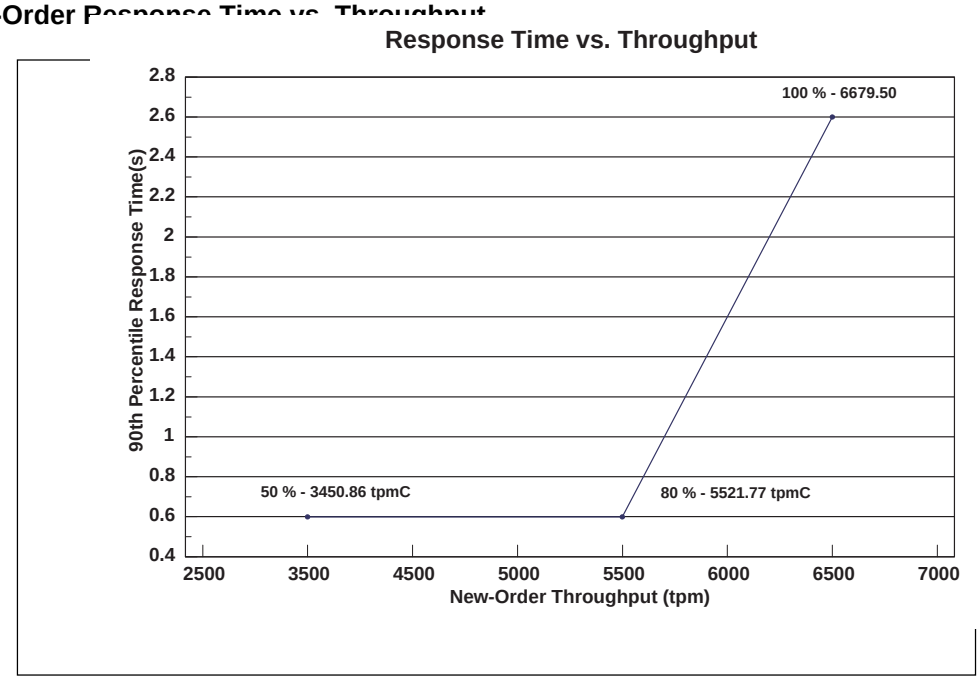
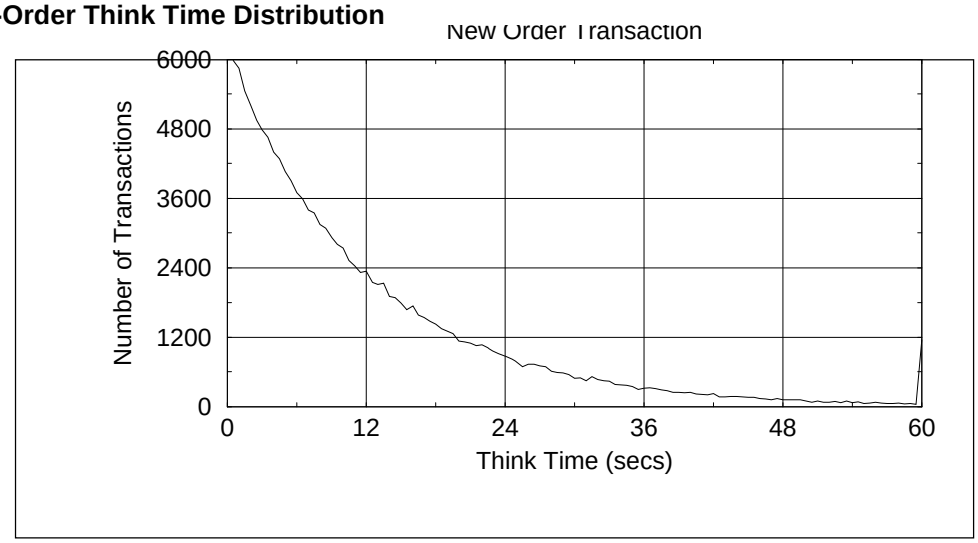


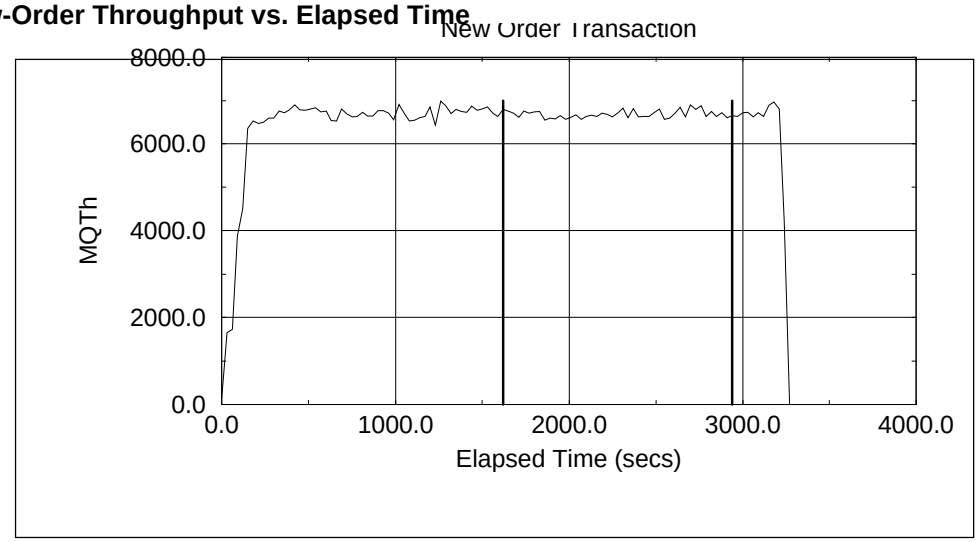
Figure 5-7. New-Order Think Time Distribution



Throughput vs. Elapsed Time

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

Figure 5-8. New-Order Throughput vs. Elapsed Time



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 ramp-up. See the Numerical Quantities Summary on page 9 for the ramp-up times for the measured system. Figure 5-8 shows that the system was in steady state at the beginning of the measurement interval.

Work Performed during Steady State

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

Transaction Flow

The steps described in this section were executed for each of the TPC Benchmark C transaction types.

Tuxedo /T was used as a transaction manager. Each transaction was divided into three programs: a front-end program, which handled all screen I/O; a database client program, which connected to the database and served as a Tuxedo /T server (a back-end program); and a database server program, which handled all database operations at the SUT. Both the front-end and back-end programs ran on the client system. The front-end program communicated with the database client program through Tuxedo /T messages. The database client program communicated with the server system over Ethernet using SQL* Net calls. Besides calling Tuxedo /T functions for user connection and message communication, all other functions were transparent to the application code. Tuxedo /T routes the transaction and balances the load according to the options defined in the Tuxedo /T configuration file listed in Appendix B. The transaction flow is described below:

- When Tuxedo /T boots up, it creates one or more server processes for each transaction. Several server processes were defined in the Tuxedo /T configuration file.
- Each TPC-C user invokes the TPC-C main (front-end) program.
- The TPC-C main program connects to Tuxedo /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 /T function call. Each TPC-C back-end program has a “service name,” which is specified whenever the TPC-C main program requests a Tuxedo /T service. Tuxedo /T routes that message according to the service name and the information defined in the Tuxedo /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 /T message.
- Once the transaction is committed, the TPC-C back-end server programs loads the message buffer with the transaction output and returns control to the Tuxedo /T manager.
- Tuxedo /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.

Database Transaction

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

resulted in many transactions' redo log data being written to disk (group commit).

Checkpoints

During an Oracle7 checkpoint, all modified blocks in the shared buffer cache, which had not been written to disk since the last checkpoint, are written to disk.

Oracle7 performs a checkpoint for the following conditions:

- A redo log switch occurs.
- The amount of data written to a redo log reaches the log_checkpoint interval.
- The amount of time since the last checkpoint reaches the log_checkpoint_timeout.

During the benchmark measurement, a log switch was performed 10-1/2 minutes into ramp-up. After the initial checkpoint, a log switch was performed every 22 minutes.

Reproducibility

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

A repeatability measurement was taken on the IBM PC Server 704 for the same length of time as the measured run. The repeatability measurement was 6679.69tpmC.

Measurement Interval

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

A 22-minute Measurement Interval was used for the IBM PC Server 704.

Method of Regulation of Transaction Mix

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

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

Percentage of Total Mix

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

See section 2, Table 2-1.

Percentage of New-Order Transactions Rolled Back

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

See section 2, Table 2-1.

Average Number of Order Lines

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

See section 2, Table 2-1.

Percentage of Remote Order Lines

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

See section 2, Table 2-1.

Percentage of Remote Payment Transactions

The percentage of remote Payment transactions must be disclosed.

See section 2, Table 2-1.

Percentage of Customer Selections

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

See section 2, Table 2-1.

Percentage of Delivery Transactions

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

See section 2, Table 2-1.

Number of Checkpoints

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

The measurement period contained one checkpoint. The checkpoint interval was 22-minutes. The checkpoint was triggered by forcing a log switch during the measurement period.

6 Clause 6: Related Items

Description of RTE

The RTE input parameters, code fragments, functions, etc., used to generate each transaction input field must be disclosed.

SunSoft proprietary remote terminal emulation (RTE) software was used to simulate terminal users, generate random data, and record response times. Appendix D contains the RTE scripts used in the testing.

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. The results of the test described in Clause 6.6.3.4 must be disclosed.

In the benchmarked configuration, the RTE communicates with the client system over 10BaseT Ethernet. Six IBM PC Server 320 systems emulate a network of 5,640 IBM PC workstations. The communication mechanism used in the benchmark and priced configurations are the same. In the benchmarked and priced configurations, one Ethernet LAN was used to connect to the RTE to the client system.

The RTE was configured with the following terminal delays:

Transaction	Delay (in seconds)
New Order	0.21
Payment	0.32
Order Status	0.23
Delivery	0.25
Stock Level	0.10

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 (Mbps).

Operator Intervention

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

The IBM PC Server 704 configuration reported does not require any operator intervention to sustain the reported throughput during the eight-hour period.

7 Clause 7: Related Items

Hardware and Software Components

A detailed list of the hardware and software used in the priced system must be reported. Each item must have a 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 programs for the priced configuration is provided in the Executive Summary at the front of this report. The prices for all products and features provided by IBM Corporation are available the same day as product or feature availability.

Products from sources outside IBM include:

- Oracle7 v. 7.3.3 prices include SQL* Net. Both software license and support prices are quoted by Oracle Corporation.
- The Gibraltar Computer Group provided pricing for the AccuLAN and Asante Ethernet hubs. These hubs have a five-year return-to-factory warranty.
- The Gibraltar Computer Group also provided pricing for the Solaris Workgroup Server Version 2.5.1 (includes patch 103641-01).
- Tuxedo /T Version 4.2.2 prices are provided by BEA Systems, Inc. Service for five years is 15 percent annually of the list price.
- Distributed Processing Technology provided prices for the DPT disk controller and DPT Dual-Channel Expansion Module.

Five-Year Cost of System Configuration

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

See the Executive Summary at the front of this report for the 5-year price of the system configuration.

Availability, Throughput and Price Performance

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.

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

- Maximum Qualified Throughput: 6679.50 tpmC
- Price per tpmC: $88.00/tpmC
- Five-year cost of ownership: $587,986
- Available:
  - All hardware and the Solaris software are available now.
  - Oracle7 v. 7.3.3 will be available May 30, 1997.

8 Clause 9: Related Items

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

Auditor's Report

This implementation of the TPC-C benchmark was audited by Richard Gimarc of Performance Metrics, Inc.

Richard Gimarc (916-635-2822)

Performance Metrics, Inc.
2229 Benita Drive, Suite 101
Rancho Cordova, CA 95670

The auditor's attestation letter is provided in this section.

Availability of the Full Disclosure Report

The Full Disclosure Report must be readily available to the public at a reasonable charge, similar to the charges for similar documents by the test sponsor. The report must be made available when results are made public. In order to use the phrase "TPC Benchmark™C," the Full Disclosure Report must have been submitted to the TPC Administrator as well as written permission obtained to distribute same.

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

Transaction Processing Performance Council
c/o Shanley Public Relations
777 North First Street, Suite 600
San Jose, CA 95112-6311

PERFORMANCE METRICS INC.
TPC Certified Auditors

December 17, 1996

William D. Hall
PC Server Performance
IBM Corporation
3039 Cornwallis Road
Research Triangle Park, NC 27709

Karl Haas
Director, Open Systems Performance
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065

Abe Ellenberg
Manager, Platform Engineering
SunSoft
6601 Center Drive West
Los Angeles, CA 90045

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

Platform: IBM PC Server 704 Model 8650-4M0
Database Manager: Oracle7 Version 7.3.3
Operating System: Solaris Workgroup Server Version 2.5.1
Transaction Manager: Tuxedo ETP System Version 4.2.2

CPUs	Memory	Disks	New-Order Response Time @ 90%	tpmC
Server: IBM PC Server 704 Model 8650-4M0				
4 Pentium Pro @ 200 MHz	Main: 2048 MB Cache: 512K	108 @ 4.5 GB	2.60 sec	6,679.50
6 Clients: IBM PC Server 320 Model 8640-0DV				
1 Pentium @ 133 MHz	256 MB	1 @ 2.25 GB	n.a.	n.a.

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark.

The following attributes of the benchmark were given special attention:

- The transactions were correctly implemented.
- The database files were properly sized and populated.
- The database was properly scaled with 564 warehouses.
- The ACID properties were met, including phantom protection.
- Input data was generated according to the specified percentages.
- Eight hours of mirrored log space was configured on the priced system.
- The data for the 180-day space calculation was verified.
- Emulated delays were added to the response time component for each transaction type.
- The steady state portion of the test was 22 minutes.
- One checkpoint was taken during the steady state portion of the test.
- Checkpoints were verified to be clear of the guard zones.

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

- One checkpoint was taken during the steady state portion of the test.
- Checkpoints were verified to be clear of the guard zones.
- There were 5,640 user contexts on the system.
- Each emulated user had a unique starting random number seed.
- The NURand constants used for database load and run-time random number generation were verified.
- System pricing was checked for major components and maintenance.

Additional Audit Notes: (none)

Regards,

Richard L. Gimarc

Richard L. Gimarc
Auditor

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

Appendix A: TPC-C Application Source

Client/Terminal Handler Code

makefile

```
CLIENT_OBJECTS = tpcc_client.o tpcc_tux.o tpcc_log.o tpcc_forms.o
TUXINC = /usr/tuxedo/include
#DEBUG = -DDEBUG
#OPT = -g
SHR =
CLIENT_FLAGS = -DTPMONITOR -I$(TUXINC) -I /usr/include/sys
CLIENT_LDFLAGS = $(SHR)
CFLAGS = $(OPT) $(DEBUG) $(CLIENT_FLAGS) $(SVR_FLAGS)
tpcc_tuxclient: $(CLIENT_OBJECTS)
        CFLAGS="$(CFLAGS)  -ldl -Bstatic -lc -lw -lintl" \
        buildclient -v -o $@ -f "$(CLIENT_OBJECTS)"
clean:
        rm -f *.o tpcc_tuxclient
tpcc_tux.o: tpcc_client.h tpcc_forms.h tpcc_tux.h
tpcc_client.o: tpcc_client.h tpcc_forms.h tpcc_tux.h
tpcc_tux.o: tpcc_client.h tpcc_forms.h tpcc_tux.h
tpcc_log.o: tpcc_client.h tpcc_forms.h tpcc_tux.h
```

tpcc_client/c

```
/*******************************************************************************
tpcc_client.c
*******************************************************************************/
#include    <stdio.h>
#include    <string.h>
#include    <unistd.h>
#include    <sys/types.h>
#include    <sys/time.h>
#include    <sys/procset.h>
#include    <sys/priocntl.h>
#include    <sys/rtpriocntl.h>
#include    <sys/param.h>
#include    <limits.h>
#include    <errno.h>
#include    "priocntl.h"
#include <stdlib.h>
#include <errno.h>
#include "tpcc_client.h"
#include "tpcc_tux.h"
main()
{
        int         menu_selection;
        void        do_transaction(int);
        initialize();
        Send_Menu();
        while ((menu_selection = sel_trans()) != 9) {
                if ((menu_selection < 1) || (menu_selection > 5))
                        continue;
                do_transaction(menu_selection - 1);
                Send_Menu();
        }
        rundown();
}
initialize()
{
        int         menu_selection, start, m, n;
        char list[] =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
;
        tty_in = 0;
        tty_out = 1;
        if (Init_Monitor()) {
        fprintf(stderr, "\033[24;1H\033[mUnable to connect to TP
 Monitor\n\01");
        exit(1);
        }
        get_wd();
        set_display();
}
rundown()
{
        restore_terminal();
        Rundown_Monitor();
}
get_wd(int num)
{
        num = 5 ;
        setup_wd();
        display_screen(num);
        get_inputs(num);
}
#ifdef      NULL_TRANS
#define     NEWO_MOD        20
#define     NEWO_MAXSLEEP   25
#define     PAYM_MOD        15
#define     PAYM_MAXSLEEP   20
#define     ORD_MOD         10
#define     ORD_MAXSLEEP    15
#define     STK_MOD         8
#define     STK_MAXSLEEP    12
#define     DEL_MOD         12
#define     DEL_MAXSLEEP    18
#endif
void
do_transaction(int num)
{
        int         status, sleep_time;
        display_screen(num);
        status = get_inputs(num);
        if (status == 3)
                return;
#ifdef      NULL_TRANS
        Clog("Txn type =  %d\n", num);
        switch (num) {
        case NEWORDER:
                if ((sleep_time  = w_id % NEWO_MOD) == 0)
                        sleep_time = NEWO_MAXSLEEP;
                break;
        case PAYMENT:
                if ((sleep_time  = w_id % PAYM_MOD) == 0)
                        sleep_time = PAYM_MAXSLEEP;
```

```
                break;
        case ORDSTAT:
                if ((sleep_time  = w_id % ORD_MOD) == 0)
                        sleep_time = ORD_MAXSLEEP;
                break;
        case STOCKLEV:
                if ((sleep_time  = w_id % STK_MOD) == 0)
                        sleep_time = STK_MAXSLEEP;
                break;
        case DELIVERY:
                if ((sleep_time  = w_id % DEL_MOD) == 0)
                        sleep_time = DEL_MAXSLEEP;
                break;
        }
        sleep(sleep_time);
#else
        if (Snd_Txn_To_Monitor(num)){
                syserr("\033[24;1H\033[mTPM Error detected -- See
$TPCC_HOME/CLIENTLOG for details\n");
                return;
        }
#endif
        display_output(num);
}
```

tpcc_client.h

```
/********************************************************** **********************
****************************** tpcc_client.h ***********************************
************************************************************ **********************/
#include <time.h>
#include <sys/types.h>
#include <time.h>
#define BOOLEAN int
#define LINEMAX 256
#define FALSE 0
#define TRUE 1
#define NEWORDER 0
#define PAYMENT 1
#define ORDSTAT 2
#define DELIVERY 3
#define STOCKLEV 4
#define WD 5
#define MAX_OL 15
#define TPM_ERROR 1
char        date_field[80];
char        tty_name[11];
int         w_id;
int         d_id;
int         xact_type;
/*
 * * Data structures of input data for each transaction type
 */
/*
 * * Data structures descriptions for IO data for each transaction type *
 *
 */
struct no_itm_struct {
        int         ol_supply_w_id;
        int         ol_i_id;
        char        i_name[25];
        int         ol_quantity;
        int         s_quantity;
        char        brand[2];
        double      i_price;
        double              ol_amount;
};
struct no_struct {
        int         w_id;
        int         d_id;
        int         c_id;
        int         o_id;
        int         o_ol_cnt;
        double      c_discount;
        double      w_tax;
        double      d_tax;
        char        o_entry_d[20];
    char        c_credit[3];
        char        c_last[17];
        struct              no_itm_struct o_ol[15];
    char            status[25];
        double      total;
};
struct pay_struct {
        int         w_id;
        int         d_id;
        int         c_id;
        int         c_w_id;
        int         c_d_id;
        double      h_amount;
        double      c_credit_lim;
        double      c_balance;
        double      c_discount;
        char        h_date[20];
        char        w_street_1[21];
        char        w_street_2[21];
        char        w_city[21];
        char        w_state[3];
        char        w_zip[11];
        char        d_street_1[21];
        char        d_street_2[21];
        char        d_city[21];
        char        d_state[3];
        char        d_zip[11];
        char        c_first[17];
        char        c_middle[3];
    char            c_last[17];
        char        c_street_1[21];
        char        c_street_2[21];
        char        c_city[21];
        char        c_state[3];
        char        c_zip[11];
        char        c_phone[17];
        char        c_since[11];
        char        c_credit[3];
        char c_data_1[51];
        char c_data_2[51];
        char c_data_3[51];
        char c_data_4[51];
};
struct ord_itm_struct {
```

```
        int             ol_supply_w_id;
        int             ol_i_id;
        int             ol_quantity;
        double          ol_amount;
        char            ol_delivery_d[11];
};
struct ord_struct {
        int             ol_cnt;
        int             w_id;
        int             d_id;
        int             c_id;
        int             o_id;
        int             o_carrier_id;
        double          c_balance;
        char            c_first[17];
        char            c_middle[3];
        char            c_last[17];
        char            o_entry_d[20];
        struct ord_itm_struct s_ol[MAX_OL];
};
struct del_struct {
        int             w_id;
        int             o_carrier_id;
        time_t          queue_time;
};
struct stock_struct {
        int             w_id;
        int             d_id;
        int             threshold;
        int             low_stock;
};
struct menu_struct {
        int             w_id;
        int             d_id;
};
/*
 * * Data structure for input & output data
 */
typedef union info {
                struct no_struct neworder;
                struct pay_struct payment;
                struct ord_struct ordstat;
                struct del_struct delivery;
                struct stock_struct stocklev;
                struct menu_struct wd;
        }       info_t;
struct io_tpcc {
        int             type;
        info_t                  info;
};
```

tpcc_forms.c

```
/*******************************************************************************
tpcc_forms.c
*******************************************************************************/
#include <stdio.h>
#include <sys/termio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include "tpcc_client.h"
#include "tpcc_forms.h"
#include "tpcc_tux.h"

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

void
setraw()
{
        /** put screen in raw mode **/

        extern struct tbufsave;
        struct termio   tbuf;
        int             status;
        if (ioctl(tty_in, TCGETA, &tbuf) == -1)
                syserr("ioctl_ERROR#1 - getting the original input term
setting error");
        tbufsave = tbuf;
        tbuf.c_iflag &= ~(INLCR | ICRNL | IUCLC | ISTRIP | IXON | BRKINT);
        tbuf.c_oflag &= ~OPOST;
        tbuf.c_lflag &= ~(ICANON | ISIG | ECHO);
        tbuf.c_cc[VMIN] = LEAVE_SCREEN_MIN;
        tbuf.c_cc[VTIME] = LEAVE_SCREEN_TIMEOUT;

        if (ioctl(tty_out, TCSETAF, &tbuf) == -2)
                syserr("ioctl_ERROR#2 - setting raw mode for STDIN
error");
}
void
restore_terminal()
{                                               /** restore terminal flags **/
        extern struct tbufsave;

        struct termio   tbuf;
        int             status;

        if (ioctl(tty_out, TCSETAF, &tbufsave) == -1)
                syserr("ioctl_ERROR#3 - restoring original input terminal
settings error");

        tbuf = tbufsave;
        if (ioctl(tty_out, TCSETAF, &tbuf) == -1)
                syserr("ioctl_ERROR#4 - Forcing the original settings back
for STDIN error");
}

int
sel_trans()
{
        int             c, read_count;
        static char     inbuf[2] = "\0\0";
        int             i = 0;

        read_count = read(tty_in, inbuf, 1);
        if (read_count == 0)
                syserr("TTY lost connection");
        if (inbuf[0] == QUIT)
                return 9;
```

```
		switch (inbuf[0]) {
		case 'n':
			c = 1; break;
		case 'p':
			c = 2; break;
		case 'o':
			c = 3; break;
		case 'd':
			c = 4; break;
		case 's':
			c = 5; break;
		case 'e':
			c = 9; break;
		}
		return c;
}

int	newo_val(int *);
int	paym_val(int *);
int	ords_val(int *);
int	del_val(int *);
int	stock_val(int *);
int	wd_val(int *);
int(*p_check_function[]) () = {
			&newo_val,
			&paym_val,
			&ords_val,
			&del_val,
			&stock_val,
			&wd_val
};

int
get_inputs(int txn_type)
{
		int	done = FALSE;
		int	i, returned_key;
		io_elem	*ioptr;
		int	last_input;
		float			float_h_amount = 0.0;

		memset(tuxibuf, '\0', sizeof(info_t ));
		int_h_amount = 0;

		last_input = Forms[txn_type].num_input_elems - 1;
		i = 0;
		while (done == FALSE) {

			ioptr = &Forms[txn_type].input_elems[i];

			if (txn_type == PAYMENT){
				if (i == 5)
					payment_input = TRUE;
				else
					payment_input = FALSE;
			}

			returned_key = (ioptr->fptr) (ioptr->x, ioptr->y,
 ioptr->len,ioptr->flags, ioptr->dptr);

			switch (returned_key) {
			case BACKTAB:
				if (i == 0)
					i = last_input;
				else
					i--;
				break;
			case TAB:
				if (i == last_input)
					i = 0;
				else
					i++;
				break;
			case QUIT:
				done = TRUE;
				break;
			case SUBMIT:
			case LF:
				if (screen_bufindex) {
					PAINTSCRLEN(screen_buf,
 screen_bufindex);
					screen_bufindex = 0;
				}
				payment_input = FALSE;
				done = (p_check_function[txn_type]) (&i);
				break;
			}
		}
		return returned_key;
}
int
newo_val(int *pos)
{
		int	done = FALSE;
		struct no_itm_struct *ol_ptr;
		int		blank_line = 0, i;

		iNO->w_id = w_id;

		if (iNO->d_id <= 0) {
			*pos = 0;
			message = TRUE;
			PAINTSCR(MANDATORY_MSG);

		} else if (iNO->c_id <= 0) {
			*pos = 1;
			message = TRUE;
			PAINTSCR(MANDATORY_MSG);

		} else {

			ol_ptr = iNO->o_ol;

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

				if (ol_ptr->ol_i_id || ol_ptr->ol_supply_w_id
				    || ol_ptr->ol_quantity)
				{
```

```
                                        /* and is that data complete */
                                        if (ol_ptr->ol_i_id &&
 ol_ptr->ol_supply_w_id
                                           && ol_ptr->ol_quantity)
                                        {
                                                if (blank_line == 0){

iNO->o_ol_cnt++;
                                                }else{
                                                        *pos = 2;
                                        PAINTSCR(INCOMPLINE_MSG);
                                        message = TRUE;
                                        iNO->o_ol_cnt = 0;
                                        return FALSE;
                                                }
                                        } else {
                                                *pos = 2 + 3 * i;

PAINTSCR(INCOMPLINE_MSG);
                                                message = TRUE;
                                                iNO->o_ol_cnt = 0;
                                                return FALSE;
                                        }
                                } else blank_line=1;
                        }
                        if (!iNO->o_ol_cnt) {
                                *pos = 2;
                                PAINTSCR(MANDATORY_MSG);
                                message = TRUE;
                                iNO->o_ol_cnt = 0;
                                return FALSE;
                        }
                        done = TRUE;
                }
                return done;
}
int paym_val(int *pos)
{
                int     done = FALSE;
                iPT->w_id = w_id;
                if (iPT->d_id <= 0) {
                        *pos = 0;
                        message = TRUE;
                        PAINTSCR(MANDATORY_MSG);
                } else if (iPT->c_w_id <= 0) {
                        *pos = 2;
                        message = TRUE;
                        PAINTSCR(MANDATORY_MSG);
                } else if (iPT->c_d_id <= 0) {
                        *pos = 3;
                        message = TRUE;
                        PAINTSCR(MANDATORY_MSG);
                } else if (int_h_amount <= 0) {
                        *pos = 5;
                        message = TRUE;
                        PAINTSCR(MANDATORY_MSG);
                } else if (iPT->c_id <= 0) {
                        if (iPT->c_last[0] == '\0') {
                                message = TRUE;
                                PAINTSCR(ID_OR_LAST_MSG);
                                *pos = 1;
                        } else {
#ifdef          NULL_TRANS
                                Clog("Customer lastname  %s\n",
 &iPT->c_last[0]);
#endif
                                done = TRUE;
                        }
                } else {
#ifdef          NULL_TRANS
                                Clog("Customer id  %d\n", iPT->c_id);
#endif
                        done = TRUE;
                }
                iPT->h_amount = int_h_amount ;
                return done;
}
int ords_val(int *pos)
{
                int     done = FALSE;
                iOS->w_id = w_id;
                if (iOS->d_id <= 0) {
                        *pos = 0;
                        message = TRUE;
                        PAINTSCR(MANDATORY_MSG);
                } else if (iOS->c_id <= 0) {
                        if (iOS->c_last[0] == '\0') {
                                message = TRUE;
                                PAINTSCR(ID_OR_LAST_MSG);
                                *pos = 1;
                        } else {
                                done = TRUE;
                        }
                } else
                        done = TRUE;
                return done;
}
int del_val(int *pos)
{
                int     done = FALSE;
                iDY->w_id = w_id;
                if (iDY->o_carrier_id <= 0) {
                        message = TRUE;
                        PAINTSCR(MANDATORY_MSG);
                } else {
                        time(&iDY->queue_time);
                        done = TRUE;
                }
                return done;
}
int stock_val(int *pos)
{
                int     done = FALSE;
                iSL->w_id = w_id;
                iSL->d_id = d_id;
                if (iSL->threshold <= 0) {
                        message = TRUE;
                        PAINTSCR(MANDATORY_MSG);
                } else
                        done = TRUE;
```

```
        return done;
}
int wd_val(int *pos)
{
        int     done = FALSE;
        if (iWD->w_id == 0 || iWD->d_id == 0) {
                message = TRUE;
                PAINTSCR(MANDATORY_MSG);
        } else {
                w_id = iWD->w_id ;
                d_id = iWD->d_id;
                done = TRUE;
        }
        return done;
}
void setup_wd()
{
        io_elem     *p;
    char        buf[128];
    void        setup_io_elems();
    setraw();
    setup_screen_buffer(&Forms[5], 5);
        p = Forms[WD].input_elems;
    p++->dptr = &iWD->w_id;
    p++->dptr = &iWD->d_id;
        CLRSCN(buf);
        PAINTSCR(buf);
}
void set_display()
{
        int         i;
        char        buf[128];
        void        setup_io_elems();
        for (i = 0; i < MAX_FORMS; i++)
                setup_screen_buffer(&Forms[i], i);
        setup_io_elems();
        CLRSCN(buf);
        PAINTSCR(buf);
}
void display_screen(int screen_num)
{
        if (PAINTSCRLEN(Forms[screen_num].blank_form,
           Forms[screen_num].blank_formlen) == -1)
                syserr("Can't write out form");
}
void Send_Menu()
{
        if (PAINTSCRLEN(menu_buf, menu_buflen) == -1)
                syserr("Can't send menu");
}
void setup_io_elems()
{
        io_elem     *p;
        int         i;
        p = Forms[NEWORDER].input_elems;
        p++->dptr = &iNO->d_id;
        p++->dptr = &iNO->c_id;
        for (i = 0; i < 15; i++) {
                p++->dptr = &iNO->o_ol[i].ol_supply_w_id;
                p++->dptr = &iNO->o_ol[i].ol_i_id;
                p++->dptr = &iNO->o_ol[i].ol_quantity;
        }
        p = Forms[PAYMENT].input_elems;
        p++->dptr = &iPT->d_id;
        p++->dptr = &iPT->c_id;
        p++->dptr = &iPT->c_w_id;
        p++->dptr = &iPT->c_d_id;
        p++->dptr = (int *) &iPT->c_last[0];
        p->dptr = &int_h_amount;
        p = Forms[ORDSTAT].input_elems;
        p++->dptr = &iOS->d_id;
        p++->dptr = &iOS->c_id;
        p->dptr = (int *) &iOS->c_last[0];
        p = Forms[DELIVERY].input_elems;
        p->dptr = &iDY->o_carrier_id;
        p = Forms[STOCKLEV].input_elems;
        p->dptr = &iSL->threshold;
}
int
setup_screen_buffer(struct form_info * form_ptr, int txn_type)
{
        FILE        *ifile;
        const text_elem     *tbuf;
        char        *bufp;
        int         ct;
        char        input_display_buf[64];
        io_elem     *io_ptr;
        bufp = screen_buf;
        bufp += CLRSCN(bufp);
        tbuf = form_ptr->tp;
        while (tbuf->text) {
                bufp += DISPLAY(bufp, tbuf->y, tbuf->x, tbuf->text);
                tbuf++;
        }
        bufp += SWITCH_TO_UNDERL(bufp);
        ct = 0;
        for (io_ptr = form_ptr->input_elems; io_ptr->y != 999; io_ptr++) {
                strncpy(input_display_buf, blanks, io_ptr->len);
                input_display_buf[io_ptr->len] = '\0';
                bufp += DISPLAY(bufp, io_ptr->x, io_ptr->y,
 input_display_buf);
                ct++;
        }
        form_ptr->num_input_elems = ct;
        bufp += SWITCH_TO_NORMAL(bufp);
        if (txn_type == PAYMENT)
                bufp += DISPLAY_INT(bufp, 4, 12, 4, w_id);
        else if (txn_type != 5)
                bufp += DISPLAY_INT(bufp, 4, 12, 2, w_id);
        if (txn_type == STOCKLEV)
                bufp += DISPLAY_INT(bufp, 2, 29, 2, d_id);
        bufp += SWITCH_TO_UNDERL(bufp);
        *bufp++ = '\1';
        *bufp = '\0';
        form_ptr->blank_formlen = bufp - screen_buf + 1;
        if (!form_ptr->blank_form &&
           ((form_ptr->blank_form = malloc(form_ptr->blank_formlen)) ==
 NULL)) {
                Clog("setup_screen_buffer: malloc failed\n");
                exit(1);
```

```
		}
		memcpy(form_ptr->blank_form, screen_buf, form_ptr->blank_formlen);
		memset(screen_buf, '\0', form_ptr->blank_formlen);
}
int
read_integer(col, row, size, flags, data)
		int		col, row, size, flags, *data;
/** Function to read in integer data from the screen.
col - first column position of the data field on the screen.
row - row position of the data field on the screen.
size - the length of the data field in number of characters to read in.
flags - boolean flag that was used to identify the mandatory data fields.
data - pointer to the actual location where the returning value
should be stored in.  This function reads in the input data until either a
TAB, BACKTAB, LINEFEED, SUBMIT character is entered or the maximum
 number of
characters have been entered into the data field.
The data is read as character and converted to an integer before
being stored in the appropriate location.
**/
{
		int	exit_read_function = FALSE, previous_data_exists = FALSE;
		int		return_status = TAB, bytes_read = 0, i = 0, j = 0, k = 0,
				size1 = 0, cur_col = col;
		char		*bufp, temp[50];
		float		q;
		char		erase_field[20];
		strncpy(temp, " " , 1);
		bufp = screen_buf + screen_bufindex;
		/* Position cursor at start of field */
		if (curbuf_read == read_count || curbuf_read == 0) {
				screen_buf[0] = '\0';
				bufp += GOTOXY(bufp, col + size - 1, row);
				PAINTSCRLEN(screen_buf, bufp - screen_buf);
				bufp = screen_buf;
		}
		size1 = size;
		if (*data > 0)
				previous_data_exists = TRUE;
		while (exit_read_function == FALSE) {
				/*
				 * Below we read from standard input into the array curbuf.
				 * curbuf_read is the pointer to the array curbuf indicating
				 * the position upto which the curbuf has been parsed.
				 * curbuf_consumed is the number of elements in the buffer
				 * temp that holds the array that is to be displayed.
				 * Elements of curbuf_consumed is selectively copied from
				 * curbuf Note:read_count is the total number of characters
				 * in the buffer curbuf. curbuf_read is always less than or
				 * equal to read_count.
				 */
				if (curbuf_read == read_count || curbuf_read == 0) {
						curbuf_read = 0;
						read_count = read(tty_in, curbuf,
 sizeof(curbuf));
						if (read_count == 0)
								syserr("TTY lost connection");
				}
				/*
				 * int message prevents unnecessary display of warning
				 * messages
				 */
				if (message == TRUE) {
						bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW, ERASE_MSG);
						message = FALSE;
				}
				if (previous_data_exists == TRUE) {
						if (curbuf[curbuf_read] == DELETE) {
								previous_data_exists = FALSE;
								strncpy(erase_field, blanks, size);
								erase_field[size] = '\0';
								bufp += DISPLAY(bufp, col, row,
erase_field);
								bufp += GOTOXY(bufp, col + size -
1, row);
						} else {
								if (curbuf[curbuf_read] < '0' ||
curbuf[curbuf_read] > '9') {
										exit_read_function =
TRUE;
										previous_data_exists =
FALSE;
										return_status =
curbuf[curbuf_read];
										curbuf[curbuf_read] =
'\0';
								} else {
										previous_data_exists =
FALSE;
										strncpy(erase_field,
blanks, size);
										erase_field[size] = '\0';
										bufp += DISPLAY(bufp,
col, row, erase_field);
										/*
										 * bufp = screen_buf;
										 */
								}
						}
				}			/* if previous_data_exists */
				while ((curbuf_read < read_count) && (exit_read_function
== FALSE)) {
						/*
						 * intermediate variable size1 for cases when
						 * floating point field whose size is less than
						 * actual size by 1 because of decimal.
						 */
						if (payment_input == TRUE)
								size1 = size - 1;
						/*
						 * Test for integer
						 */
						if ((curbuf[curbuf_read] >= '0' &&
curbuf[curbuf_read] <= '9') || (curbuf[curbuf_read] == '.')) {
								/*
								 * Consume all integers in buffer
								 */
								for (; curbuf_read < read_count &&
								    ((curbuf[curbuf_read] >= '0'
```

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

```
blanks, size);
                                                        erase_field[size] = '\0';
                                                        bufp = screen_buf;
                                                        screen_bufindex = 0;
                                                        bufp += DISPLAY(bufp,
col, row, erase_field);
                                                        if (curbuf_consumed <
3)
                                                                bufp +=
DISPLAY_FLOAT(bufp, 2,
                                                                 (col + size
- 4), row, q);
                                                        else
                                                                bufp +=
DISPLAY_FLOAT(bufp, 2,
    (col + size - curbuf_consumed - 1), row, q);
                                                        if (cur_col != 0)
                                                                cur_col--;
                                                        if (curbuf_read < 40)

curbuf_read++;          /* pressed key overflow

* situations */
                                                        bufp += GOTOXY(bufp,
col + size, row);
                                                } else {
                                                        if (curbuf_consumed !=
0)

curbuf_consumed--;
                                                        curbuf[curbuf_read] =
'\0';
                                                        curbuf_read++;
                                                        if (message == TRUE) {
                                                                bufp +=
DISPLAY(bufp, MESSAGE_COL,

MESSAGE_ROW,

ERASE_MSG);
                                                                message =
FALSE;
                                                        }
                                                        OVERFLOW = FALSE;

PAINTSCR(screen_buf);
                                                        temp[curbuf_consumed]
= '\0';
                                                        strncpy(erase_field,
blanks, size);
                                                        erase_field[size] = '\0';
                                                        bufp = screen_buf;
                                                        screen_bufindex = 0;
                                                        bufp += DISPLAY(bufp,
col, row, erase_field);
                                                        bufp += DISPLAY(bufp,
(col + size -
                                                             curbuf_consumed),
row, temp);
                                                        if (cur_col != 0)
                                                                cur_col--;
                                                        bufp += GOTOXY(bufp,
col + size, row);
                                                }
                                        }
                                        /* end of if DELETE */
                                         /* could be a ^C */
                                        else if (curbuf[curbuf_read] == QUIT) {
                                                temp[0] = '\0';
                                                return_status = QUIT;
                                                curbuf[curbuf_read] = '\0';
                                                exit_read_function = TRUE;
                                        } else {
                                /** Any other character entered at the keyboard ... **/
                                                if (message == FALSE) {
                                                        bufp += DISPLAY(bufp,
MESSAGE_COL, MESSAGE_ROW, INVALID_MSG);
                                                        bufp += GOTOXY(bufp,
col + size, row);

PAINTSCR(screen_buf);
                                                        bufp = screen_buf;
                                                        screen_bufindex = 0;
                                                        message = TRUE;
                                                }
                                                curbuf_read++;
                                        }
                                }                       /** End of the WHILE loop **/
                                if (OVERFLOW == TRUE && exit_read_function ==
FALSE) {
                                        /*
                                         * if number of characters are exceeding the
field
                                         * limit beep and warning message is necessary
                                         */
                                        if (message == FALSE) {
                                                bufp += DISPLAY(bufp,
MESSAGE_COL,
                                                                MESSAGE_ROW,
EXC_FLD_LIM_MSG);
                                                PAINTSCR(screen_buf);
                                                bufp = screen_buf;
                                                screen_bufindex = 0;
                                                message = TRUE;
                                        }
                                        *data = atoi(temp);
                                        return_status = curbuf[curbuf_read];
                                        curbuf[curbuf_read] = '\0';
                                        curbuf_read = 0;
                                        OVERFLOW = FALSE;
                                } else {
                                        screen_bufindex = bufp - screen_buf;
                                        if ((curbuf_read == read_count) || (curbuf_read
== 0)
                                           || (screen_bufindex > SCRBUF_LEN -
CURBUFLEN)) {
                                                PAINTSCRLEN(screen_buf,
screen_bufindex);
                                                screen_bufindex = 0;
```

```
                                        bufp = screen_buf;
                                }
                        }
                }
                /* ensuring unnecessary warning messages are removed */
                if (message == TRUE) {
                        bufp += DISPLAY(bufp, MESSAGE_COL,
                                        MESSAGE_ROW, ERASE_MSG);
                        message = FALSE;
                        PAINTSCR(screen_buf);
                        bufp = screen_buf;
                        screen_bufindex = 0;
                }
                return (return_status);
}
int
read_string(col, row, size, flags, data)
                int             col, row, size, flags;
                char            *data;
{
                int         exit_read_function = FALSE, previous_data_exists = FALSE,
 data_full = FALSE;
                int             return_status = TAB, bytes_read = 0, i = 0, j = 0,
                                size_tot = 0;
                char            *bufp, temp[80];
                char            erase_field[20];
                strncpy(temp, "\0", 1);
                curbuf_consumed = 0;
                bufp = screen_buf + screen_bufindex;
                /* Position cursor at start of field */
                if (curbuf_read == read_count || curbuf_read == 0) {
                        screen_buf[0] = '\0';
                        bufp += GOTOXY(bufp, col, row);  /* Goto input area */
                        PAINTSCRLEN(screen_buf, bufp - screen_buf);
                        bufp = screen_buf;
                }
                if ((*(char *) data) != '\0')
                        previous_data_exists = TRUE;
                while (exit_read_function == FALSE) {
                        /*
                         * Below we read from standard input into the array curbuf.
                         * curbuf_read is the pointer to the array curbuf indicating
                         * the position upto which the curbuf has been parsed.
                         * curbuf_consumed is the number of elements in the buffer
                         * temp that holds the array that is to be displayed.
                         * Elements of curbuf_consumed is selectively copied from
                         * curbuf Note:read_count is the total number of characters
                         * in the buffer curbuf. curbuf_read is always less than or
                         * equal to read_count.
                         */
                        if (curbuf_read == read_count) {
                                curbuf_read = 0;
                                read_count = read(tty_in, curbuf, size -
 size_tot);
                                if (read_count == 0)
                                        syserr("TTY lost connection");
                        }
                        if (message == TRUE) {
                                bufp += DISPLAY(bufp, MESSAGE_COL,
                                                MESSAGE_ROW,
 ERASE_MSG);
                                message = FALSE;
                        }
                        if (previous_data_exists == TRUE) {
                                if (curbuf[curbuf_read] == DELETE) {
                                        previous_data_exists = FALSE;
                                        strncpy(erase_field, blanks, size);
                                        erase_field[size] = '\0';
                                        bufp += DISPLAY(bufp, col, row,
 erase_field);
                                        bufp += GOTOXY(bufp, col, row);
                                } else {
                                        if (curbuf[curbuf_read] < ' ' ||
 curbuf[curbuf_read] > '~') {
                                                exit_read_function =
 TRUE;
                                                previous_data_exists =
 FALSE;
                                                return_status =
 curbuf[curbuf_read];
                                                curbuf[curbuf_read] =
 '\0';
                                        } else {
                                                previous_data_exists =
 FALSE;
                                                strncpy(erase_field,
 blanks, size);
                                                erase_field[size] = '\0';
                                                bufp += DISPLAY(bufp,
 col, row, erase_field);
                                                bufp += GOTOXY(bufp,
 col, row);
                                        }
                                }
                        }
                        while ((curbuf_read < read_count) && (exit_read_function
 == FALSE)) {
                                if (curbuf[curbuf_read] >= ' ' &&
 curbuf[curbuf_read] <= '~') {       /** if between ASCII space (040) through ~
 (0176) **/
                                        for (; curbuf[curbuf_read] >= ' '
                                            && curbuf[curbuf_read] <= '~';
 curbuf_read++) {
                                                /*
                                                 * ensuring the
 curbuf_consumed is
                                                 * not more than field
 size
                                                 */
                                                if (curbuf_consumed <
 size) {

temp[curbuf_consumed] = curbuf[curbuf_read];

curbuf_consumed++;
                                                }
                                                /* else overflow
 condition */
                                                else
```

```
OVERFLOW = TRUE;
                                                curbuf[curbuf_read] =
'\0';          /* erasing characters
                * already read from the
                * buffer */
                                        }
                                        temp[curbuf_consumed] = '\0';
/* terminate temp string */
                                        bufp += DISPLAY(bufp, col, row,
temp);
                                        return_status =
curbuf[curbuf_read];
                                } else if (curbuf[curbuf_read] == TAB
                                        || curbuf[curbuf_read] == LF
                                        || curbuf[curbuf_read] ==
BACKTAB
                                        || curbuf[curbuf_read] ==
SUBMIT) {
                                        if (curbuf_consumed > 0) {
                                                if (message == TRUE) {
                                                        bufp +=
DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
ERASE_MSG);
                                                        message =
FALSE;
                                                }
                                                temp[curbuf_consumed]
= '\0';
                                                strcpy(data, temp);
                                                exit_read_function =
TRUE;
                                                return_status =
curbuf[curbuf_read];
                                                curbuf[curbuf_read] =
'\0';
                                                curbuf_read++;
                                                curbuf_consumed = 0;
                                        } else {
                                                if (message == TRUE) {
                                                        bufp +=
DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
ERASE_MSG);
                                                        message =
FALSE;
                                                }
                                                temp[curbuf_consumed]
= '\0';
                                                strcpy(data, temp);
                                                exit_read_function =
TRUE;
                                                return_status =
curbuf[curbuf_read];
                                                curbuf[curbuf_read] =
'\0';
                                                curbuf_read++;
                                        }
                                } else if (curbuf[curbuf_read] == DELETE) {
                                        for (curbuf_read = curbuf_read;
curbuf[curbuf_read] ==
                                            DELETE
                                            ; curbuf_read++) {
                                                curbuf[curbuf_read] =
'\0';
                                                temp[curbuf_consumed
- 1] = '\0';
                                                if (curbuf_consumed !=
0)
curbuf_consumed--;
                                        }
                                        if (curbuf_consumed >= 0) {
                                                bufp +=
BLANK_UNDERLINE(bufp, col, row, " ");
                                                bufp += DISPLAY(bufp,
col, row, temp);
PAINTSCR(screen_buf);
                                                bufp = screen_buf;
                                                screen_bufindex = 0;
                                        } else {
                                                if (message == FALSE)
{
                                                        bufp +=
DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
 EXC_FLD_LIM_MSG);
                                                        bufp +=
BEEP(bufp);
PAINTSCR(screen_buf);
                                                        bufp =
screen_buf;
screen_bufindex = 0;
                                                        message =
TRUE;
                                                }
                                                curbuf[curbuf_read] =
'\0';
                                                curbuf_read = 0;
                                        }
                                } else if (curbuf[curbuf_read] == QUIT) {
                                        temp[0] = '\0';
                                        return_status = QUIT;
                                        curbuf[curbuf_read] = '\0';
                                        exit_read_function = TRUE;
                                } else {/** Any other character entered at the
keyboard ... **/
                                        if (message == FALSE) {
                                                bufp += DISPLAY(bufp,
MESSAGE_COL, MESSAGE_ROW, INVALID_MSG);
                                                bufp += GOTOXY(bufp,
```

```
col, row);
                                                message = TRUE;
                                        }
                                        curbuf_read++;
                                }
                        }               /** End of the WHILE loop **/
                        if (OVERFLOW == TRUE && exit_read_function ==
FALSE)
                                /** If read enough to fill the size already **/
                        {
                                if (message == FALSE) {
                                        bufp += DISPLAY(bufp,
MESSAGE_COL,
                                                        MESSAGE_ROW,
EXC_FLD_LIM_MSG);
                                        PAINTSCR(screen_buf);
                                        bufp = screen_buf;
                                        screen_bufindex = 0;
                                        message = TRUE;
                                }
                                OVERFLOW = FALSE;
                                temp[curbuf_consumed] = '\0';
                                strcpy(data, temp);
                                curbuf_consumed--;
                                return_status = curbuf[curbuf_read];
                        } else {
                                screen_bufindex = bufp - screen_buf;
                                if ((curbuf_read == read_count) || (curbuf_read
== 0)
                                   || (screen_bufindex > SCRBUF_LEN -
CURBUFLEN)) {
                                        PAINTSCRLEN(screen_buf,
screen_bufindex);
                                        screen_bufindex = 0;
                                        bufp = screen_buf;
                                }
                        }
                }
                if (message == TRUE) {
                        bufp += DISPLAY(bufp, MESSAGE_COL,
MESSAGE_ROW,
                                        ERASE_MSG);
                        message = FALSE;
                        PAINTSCR(screen_buf);
                        screen_bufindex = 0;
                }
                return (return_status);
}
void    display_newo();
void    display_paym();
void    display_ords();
void    display_del();
void    display_stock();
void    (*p_print_function[]) () = {
                        &display_newo,
                        &display_paym,
                        &display_ords,
                        &display_del,
                        &display_stock
};

display_output(int txn_type)
{
                char                    c;
                (p_print_function[txn_type]) ();
                read(tty_in, &c, 1);
}
void
display_newo()
{
                struct no_itm_struct *ol_ptr, *ool;
                char            *bufp;
                int             i, r;
                double          ol_amount, total_amount = 0.0;
                bufp = output_screen;
                if (oNO->status == '\0') {
                        PAINTSCR(EXECUTION_STATUS_MSG);
                        return;
                } else {
                        bufp += SWITCH_TO_NORMAL(bufp);
                        bufp += DISPLAY(bufp, 61, 2, oNO->o_entry_d);
                        bufp += DISPLAY(bufp, 25, 3, oNO->c_last);
                        bufp += DISPLAY(bufp, 52, 3, oNO->c_credit);
                        bufp += DISPLAY_FLOAT(bufp, 5, 64, 3, oNO->c_discount
* 100.0);
                        bufp += DISPLAY_INT(bufp, 8, 15, 4, oNO->o_id);
                        bufp += DISPLAY_INT(bufp, 2, 42, 4, oNO->o_ol_cnt);
                        bufp += DISPLAY_FLOAT(bufp, 5, 59, 4, oNO->w_tax *
100.0);
                        bufp += DISPLAY_FLOAT(bufp, 5, 74, 4, oNO->d_tax *
100.0);
                        ol_ptr = iNO->o_ol;
                        ool = oNO->o_ol;
                        for (i = 0, r = FIRST_OL_ROW; i < iNO->o_ol_cnt;
                           r++, i++, ol_ptr++, ool++) {
                                ol_amount = ol_ptr->ol_quantity * ool->i_price;
                                total_amount += ol_amount;
                                bufp += DISPLAY(bufp, 19, r, ool->i_name);
                                bufp += DISPLAY_INT(bufp, 3, 51, r,
ool->s_quantity);
                                bufp += DISPLAY(bufp, 58, r, ool->brand);
                                bufp += DISPLAY_MONEY(bufp, 6, 62, r,
ool->i_price);
                                bufp += DISPLAY_MONEY(bufp, 7, 71, r,
ol_amount);
                        }
                total_amount *= (1+oNO->w_tax+oNO->d_tax)*(1-oNO->c_discount);
                        bufp += DISPLAY_MONEY(bufp, 8, 70, 22, total_amount);
                        bufp += DISPLAY(bufp, 23, 75, "**((");
                        *bufp++ = '\0';
                        PAINTSCRLEN(output_screen, bufp - output_screen);
                }
#ifdef DEBUG
                Clog("DBG: Screen output chars = %d\n", (bufp - &output_screen[0]));
#endif
}
void
display_paym()
{
                char            *bufp, temp[51], tempbuf2[201];
                char            *make_phone(char *), *make_zip(char *);
```

```
	bufp = output_screen;
	bufp += SWITCH_TO_NORMAL(bufp);		/* jr */
	bufp += DISPLAY(bufp, 7, 2, oPT->h_date);
	bufp += DISPLAY(bufp, 1, 5, oPT->w_street_1);
	bufp += DISPLAY(bufp, 1, 6, oPT->w_street_2);
	bufp += DISPLAY(bufp, 1, 7, oPT->w_city);
	bufp += DISPLAY(bufp, 22, 7, oPT->w_state);
	bufp += DISPLAY(bufp, 25, 7, make_zip(oPT->w_zip));
	bufp += DISPLAY(bufp, 42, 5, oPT->d_street_1);
	bufp += DISPLAY(bufp, 42, 6, oPT->d_street_2);
	bufp += DISPLAY(bufp, 42, 7, oPT->d_city);
	bufp += DISPLAY(bufp, 63, 7, oPT->d_state);
	bufp += DISPLAY(bufp, 66, 7, make_zip(oPT->d_zip));
	bufp += DISPLAY_INT(bufp, 4, 11, 9, oPT->c_id);
	bufp += DISPLAY(bufp, 29, 10, oPT->c_last);
	bufp += DISPLAY(bufp, 9, 10, oPT->c_first);
	bufp += DISPLAY(bufp, 26, 10, oPT->c_middle);
	bufp += DISPLAY(bufp, 9, 11, oPT->c_street_1);
	bufp += DISPLAY(bufp, 9, 12, oPT->c_street_2);
	bufp += DISPLAY(bufp, 9, 13, oPT->c_city);
	bufp += DISPLAY(bufp, 30, 13, oPT->c_state);
	bufp += DISPLAY(bufp, 33, 13, make_zip(oPT->c_zip));
	bufp += DISPLAY(bufp, 58, 10, oPT->c_since);
	bufp += DISPLAY(bufp, 58, 11, oPT->c_credit);
	bufp += DISPLAY_FLOAT(bufp, 5, 58, 12, oPT->c_discount * 100.0);
	bufp += DISPLAY(bufp, 58, 13, make_phone(oPT->c_phone));
	bufp += DISPLAY_MONEY(bufp, 14, 55, 15, oPT->c_balance);
	bufp += DISPLAY_MONEY(bufp, 13, 17, 16, oPT->c_credit_lim);
	if (oPT->c_data_1[0] != NULL) {
		bufp += DISPLAY50(bufp, 12, 18, oPT->c_data_1);
		bufp += DISPLAY50(bufp, 12, 19, oPT->c_data_2);
		bufp += DISPLAY50(bufp, 12, 20, oPT->c_data_3);
		bufp += DISPLAY50(bufp, 12, 21, oPT->c_data_4);
	}
	if (!oPT->h_date)
		bufp += DISPLAY(bufp, MESSAGE_COL,
 MESSAGE_ROW - 2, BAD_INPUTS);
	bufp += DISPLAY(bufp, 23, 75, "**((");
	*bufp++ = '\0';
	PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
	Clog("DBG: Screen output chars = %d\n", (bufp - &output_screen[0]));
#endif
}
void
display_ords()
{
	struct ord_itm_struct *sol;
	char	*bufp;
	int	i = 0, r = 8;
	bufp = output_screen;
	bufp += SWITCH_TO_NORMAL(bufp);
	bufp += DISPLAY_INT(bufp, 4, 11, 3, oOS->c_id);
	bufp += DISPLAY(bufp, 44, 3, oOS->c_last);
	bufp += DISPLAY(bufp, 24, 3, oOS->c_first);
	bufp += DISPLAY(bufp, 41, 3, oOS->c_middle);
	bufp += DISPLAY_MONEY(bufp, 9, 15, 4, oOS->c_balance);
	bufp += DISPLAY_INT(bufp, 8, 15, 6, oOS->o_id);
	bufp += DISPLAY(bufp, 38, 6, oOS->o_entry_d);
	bufp += DISPLAY_INT(bufp, 2, 76, 6, oOS->o_carrier_id);
	for (i = 0; i < oOS->ol_cnt; i++) {
		sol = &oOS->s_ol[i];
		if (sol->ol_supply_w_id > 0) {
			bufp += DISPLAY_INT(bufp, 4, 3, r,
 sol->ol_supply_w_id);
			bufp += DISPLAY_INT(bufp, 6, 14, r,
 sol->ol_i_id);
			bufp += DISPLAY_INT(bufp, 2, 25, r,
 sol->ol_quantity);
			bufp += DISPLAY_MONEY(bufp, 8, 32, r,
 sol->ol_amount);
			bufp += DISPLAY(bufp, 47, r,
 sol->ol_delivery_d);
			r++;
		}
	}
	if (!oOS->ol_cnt)
		bufp += DISPLAY(bufp, MESSAGE_COL,
 MESSAGE_ROW - 2, BAD_INPUTS);
	bufp += DISPLAY(bufp, 23, 75, "**((");
	*bufp++ = '\0';
	PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
	Clog("DBG: Screen output chars = %d\n", (bufp - &output_screen[0]));
#endif
}
void
display_del()
{
	char	*bufp;
	bufp = output_screen;
	/*PAINTSCR(DELIVERY_QUEUED_MSG);*/
	bufp += sprintf(bufp,"%s",DELIVERY_QUEUED_MSG);
	bufp += DISPLAY(bufp, 23, 75, "**((");
	*bufp++ = '\0';
	PAINTSCRLEN(output_screen, bufp - output_screen);
#ifdef DEBUG
	Clog("DBG: Screen output chars = %d\n", (bufp -

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

&output_screen[0]));
#endif
}
```

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

tpcc_forms.h

```
/*******************************************************************************
tpcc_forms.h
*******************************************************************************/

#include <sys/termio.h>
extern int      tty_in;
extern int      tty_out;
#define MAX_FORMS 6
#define MESSAGE_ROW 24
#define MESSAGE_COL 1
#define RTE_SYNCH_CHARACTER '\1'
#define SCRBUF_LEN 1536
#define FIRST_OL_ROW 7
#define CLRSCN(buf) sprintf(buf,"\033[H\033[2J")
#define DISPLAY_INT(buf,wid,x,y,ip) sprintf(buf,"\033[%d;%dH%*.1d", y, x,wid,ip)
#define DISPLAY_MONEY(buf,wid,x,y,fp)
 sprintf(buf,"\033[%d;%dH$%#*.2f",y,x,wid,fp)
#define DISPLAY_FLOAT(buf,wid,x,y,fp) sprintf(buf,"\033[%d;%dH%#*.2f", y,
 x,wid,fp)
#define DISPLAY(buf,x,y,txt) sprintf(buf,"\033[%d;%dH%s", y, x,txt)
#define DISPLAY50(buf,x,y,txt) sprintf(buf,"\033[%d;%dH%50.50s", y, x,txt)
#define PAINTSCR(buf) write(tty_out,buf,strlen(buf))
#define PAINTSCRLEN(buf, len) write(tty_out,buf,len)
#define SWITCH_TO_NORMAL(buf) sprintf(buf,"\033[m")
#define SWITCH_TO_UNDERL(buf) sprintf(buf,"\033[4m")
#define GOTOXY(buf,x,y) sprintf(buf,"\033[%d;%dH", y, x)
#define BEEP(buf) sprintf(buf,"\007")
#define BLANK_UNDERLINE(buf,x,y,txt)
 sprintf(buf,"\033[4m;\033[%d;%dH%s",y,x,txt);

#define CLRSCN_STR "'\033[H\033[2J'
#define DISPLAY_STR(x,y,txt) '\033[/**/y;/**/xH/**/txt'

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

#define CANCELLED 3
#define PREVIOUS_FIELD 4

/** 52 Possible key strokes read in by the read functions. Some are also
returned as status from the read functions.  **/

#define BACKTAB 2               /** Decided to use the CTRL B for now **/
#define DELETE 8
#define ESCAPE 27
#define LF 10
#define QUIT 3                          /** CNTRL-C Key stroke to quit fo
 now **/
#define SPACE 32
#define SUBMIT 13               /** Done with screen and submit key: CR **/

#define TAB 9
#define UNDERLINE 95
#define LEAVE_SCREEN_MIN 300            /** Minimum # of characters
                                        to leave screen **/
#define LEAVE_SCREEN_TIMEOUT 2          /** Minimum time to leave
                                        screen, 10=1sec **/
static int      curbuf_consumed = 0;
static int      curbuf_read = 0;
static int      read_count = 0;
#define       CURBUFLEN               300
static char     curbuf[CURBUFLEN];
static BOOLEAN  OVERFLOW = FALSE;
static BOOLEAN  message;        /* for suppressing warning messages */
BOOLEAN         payment_input = FALSE;      /* for setting money condition in
                                                        * read_int */
static struct termio tbufsave;
extern void     syserr();
void            Init_Screen();
void            display_screen_array(int);
void            Send_Menu();
int             Get_Menu_Input();
/**
The following is the struct type used to define the I/O element
y is the row position on the screen.
x is the column position on the screen.
len is the size of the data field in bytes.
flags is the indicator of manadatory data fields. '1' means required.
dptr is the pointer to the data.
fptr is the pointer to the read funtion for the type of data dptr
points to.
**/
typedef struct {
                int             y;
                int             x;
                int             len;
                int             flags;
                int             *dptr;
                int             (*fptr) ();
}               io_elem;
/*
 * Temporary structures for storing data that needs manipulation before
 */
int             int_h_amount;
```

```
/* All the possible messages to print out */
const static char     MANDATORY_MSG[] =
"\033[24;1H\033[mMandatory data field! Please enter data.";
const static char     INVALID_MSG[] =
"\007\033[24;1HAn invalid character was entered. Please enter again.";
const static char     ERASE_MSG[] = "\033[24;1H\033[K\033[4m";
const static char     MIN1DIGIT_MSG[] = "\033[24;1H\033[mYou must enter atleast
 1 digit. Please reenter.\033[4m\1";
const static char     BAD_INPUTS[] = " #### Bad input data was entered -- Select
 again #### \1";
const static char     INCOMPLINE_MSG[] = "\033[24;1H\033[mOrder line is
 incomplete. Please complete the whole line.\033[4m\1";
const static char     ID_OR_LAST_MSG[] = "\033[24;1H\033[mYou must enter
 either the Last Name or the Customer Number.\033[4m\1";
const static char     EXC_MAX_LFT_DEC_DGT_MSG[] =
 "\033[24;1H\033[mMaximum digits left of decimal point already entered. '.'
 expected\033[4m\1";
const static char     EXC_FLD_LIM_MSG[] = "\007\033[24;1H\033[mMaximum
 digits already entered. Tab or <CR> expected\033[4m\1"; /* jr */
const static char     EXECUTION_STATUS_MSG[] = "\033[m\033[22;18HItem
 number is not valid";
const static char     DELIVERY_QUEUED_MSG[] = "\033[m\033[6;19HDelivery
 has been queued";
int           read_integer(int, int, int, int, int *);
int           read_money(int, int, int, int, float *);
int           read_string(int, int, int, int, char *);
char          menu_buf[] = "\033[H\033[J\033[mNew-Order(n)  Payment(p)
Order-Status(o)  Delivery(d)  Stock-Level(s)  Exit(e)";

int                         menu_buflen = sizeof (menu_buf);
io_elem       neworder_inputs[] = {
              /* y x len flags ptr to data ptr to read function */
              /* - - --- ----- ----------- ------------------- */
              2, 29, 2, 0, 0, &read_integer,
              3, 12, 4, 0, 0, &read_integer,
              7, 3, 4, 0, 0, &read_integer,
              7, 10, 6, 0, 0, &read_integer,
              7, 45, 2, 0, 0, &read_integer,
              8, 3, 4, 0, 0, &read_integer,
              8, 10, 6, 0, 0, &read_integer,
              8, 45, 2, 0, 0, &read_integer,
              9, 3, 4, 0, 0, &read_integer,
              9, 10, 6, 0, 0, &read_integer,
              9, 45, 2, 0, 0, &read_integer,
              10, 3, 4, 0, 0, &read_integer,
              10, 10, 6, 0, 0, &read_integer,
              10, 45, 2, 0, 0, &read_integer,
              11, 3, 4, 0, 0, &read_integer,
              11, 10, 6, 0, 0, &read_integer,
              11, 45, 2, 0, 0, &read_integer,
              12, 3, 4, 0, 0, &read_integer,
              12, 10, 6, 0, 0, &read_integer,
              12, 45, 2, 0, 0, &read_integer,
              13, 3, 4, 0, 0, &read_integer,
              13, 10, 6, 0, 0, &read_integer,
              13, 45, 2, 0, 0, &read_integer,
              14, 3, 4, 0, 0, &read_integer,
              14, 10, 6, 0, 0, &read_integer,
              14, 45, 2, 0, 0, &read_integer,
              15, 3, 4, 0, 0, &read_integer,
              15, 10, 6, 0, 0, &read_integer,
              15, 45, 2, 0, 0, &read_integer,
              16, 3, 4, 0, 0, &read_integer,
              16, 10, 6, 0, 0, &read_integer,
              16, 45, 2, 0, 0, &read_integer,
              17, 3, 4, 0, 0, &read_integer,
              17, 10, 6, 0, 0, &read_integer,
              17, 45, 2, 0, 0, &read_integer,
              18, 3, 4, 0, 0, &read_integer,
              18, 10, 6, 0, 0, &read_integer,
              18, 45, 2, 0, 0, &read_integer,
              19, 3, 4, 0, 0, &read_integer,
              19, 10, 6, 0, 0, &read_integer,
              19, 45, 2, 0, 0, &read_integer,
              20, 3, 4, 0, 0, &read_integer,
              20, 10, 6, 0, 0, &read_integer,
              20, 45, 2, 0, 0, &read_integer,
              21, 3, 4, 0, 0, &read_integer,
              21, 10, 6, 0, 0, &read_integer,
              21, 45, 2, 0, 0, &read_integer,
              999
};
io_elem       payment_inputs[] = {
              /* y x len flags ptr to data ptr to read function */
              /* - - --- ----- ----------- ------------------- */
              4, 52, 2, 0, 0, &read_integer,
              9, 11, 4, 0, 0, &read_integer,
              9, 33, 4, 0, 0, &read_integer,
              9, 54, 2, 0, 0, &read_integer,
              10, 29, 16, 0, 0, &read_string,
              15, 24, 7, 0, 0, &read_integer,
              999
};
io_elem       ordstat_inputs[] = {
              /* y x len flags ptr to data ptr to read function */
              /* - - --- ----- ----------- ------------------- */
              2, 29, 2, 0, 0, &read_integer,
              3, 11, 4, 0, 0, &read_integer,
              3, 44, 16, 0, 0, &read_string,
              999
};
io_elem       delivery_inputs[] = {
              /* y x len flags ptr to data ptr to read function */
              /* - - --- ----- ----------- ------------------- */
              4, 17, 2, 0, 0, &read_integer,
              999
};
io_elem       stocklev_inputs[] = {
              /* y x len flags ptr to data ptr to read function */
              /* - - --- ----- ----------- ------------------- */
              4, 24, 2, 0, 0, &read_integer,
              999
};
io_elem       wd_inputs[] = {
              /* y x len flags ptr to data ptr to read function */
              /* - - --- ----- ----------- ------------------- */
              2, 16, 4, 0, 0, &read_integer,
              2, 43, 4, 0, 0, &read_integer,
              999
};
```

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

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

char        output_screen[SCRBUF_LEN];

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

clientlog.c

```
/********************************************************** **********************
****************************** clientlog.c ***********************************
********************************************************** **********************/
/*
 * ** clientlog.c -- Routine for writing out messages form client processes - *
 * useful for detailed error reporting and for debugging
 */

#include <stdio.h>
#include <stdarg.h>
#define BACKTAB 2                   /** Decided to use the CTRL B for now **/
```

```
#define DELETE 127
#define ESCAPE 27
#define LF 10
#define QUIT 3                                  /** CNTRL-C Key stroke to quit for
 now **/
#define SPACE 32
#define SUBMIT 13                       /** Done with screen and submit key: CR **/
#define TAB 9
#define RTE_SYNCH_CHARACTER '\1'

static FILE    *clientlog;
static int      Clog_open = 0;
void
Clog(char *fmt,...)
{
        char            tmpfname[256];
        char            fname[100] ;
        va_list         argp;
        if (!Clog_open) {

#ifdef          NULL_TRANS
        sprintf(fname,"%s/%s.%d",getenv("TMPDIR"),"CLIENTLOG", getpid());
#else
        sprintf(fname,"%s/%s",getenv("TMPDIR"),"CLIENTLOG");

#endif

                clientlog = fopen(fname, "w");
                Clog_open = 1;
        }
        va_start(argp, fmt);
        vfprintf(clientlog, fmt, argp);
        va_end(argp);
        fflush(clientlog);
}
void
SCREENlog(int *flag, char *screen)
{
        char            fname[100];
        int             i, char_ct;
        if (!Clog_open) {
                sprintf(fname, "%s/%s.%d", getenv("TMPDIR"),
 "CLIENTLOG",
                        getpid());
                clientlog = fopen(fname, "w");
                Clog_open = 1;
        }
        fprintf(clientlog, "** %d **\n", flag);
        char_ct = 0;
        fprintf(clientlog, "SCR: ");
        for (i = 0; screen[i] != 0; char_ct++, i++) {
                switch (screen[i]) {
                case BACKTAB:
                        fprintf(clientlog, "<BACKTAB>");
                        break;
                case DELETE:
                        fprintf(clientlog, "<DEL>");
                        break;
                case ESCAPE:
                        fprintf(clientlog, "<ESC>");
                        break;
                case LF:
                        fprintf(clientlog, "<LF>");
                        break;
                case QUIT:
                        fprintf(clientlog, "<^C>");
                        break;
                case SUBMIT:
                        fprintf(clientlog, "<CR>");
                        break;
                case TAB:
                        fprintf(clientlog, "<TAB>");
                        break;
                case RTE_SYNCH_CHARACTER:
                        fprintf(clientlog, "<^A>");
                        break;
                default:
                        fprintf(clientlog, "%c", screen[i]);
                }
                if (char_ct > 192) {
                        char_ct = 0;
                        /* fprintf(screenlog,"\n"); */

                }
        }
        fprintf(clientlog, "\n");
        fflush(clientlog);
}

void
syserr(msg)                                     /* print system call error message
 and
                                                 * terminate */
        char          *msg;
{
        extern int      errno, sys_nerr;
        extern char    *sys_errlist[];
        extern char     tty_name[];
        fprintf(stderr, "\007ERROR: (%s) %s (%d", tty_name, msg, errno);
        if (errno > 0 && errno < sys_nerr)
                fprintf(stderr, ";%s)\n", sys_errlist[errno]);
        else
                fprintf(stderr, ")\n");
        exit(1);
}
```

monitor.c

```
/************************************************** ********************
************************** monitor.c ***************************************
************************************************** ********************/
/* ** monitor.c -- All functions for Tuxedo call and return */
#include <stdio.h>
#include <stdarg.h>
#include "tpcc_client.h"
#include <atmi.h>
#include "tpcc_tux.h"

const char          *svc_names[] = {"NEWO", "PAYM", "ORDS", "DEL", "STOCK"};
int
Snd_Txn_To_Monitor(int txn_type)
```

```
{

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

        if (txn_type == DELIVERY) {

                if ( tpacall((char *)svc_names[txn_type], tuxibuf, ilen,
 TPNOREPLY) == -1){

                        Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
                            svc_names[txn_type], tpstrerror(tperrno));
                        return (-100);
                }
                return(0);
        } else {

                if (tpcall((char *)svc_names[txn_type],
                        (char *)tuxibuf, ilen,
                                &tuxobuf, &olen, 0) == -1){
                        Clog("ERR: Tuxedo tpcall(%s) failed \n\t%s",
                                svc_names[txn_type],
 tpstrerror(tperrno));

                        return(-100);
                }
                return(0);
        }
}

int Init_Monitor()
{
        char    *text;
        ilen = sizeof(struct io_tpcc);
        olen = sizeof(struct io_tpcc);
        if (tpinit(NULL) == -1) {
                tpmerror("tpinit", tperrno);
                return -1;
        }
        if ((tuxibuf = tpalloc("CARRAY", NULL, ilen)) == NULL) {
                tpmerror("tpalloc", tperrno);
                return (-1);
        }
        if ((tuxobuf = tpalloc("CARRAY", NULL, ilen)) == NULL) {
                tpmerror("tpalloc", tperrno);
                return (-1);
        }
        return (NULL);
}
Rundown_Monitor()
{
        int     status;

        tpfree(tuxibuf);
        status = tpterm();

#ifdef DEBUG
        Clog("terminated Tuxedo connection with status %d\n", status);
#endif
}
tpmerror(char *service_called, int errnum)
{
        char    errmsg[256];
        fprintf(stderr, "\033[24;1H\033[mTUXEDO: Failed %s with error: %s\n",
                service_called, tpstrerror(errnum));
        fprintf(stderr, "\n");
}
#ifdef DEBUG
print_input_data(int type)
{
        int     i;
        time_t  the_time;
        the_time = time(&the_time);
        Clog("DBG:=============TIME: %s == == == == == ==
 ==\n",ctime(&the_time));
            switch (type) {
        case NEWORDER:
                Clog("DBG: NEWORDER INPUTS at %s\n",
 ctime(&the_time));
                Clog("DBG: w_id: %d, d_id: %d, c_id: %d o_ol_cnt: %d\n",
                    iNO->w_id, iNO->d_id, iNO->c_id, iNO->o_ol_cnt);
                for (i = 0; i < iNO->o_ol_cnt; i++)
                Clog("DBG: ol_i_id: %d, ol_supply_w_id: %d, ol_quantity:
 %d \n ",iNO->o_ol[i].ol_i_id, iNO->o_ol[i].ol_supply_w_id,iNO->o_ol[i].ol_quantity);
                break;
        case PAYMENT:
                Clog("DBG: PAYMENT INPUTS at %s \n
 ",ctime(&the_time));
                    Clog("DBG: w_id: %d, d_id: %d\n", iPT->w_id,
 iPT->d_id);
                Clog("DBG: c_last: %s ", iPT->c_last);
                Clog(" c_id: %d", iPT->c_id);
                Clog(" c_w_id: %d, c_d_id: %d\n", iPT->c_w_id,
 iPT->c_d_id);
                Clog("DBG: h_amount: %f\n", iPT->h_amount);
                break;
        case ORDSTAT:
                Clog("DBG: ORDER STATUS INPUTS at %s \n
 ",ctime(&the_time));
                    Clog("DBG: w_id: %d, d_id: %d\n", iOS->w_id,
 iOS->d_id);
                Clog("DBG: c_id: %d, c_last: %s\n",
                    iOS->c_id, iOS->c_last);
                break;
        case DELIVERY:
                Clog("DBG: DELIVERY INPUTS at %s\n",
 ctime(&the_time));
                Clog("DBG: w_id: %d, o_carrier_id: %d\n", iDY->w_id, iDY
 ->o_carrier_id);
                break;
        case STOCKLEV:
                Clog("DBG: STOCK LEVEL INPUTS at %s \n
 ",ctime(&the_time));
                Clog("DBG: w_id: %d, d_id: %d, threshold: %d\n", iSL
 ->w_id, iSL->d_id,iSL->threshold);
                break;
other:
                Clog("DBG: Txn_type = %d is illegal at %s
```

```
 \n",type,ctime(&the_time));
                                }
        return;
}
#endif                                          /* ifdef DEBUG */
```

monitor.h

```
/*******************************************************************************
****************************** monitor.h ***************************************
*******************************************************************************/
/* ** monitor.h -- All Tuxedo definitions and storage ** */
long        ilen;
long        olen;
int tty_in;
int tty_out;

/*
 * Data is returned in tuxibuf. Some #defines are set up here for easy
 * access/naming of the input and output areas
 */
char        *tuxibuf;
char        *tuxobuf;
extern void     Clog(char *,...);
#define oNO (&((info_t *) tuxobuf)->neworder)
#define oPT (&((info_t *) tuxobuf)->payment)
#define oOS (&((info_t *) tuxobuf)->ordstat)
#define oDY (&((info_t *) tuxobuf)->delivery)
#define oSL (&((info_t *) tuxobuf)->stocklev)
#define iNO (&((info_t *) tuxibuf)->neworder)
#define iPT (&((info_t *) tuxibuf)->payment)
#define iOS (&((info_t *) tuxibuf)->ordstat)
#define iDY (&((info_t *) tuxibuf)->delivery)
#define iSL (&((info_t *) tuxibuf)->stocklev)
#define iWD (&((info_t *) tuxibuf)->wd)
```

Transaction Source

makefile

```
#
# Copyright (c) 1995 by Sun Microsystems, Inc.
#
# ident "@(#)Makefile.73                1.2             96/03/12     SMI"
#
# Makefile to create all TPCSO programs for Oracle

GENSRC = $(HOME)/generic/c
TPCCSRC = $(HOME)/generic/TPCC/c
GENFRM = $(HOME)/generic/TPCC/c/tuxforms
ROOTDIR = /dbbench/tuxedo/tuxedo
TUXLIB = $(ROOTDIR)/lib
TUXINC = $(ROOTDIR)/include
CFLAGS= -O -I. -I$(GENFRM) -I$(GENSRC) -I$(TPCCSRC) -I$(TUXINC)
 -I$(ORACLE_HOME)/rdbms/demo -DTELNET -DVER73 -DPTY
 -D_SVID_GETTOD -DWEIGHTED_DISTRIBUTION
# The next line is required to compile embedeed PL/SQL
SQLCHECK="SQLCHECK=SEMANTICS"
USERID="USERID=tpcc/tpcc"
BIN = $(HOME)/vendors/oracle/TPCC/bin

REXECS = tpcc_master_rte tpcc_slave_rte tpcc_report tpcc_trigger tpcc_monitor
SEXECS = tpcc_srv_newo tpcc_srv_paym tpcc_srv_ords tpcc_srv_del \
                                                tpcc_srv_stock
CEXECS = tpcc_tuxclient
EXECS = $(REXECS) $(SEXECS) $(CEXECS)

all:    all_client all_rte all_servers
all_servers: $(SEXECS)
all_client: $(CEXECS)
all_rte: $(REXECS)

install: all
        cp $(EXECS) $(BIN)

clean:
        rm -f $(EXECS) *.o $(BIN)/$(EXECS)

CC=cc

# Oracle generating/linking libs/utils

PROC=$(ORACLE_HOME)/bin/proc
PCCINCLUDE= include=$(GENSRC) include=$(GENFRM) include=$(TUXINC)
 include=$(ORACLE_HOME)/rdbms/demo

PROCFLAGS= $(PCCINCLUDE) ireclen=132 oreclen=132 select_error=no
 $(SQLCHECK) $(USERID)

ORACLEBIN = $(ORACLE_HOME)/bin
LIBHOME= $(ORACLE_HOME)/lib

#EPCNI = $(ORACLE_HOME)/rdbms/lib/epcni.o
EPCNI =
LIBPLS= $(LIBHOME)/libpls.a
#LIBORA = $(LIBHOME)/libora.a $(LIBHOME)/libnlsrtl3.a $(LIBHOME)/libc3v6.a
 $(LIBCORE)
LIBORA = $(LIBHOME)/libclient.a $(LIBHOME)/libncr.a $(LIBHOME)/libgeneric.a
 $(LIBHOME)/libcommon.a $(LIBHOME)/libepc.a
 $(LIBHOME)/libnlsrtl3.a $(LIBHOME)/libc3v6.a $(LIBCORE)
LIBSQL = $(LIBHOME)/libsql.a
LIBKNL= $(LIBHOME)/libknl.a
LIBCGEN= $(LIBHOME)/proc/lib/libcgen.a
LIBSQLNET= $(LIBHOME)/libsqlnet.a
LIBCORE=$(LIBHOME)/libcore3.a $(LIBHOME)/libnlsrtl3.a
LIBNTP=$(LIBHOME)/libntp.a
NETLIBS = $(LIBHOME)/libsqlnet.a
PCCINC = $(LIBHOME)/c/lib
STLIBS= $(LIBHOME)/osntabst.o $(LIBHOME)/config.o
LIBOCIC =  $(LIBHOME)/libclient.a
OCILDLIBS = $(LIBOCIC) $(NETLIBS) $(LIBORA) $(LIBSQL) $(LIBSQLNET)
 $(LIBORA) $(LIBSQL) $(EPCNI)
.SUFFIXES:              .exe .o .c .pc

.pc.c:
        $(PROC) $(PROCFLAGS) iname=$*.pc

.pc.o:
        $(PROC) $(PROCFLAGS) iname=$*.pc ; $(CC) $(CFLAGS) -c $*.c
```

```
.c.o:
	$(CC) $(CFLAGS) -c $*.c

##############
#CLIENT side
##############
OBJS = tpcc_client.o tpcc_forms.o tpcc_term.o tpcc_newo.o tpcc_paym.o
 tpcc_ords.o tpcc_stock.o tpcc_del.o

tpcc_tuxclient: $(OBJS)
	CC=$(CC) CFLAGS="$(CFLAGS)  -ldl -Bstatic -lc -lw -lintl"
 ROOTDIR="$(ROOTDIR)" \
	$(ROOTDIR)/bin/buildclient -v -o tpcc_tuxclient -f "$(OBJS)" -l
 "-lcurses"

tpcc_forms.o:	$(GENFRM)/tpcc_forms.c $(GENFRM)/tpcc_client.h
 $(TPCCSRC)/tpcc.h
	$(CC) -c $(CFLAGS) $(GENFRM)/tpcc_forms.c

tpcc_term.o:	$(GENFRM)/tpcc_term.c $(GENFRM)/tpcc_term.h
	$(CC) -c $(CFLAGS) $(GENFRM)/tpcc_term.c

tpcc_client.o:	$(GENFRM)/tpcc_client.c $(GENFRM)/tpcc_client.h
	$(CC) -c $(CFLAGS) $(GENFRM)/tpcc_client.c

tpcc_newo.o:		$(GENFRM)/tpcc_newo.c
  $(GENFRM)/tpcc_client.h
	$(CC) -c $(CFLAGS) $(GENFRM)/tpcc_newo.c
tpcc_paym.o:		$(GENFRM)/tpcc_paym.c
 $(GENFRM)/tpcc_client.h
	$(CC) -c $(CFLAGS) $(GENFRM)/tpcc_paym.c
tpcc_ords.o:		$(GENFRM)/tpcc_ords.c
 $(GENFRM)/tpcc_client.h
	$(CC) -c $(CFLAGS) $(GENFRM)/tpcc_ords.c
tpcc_del.o:	$(GENFRM)/tpcc_del.c $(GENFRM)/tpcc_client.h
	$(CC) -c $(CFLAGS) $(GENFRM)/tpcc_del.c
tpcc_stock.o:		$(GENFRM)/tpcc_stock.c
 $(GENFRM)/tpcc_client.h
	$(CC) -c $(CFLAGS) $(GENFRM)/tpcc_stock.c

#################
# SERVER side
#################

tpcc_srv_newo:	tpcc_srv_newo.o ora_errrpt.o
	CC=$(CC) CFLAGS="$(CFLAGS)" \
	$(ROOTDIR)/bin/buildserver -v -o tpcc_srv_newo -s NEWO \
	-f tpcc_srv_newo.o -f ora_errrpt.o \
	-f '-L$(LIBDIR) -L${ROOTDIR}/lib' \
		-l '$(OCILDLIBS) $(CLIBS)' \
		-l '-lnsl -lsocket -lm -lc -ldl'

tpcc_srv_paym:	tpcc_srv_paym.o ora_errrpt.o
	CC=$(CC) CFLAGS="$(CFLAGS)" \
	$(ROOTDIR)/bin/buildserver -v -o tpcc_srv_paym -s PAYM \
	-f tpcc_srv_paym.o -f ora_errrpt.o \
	-f '-L$(LIBDIR) -L${ROOTDIR}/lib' \
		-l '$(OCILDLIBS) $(CLIBS)' \
		-l '-lnsl -lsocket -lm -lc'

tpcc_srv_ords:	tpcc_srv_ords.o ora_errrpt.o
	CC=$(CC) CFLAGS="$(CFLAGS)" \
	$(ROOTDIR)/bin/buildserver -v -o tpcc_srv_ords -s ORDS \
		-f tpcc_srv_ords.o -f ora_errrpt.o \
		-f '-L$(LIBDIR) -L${ROOTDIR}/lib' \
		-l '$(OCILDLIBS) $(CLIBS)' \
		-l '-lnsl -lsocket -lm -lc'

tpcc_srv_del:	tpcc_srv_del.o ora_errrpt.o
	CC=$(CC) CFLAGS="$(CFLAGS)" \
	$(ROOTDIR)/bin/buildserver -v -o tpcc_srv_del -s DEL \
		-f tpcc_srv_del.o -f ora_errrpt.o \
		-f '-L$(LIBDIR) -L${ROOTDIR}/lib' \
		-l '$(NETLIBS) $(LIBORA) $(LIBSQL) $(NETLIBS) \
$(LIBCORE) \
			$(LIBORA) $(LIBSQL) $(NETLIBS)
			$(NETLIBS) $(LIBORA) $(EPCNI)' \
		-l '-lnsl -lsocket -lm -lc -ldl'

tpcc_srv_stock: tpcc_srv_stock.o
	CC=$(CC) CFLAGS="$(CFLAGS)" \
	$(ROOTDIR)/bin/buildserver -v -o tpcc_srv_stock -s STOCK \
		-f tpcc_srv_stock.o \
		-f '-L$(LIBDIR) -L${ROOTDIR}/lib' \
		-l '$(NETLIBS) $(LIBORA) $(LIBSQL) $(NETLIBS) \
			$(LIBORA) $(LIBSQL) $(NETLIBS)
$(LIBCORE) \
			$(NETLIBS) $(LIBORA) $(NETLIBS) $(EPCNI)'
\
		-l '-lnsl -lsocket -lm -lc -ldl'

##########
# RTE side
##########
tpcc_master_rte: tpcc_master_rte.o tpcc_gettime.o
	$(CC) $(CFLAGS) -Bstatic -o tpcc_master_rte tpcc_master_rte.o
 tpcc_gettime.o

tpcc_slave_rte: tpcc_slave_rte.o tpcc_common.o tpcc_gettime.o
	$(CC) $(CFLAGS) -ldl -Bstatic -o tpcc_slave_rte tpcc_slave_rte.o
 tpcc_gettime.o tpcc_common.o -lsocket -lnsl -lm -lintl

tpcc_report: tpcc_report.o
	$(CC) $(CFLAGS) -o tpcc_report tpcc_report.o

tpcc_trigger: tpcc_trigger.o
	$(CC) $(CFLAGS) -o tpcc_trigger tpcc_trigger.o

tpcc_monitor: tpcc_monitor.o tpcc_gettime.o
	$(CC) $(CFLAGS) -o tpcc_monitor tpcc_monitor.o tpcc_gettime.o

tpcc_master_rte.o: $(TPCCSRC)/tpcc_master_rte.c $(TPCCSRC)/tpcc_rte.h
	$(CC) $(CFLAGS) -c $(TPCCSRC)/tpcc_master_rte.c
tpcc_slave_rte.o : $(TPCCSRC)/tpcc_slave_rte.c $(TPCCSRC)/tpcc_rte.h
 $(TPCCSRC)/tpcc.h
	$(CC) $(CFLAGS) -c $(TPCCSRC)/tpcc_slave_rte.c
tpcc_common.o : $(TPCCSRC)/tpcc_common.c $(TPCCSRC)/tpcc_rte.h
 $(TPCCSRC)/tpcc.h
	$(CC) $(CFLAGS) -c $(TPCCSRC)/tpcc_common.c
tpcc_report.o    : $(TPCCSRC)/tpcc_report.c $(TPCCSRC)/tpcc_rte.h
	$(CC) $(CFLAGS) -c $(TPCCSRC)/tpcc_report.c
```

```
tpcc_trigger.o: $(TPCCSRC)/tpcc_trigger.c
        $(CC) $(CFLAGS) -c $(TPCCSRC)/tpcc_trigger.c
tpcc_gettime.o: $(TPCCSRC)/tpcc_gettime.c
        $(CC) $(CFLAGS) -c $(TPCCSRC)/tpcc_gettime.c
tpcc_monitor.o: $(TPCCSRC)/tpcc_monitor.c
        $(CC) $(CFLAGS) -c $(TPCCSRC)/tpcc_monitor.c

##################
# CLEANUP
##################
clean:
        ${RM} *.o *.c
        ${RM} tpcc_srv_paym tpcc_srv_newo tpcc_srv_ords tpcc_srv_del
 tpcc_srv_stock
        ${RM} tpcc_master_rte tpcc_report tpcc_slave_rte tpcc_trigger
 tpcc_tuxclient
```

oci.h

```
#pragma ident "@(#)oci.h        1.1     95/09/14    SMI"

/*=============================================================
=====+
 |      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
 |                OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                   All Rights Reserved                         |
 +=============================================================
=====+
 | FILENAME
 |   tpccpl.h
 | DESCRIPTION
 |   Header file for TPC-C transactions in PL/SQL.
 +=============================================================
=====*/

#ifndef TPCCPL_H
#define TPCCPL_H

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#include <oratypes.h>
#include <ocidfn.h>
#include <ocidem.h>
#if __STDC__
#include <ociapr.h>
#else
#include <ocikpr.h>
#endif

typedef struct cda_def csrdef;
typedef struct cda_def ldadef;

/****
extern ldadef tpclda;
extern csrdef curs;
extern csrdef curd;
extern csrdef curo;
extern csrdef curp;
extern csrdef curn, curn1, curn2, curn3, curn4;
extern unsigned long tpchda[];
*****/


#ifndef DISCARD
# define DISCARD (void)
#endif

#ifndef sword
# define sword int
#endif

#define VER7          2

#define NA            -1     /* ANSI SQL NULL */
#define NLT           1      /* length for string null terminator */
#define DEADLOCK      60     /* ORA-00060: deadlock */
#define NO_DATA_FOUND  1403   /* ORA-01403: no data found */
#define NOT_SERIALIZABLE  8177  /* ORA-08177: transaction not serializable */

#ifndef NULLP
# define NULLP (void *)NULL
#endif /* NULLP */

#define ADR(object) ((ub1 *)&(object))
#define SIZ(object) ((sword)sizeof(object))

typedef char date[24+NLT];
typedef char varchar2;

#define OBNDRV(lda,cursor,sqlvar,progv,progvl,ftype)\
   if (obndrv((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(progvl),(ftype),NA,\
      (sb2 *)0, (text *)0, NA, NA))\
      {errrpt(lda,cursor,sqlvar);return(-1);}\
   else\
      DISCARD 0

#define OBNDRA(lda,cursor,sqlvar,progv,progvl,ftype,indp,alen,arcode)\
   if (obndra((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(progvl),(ftype),NA,\
      (indp),(alen),(arcode),(ub4)0,(ub4*)0,(text*)0,NA,NA))\
      {errrpt(lda,cursor,sqlvar);return(-1);}\
   else\
      DISCARD 0

#define OBNDRAA(lda,cursor,sqlvar,progv,progvl,ftype,indp,alen,arcode,ms,cs)\
   if (obndra((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(progvl),(ftype),NA,\
      (indp),(alen),(arcode),(ub4)(ms),(ub4*)(cs),(text*)0,NA,NA))\
      {errrpt(lda,cursor,sqlvar);return(-1);}\
   else\
      DISCARD 0

#define ODEFIN(lda,cursor,pos,buf,bufl,ftype,scale,indp,fmt,fmtl,fmtt,rlen,rcode)\
   if (odefin((cursor),(pos),(ub1*)(buf),(bufl),(ftype),(scale),(indp),\
      (text*)(fmt),(fmtl),(fmtt),(rlen),(rcode)))\
      {errrpt(lda,cursor,(text *)ftype);return(-1);}\
   else\
      DISCARD 0
#define OEXFET(lda,cursor,nrows,cancel,exact)\
```

```
    if (oexfet((cursor),(nrows),(cancel),(exact)))\
      {if ((cursor)->rc == 1403) DISCARD 0; \
       else if (errrpt(lda,cursor,(text *)"OEXFET")==RECOVERR) \
          {orol(lda);return(RECOVERR);} \
       else{orol(lda);return(-1);}}\
    else\
      DISCARD 0

#define OOPEN(lda,cursor)\
    if (oopen((cursor),(lda),(text*)0,NA,NA,(text*)0,NA))\
      {errrpt(lda,cursor,(text *)"OOPEN");return(-1);}\
    else\
      DISCARD 0

#define OPARSE(lda,cursor,sqlstm,sqll,defflg,lngflg)\
    if (oparse((cursor),(sqlstm),(sb4)(sqll),(defflg),(ub4)(lngflg)))\
      {errrpt(lda,cursor,sqlstm);return(-1);}\
    else\
      DISCARD 0

#define OFEN(lda,cursor,nrows)\
    if (ofen((cursor),(nrows)))\
      {if (errrpt(lda,cursor,(text *)"OFEN")==RECOVERR) \
         {orol(lda);return(RECOVERR);} \
       else{orol(lda);return(-1);}}\
    else\
      DISCARD 0

#define OEXEC(lda,cursor)\
    if (oexec((cursor)))\
      {if (errrpt(lda,cursor,(text *)"OEXEC")==RECOVERR) \
         {orol(lda);return(RECOVERR);} \
       else{orol(lda);return(-1);}}\
    else\
      DISCARD 0

#define OCOM(lda,cursor)\
    if (ocom((lda))) \
      {errrpt(lda,cursor,(text *)"OCOM");orol(lda);return(-1);}\
    else\
      DISCARD 0

#define OEXN(lda,cursor,iters,rowoff)\
    if (oexn((cursor),(iters),(rowoff))) \
      {if (errrpt(lda,cursor,(text *)"OEXN")==RECOVERR) \
         {orol(lda);return(RECOVERR);} \
       else{orol(lda);return(-1);}}\
    else\
      DISCARD 0

#endif
```

ora_errrpt.c

```
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)ora_errrpt.c    1.1        95/09/14    SMI"

/*
 * these functions actually belong in ~dbbench/generic/c/msgh_log.c.  We put them
 * here because they have database specific statements.
 */

#include "ora_err.h"
#include "oci.h"

errrpt(lda, cur, sqlvar)
ldadef *lda;
csrdef *cur;
text        *sqlvar;
{
            text msg[2048];
            if (cur->rc) {
                        oerhms(lda, cur->rc, msg, 2048);
                        userlog("%s sql_variable %s\n", msg, sqlvar);
                        if (cur->rc == DEADLOCK)
                                    return(RECOVERR);
                        else
                                    return(IRRECERR);
            }
}
```

ora_err.h

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#ifndef ORA_ERR_H
#define ORA_ERR_H

#pragma ident "@(#)ora_err.h       1.2        94/12/07    SMI"

/*
 * this is required because Oracle does not provide
 *            symbolic constants in a header file
 */

#define     EDEADLOK        60
#define     SQLNOTFOUND     1403
#define     COLUMN_NULL     -1405
#define     EDUPLICATE      -1
#define     RECOVERR        -10
#define     IRRECERR        -20

#endif ORA_ERR_H
```

tpcc_srv_del.c

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcso_srv_del.pc          1.5        94/12/07    SMI"

/*
 * File: delivery.pc
 * Delivery transaction code for Oracle using Message Handler
```

```
 * This program is different from the other servers, in that it
 * records transaction info in a results file.
 * Author : Shanti S
 * Date   : 4/18/94
 */



#include <stdlib.h>
#include <sys/types.h>
#include <time.h>
#include <sys/signal.h>
#include <sys/utsname.h>
#include <errno.h>
#include <stdio.h>
#include "ora_err.h"

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

#define MOVETO(element, struct_name) element = struct_name -> element
#define MOVEBACK(element, struct_name) struct_name -> element = element

static int w_id;
static int o_carrier_id;

int my_qid, my_id;
char my_name[] = "Del";

static int tx_count = 0;  /* Transaction counter */
static FILE *delfile;

static char outbuf[2048];               /* Buffer for results file */

#if ACID
#include <sys/types.h>
#include <time.h>
time_t curtime, *timep = &curtime;
#endif

get_del_tx_cnt()
{
        return tx_count;
}

#include "oci.h"

ldadef tpclda;
unsigned char cr_date[7];
unsigned long tpchda[256];
csrdef curs;

#define SQLTXT1 "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 1, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 1 AND o_w_id = :w_id AND o_d_id = 1
 AND \
   o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 2, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 2 AND o_w_id = :w_id AND o_d_id = 2
 AND \
   o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 3, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 3 AND o_w_id = :w_id AND o_d_id = 3
 AND \
   o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 4, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 4 AND o_w_id = :w_id AND o_d_id = 4
 AND \
   o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 5, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 5 AND o_w_id = :w_id AND o_d_id = 5
 AND \
   o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 6, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 6 AND o_w_id = :w_id AND o_d_id = 6
 AND \
   o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 7, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 7 AND o_w_id = :w_id AND o_d_id = 7
 AND \
   o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 8, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 8 AND o_w_id = :w_id AND o_d_id = 8
 AND \
   o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 9, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 9 AND o_w_id = :w_id AND o_d_id = 9
 AND \
   o_id = no_o_id AND rownum <= 1 UNION ALL \
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 10, no_o_id,
 new_order.rowid, o_c_id, orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 10 AND o_w_id = :w_id AND o_d_id = 10
 AND \
   o_id = no_o_id AND rownum <= 1"

#define SQLTXT2 "DELETE FROM new_order WHERE rowid = :no_rowid"

#define SQLTXT3 "UPDATE orders SET o_carrier_id = :carrier_id \
   WHERE rowid = :o_rowid"
```

```
#define SQLTXT4 "UPDATE order_line SET ol_delivery_d = :cr_date \
  WHERE ol_w_id = :w_id AND ol_d_id = :d_id AND ol_o_id = :o_id"

#define SQLTXT5 "\
SELECT :d_id1, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND
 \
  ol_d_id = :d_id1 AND ol_o_id = :o_id1 UNION ALL \
SELECT :d_id2, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND
 \
  ol_d_id = :d_id2 AND ol_o_id = :o_id2 UNION ALL \
SELECT :d_id3, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND
 \
  ol_d_id = :d_id3 AND ol_o_id = :o_id3 UNION ALL \
SELECT :d_id4, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND
 \
  ol_d_id = :d_id4 AND ol_o_id = :o_id4 UNION ALL \
SELECT :d_id5, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND
 \
  ol_d_id = :d_id5 AND ol_o_id = :o_id5 UNION ALL \
SELECT :d_id6, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND
 \
  ol_d_id = :d_id6 AND ol_o_id = :o_id6 UNION ALL \
SELECT :d_id7, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND
 \
  ol_d_id = :d_id7 AND ol_o_id = :o_id7 UNION ALL \
SELECT :d_id8, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND
 \
  ol_d_id = :d_id8 AND ol_o_id = :o_id8 UNION ALL \
SELECT :d_id9, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND
 \
  ol_d_id = :d_id9 AND ol_o_id = :o_id9 UNION ALL \
SELECT :d_id10, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id
 AND \
  ol_d_id = :d_id10 AND ol_o_id = :o_id10"

#define SQLTXT6 "UPDATE customer SET c_balance = c_balance + :amt, \
  c_delivery_cnt = c_delivery_cnt + 1 WHERE c_w_id = :w_id AND \
  c_d_id = :d_id AND c_id = :c_id"

#define NDISTS 10
#define ROWIDLEN 20
#define      DEL_DATE_LEN        7


struct delctx {
  sb2 del_o_id_ind[NDISTS];
  sb2 cons_ind[NDISTS];
  sb2 w_id_ind[NDISTS];
  sb2 d_id_ind[NDISTS];
  sb2 c_id_ind[NDISTS];
  sb2 del_date_ind[NDISTS];
  sb2 carrier_id_ind[NDISTS];
  sb2 amt_ind[NDISTS];
  sb2 no_rowid_ind[NDISTS];
  sb2 o_rowid_ind[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  sb2 inum_ind;
#endif

  ub2 del_o_id_len[NDISTS];
  ub2 cons_len[NDISTS];
  ub2 w_id_len[NDISTS];
  ub2 d_id_len[NDISTS];
  ub2 c_id_len[NDISTS];
  ub2 del_date_len[NDISTS];
  ub2 carrier_id_len[NDISTS];
  ub2 amt_len[NDISTS];
  ub2 no_rowid_len[NDISTS];
  ub2 o_rowid_len[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  ub2 inum_len;
#endif

  ub2 del_o_id_rcode[NDISTS];
  ub2 cons_rcode[NDISTS];
  ub2 w_id_rcode[NDISTS];
  ub2 d_id_rcode[NDISTS];
  ub2 c_id_rcode[NDISTS];
  ub2 del_date_rcode[NDISTS];
  ub2 carrier_id_rcode[NDISTS];
  ub2 amt_rcode[NDISTS];
  ub2 no_rowid_rcode[NDISTS];
  ub2 o_rowid_rcode[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  ub2 inum_rcode;
#endif

  int del_o_id[NDISTS];
  int cons[NDISTS];
  int w_id[NDISTS];
  int d_id[NDISTS];
  int c_id[NDISTS];
  int carrier_id[NDISTS];
  /* float amt[NDISTS]; Changed to int */
  int amt[NDISTS];
  char no_rowid[NDISTS][ROWIDLEN];
  char o_rowid[NDISTS][ROWIDLEN];
  unsigned char del_date[NDISTS][DEL_DATE_LEN];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  char inum[10];
#endif
};

typedef struct delctx delctx;
delctx *dctx;
csrdef curd0, curd1, curd2, curd3, curd4, curd5, curd6;


int
init_del_tx()
{
/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

        char stmbuf[4096];
        char bstr1[10], bstr2[10];
        int i;

        dctx = (delctx *) malloc(sizeof(delctx));
```

```
    if (orlon(&tpclda, (ub1 *)tpchda, (text *)"tpcc/tpcc",
        -1, (text *)0, -1, 0)) {
        errrpt(&tpclda, &tpclda, "logon");
        return(1);
    }

    /* turn off auto-commit */
    if (ocof(&tpclda)) {
        errrpt(&tpclda, &tpclda, "auto-commit");
        ologof(&tpclda);
        return(1);
    }
#if VER73
{
   /* run in serializable mode */

   if (oopen(&curs, &tpclda)) {
                        errrpt(&tpclda,&curs,"Open cursor for serializable mode");
                        ologof(&tpclda);
                        return(1);
                        }

    sprintf((char *)stmbuf, "alter session set isolation_level = serializable");

                if (oparse(&curs, stmbuf, NA, FALSE, VER7)) {
                   errrpt(&tpclda,&curs,"Parse for serializable mode");
                   oclose(&curs);
                   ologof(&tpclda);
                   return(1);
                }

                if (oexec(&curs)) {
                        errrpt(&tpclda,&curs,"oexec for serializable mode");
                        orol(&tpclda);
                        oclose(&curs);
                        ologof(&tpclda);
                        return(1);
                        }

                if (oclose(&curs)) {
                        errrpt(&tpclda,&curs,"oclose for serializable mode");
                        return(1);
                        }
}
#endif
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   OOPEN(&tpclda,&curd0);

   sprintf ((char *) stmbuf, SQLTXT0);
   OPARSE(&tpclda,&curd0,stmbuf,NA,FALSE,VER7);

   ODEFIN(&tpclda,&curd0,1,dctx->inum,SIZ(dctx->inum),SQLT_STR,NA,
        &(dctx->inum_ind),NULL,NA,NA,&(dctx->inum_len),&(dctx->inum_rcode));
#endif

   /* open first cursor */

   OOPEN(&tpclda,&curd1);

   sprintf ((char *) stmbuf, SQLTXT1);
   OPARSE(&tpclda,&curd1,stmbuf,NA,FALSE,VER7);

   /* bind variables */

   OBNDRV(&tpclda,&curd1,":w_id",ADR(w_id),SIZ(w_id),SQLT_INT);

   ODEFIN(&tpclda,&curd1,1,dctx->d_id,SIZ(dctx->d_id[0]),SQLT_INT,NA,
        dctx->d_id_ind,NULL,NA,NA,dctx->d_id_len,dctx->d_id_rcode);
   ODEFIN(&tpclda,&curd1,2,dctx->del_o_id,SIZ(dctx->del_o_id[0]),SQLT_INT,NA,
        dctx->del_o_id_ind,NULL,NA,NA,dctx->del_o_id_len,
        dctx->del_o_id_rcode);
ODEFIN(&tpclda,&curd1,3,dctx->no_rowid,SIZ(dctx->no_rowid[0]),SQLT_RID,NA,
        dctx->no_rowid_ind,NULL,NA,NA,dctx->no_rowid_len,
        dctx->no_rowid_rcode);
   ODEFIN(&tpclda,&curd1,4,dctx->c_id,SIZ(dctx->c_id[0]),SQLT_INT,NA,
        dctx->c_id_ind,NULL,NA,NA,dctx->c_id_len,dctx->c_id_rcode);
   ODEFIN(&tpclda,&curd1,5,dctx->o_rowid,SIZ(dctx->o_rowid[0]),SQLT_RID,NA,
        dctx->o_rowid_ind,NULL,NA,NA,dctx->o_rowid_len,
        dctx->o_rowid_rcode);

   /* open second cursor */

   OOPEN(&tpclda,&curd2);

   sprintf ((char *) stmbuf, SQLTXT2);
   OPARSE(&tpclda,&curd2,stmbuf,NA,FALSE,VER7);

   /* bind variables */

   OBNDRA(&tpclda,&curd2,":no_rowid",dctx->no_rowid,SIZ(dctx->no_rowid[0]),
        SQLT_RID,dctx->no_rowid_ind,dctx->no_rowid_len,dctx->no_rowid_rcode);

   /* open third cursor */

   OOPEN(&tpclda,&curd3);

   sprintf ((char *) stmbuf, SQLTXT3);
   OPARSE(&tpclda,&curd3,stmbuf,NA,FALSE,VER7);

   /* bind variables */

   OBNDRA(&tpclda,&curd3,":carrier_id",dctx->carrier_id,
        SIZ(dctx->carrier_id[0]),SQLT_INT,dctx->carrier_id_ind,
        dctx->carrier_id_len,dctx->carrier_id_rcode);
   OBNDRA(&tpclda,&curd3,":o_rowid",dctx->o_rowid,SIZ(dctx->o_rowid[0]),
        SQLT_RID,dctx->o_rowid_ind,dctx->o_rowid_len,dctx->o_rowid_rcode);

   /* open fourth cursor */

   OOPEN(&tpclda,&curd4);

   sprintf ((char *) stmbuf, SQLTXT4);
   OPARSE(&tpclda,&curd4,stmbuf,NA,FALSE,VER7);

   /* bind variables */

   OBNDRA(&tpclda,&curd4,":w_id",dctx->w_id,SIZ(dctx->w_id[0]),SQLT_INT,
        dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
   OBNDRA(&tpclda,&curd4,":d_id",dctx->d_id,SIZ(dctx->d_id[0]),SQLT_INT,
```

```
        dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
OBNDRA(&tpclda,&curd4,":o_id",dctx->del_o_id,SIZ(dctx->del_o_id[0]),SQLT_INT
,
        dctx->del_o_id_ind,dctx->del_o_id_len,dctx->del_o_id_rcode);
  OBNDRA(&tpclda,&curd4,":cr_date",
 dctx->del_date,DEL_DATE_LEN,SQLT_DAT,
                dctx->del_date_ind, dctx->del_date_len,dctx->del_date_rcode);


  /* open fifth cursor */

  OOPEN(&tpclda,&curd5);

  sprintf ((char *) stmbuf, SQLTXT5);
  OPARSE(&tpclda,&curd5,stmbuf,NA,FALSE,VER7);

  /* bind variables */

  OBNDRV(&tpclda,&curd5,":w_id",ADR(w_id),SIZ(w_id),SQLT_INT);
  for (i = 0; i < NDISTS; i++) {
    sprintf (bstr1, ":d_id%d", i + 1);
    sprintf (bstr2, ":o_id%d", i + 1);
    OBNDRA(&tpclda,&curd5,bstr1,ADR(dctx->d_id[i]),SIZ(dctx->d_id[0]),
        SQLT_INT, &(dctx->d_id_ind[i]), &(dctx->d_id_len[i]),
        &(dctx->d_id_rcode[i]));
    OBNDRA(&tpclda,&curd5,bstr2,ADR(dctx->del_o_id[i]),SIZ(dctx->del_o_id[0]),
        SQLT_INT, &(dctx->del_o_id_ind[i]), &(dctx->del_o_id_len[i]),
        &(dctx->del_o_id_rcode[i]));
  }

  ODEFIN(&tpclda,&curd5,1,dctx->cons,SIZ(dctx->cons[0]),SQLT_INT,NA,
      dctx->cons_ind,NULL,NA,NA,dctx->cons_len,dctx->cons_rcode);
  ODEFIN(&tpclda,&curd5,2,dctx->amt,SIZ(dctx->amt[0]),SQLT_INT,NA,
      dctx->amt_ind,NULL,NA,NA,dctx->amt_len,dctx->amt_rcode);

  /* open sixth cursor */

  OOPEN(&tpclda,&curd6);

  sprintf ((char *) stmbuf, SQLTXT6);
  OPARSE(&tpclda,&curd6,stmbuf,NA,FALSE,VER7);

  /* bind variables */

  OBNDRA(&tpclda,&curd6,":amt",dctx->amt,SIZ(dctx->amt[0]),SQLT_INT,
      dctx->amt_ind,dctx->amt_len,dctx->amt_rcode);
  OBNDRA(&tpclda,&curd6,":w_id",dctx->w_id,SIZ(dctx->w_id[0]),SQLT_INT,
      dctx->w_id_ind,dctx->w_id_len,dctx->w_id_rcode);
  OBNDRA(&tpclda,&curd6,":d_id",dctx->d_id,SIZ(dctx->d_id[0]),SQLT_INT,
      dctx->d_id_ind,dctx->d_id_len,dctx->d_id_rcode);
  OBNDRA(&tpclda,&curd6,":c_id",dctx->c_id,SIZ(dctx->c_id[0]),SQLT_INT,
      dctx->c_id_ind,dctx->c_id_len,dctx->c_id_rcode);

#ifdef TKPROF
        EXEC SQL ALTER SESSION SET SQL_TRACE = TRUE;
#endif      /* TKPROF */

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/
        return(0);
}


delivery_tx(rqst)
TPSVCINFO *rqst;
{
  int i, j,v;
  int rpc;
  int invalid;
  int tmp_id;
  /* float tmp_amt; changed form float to int */
  int tmp_amt;
  int del_o_id[10];
  struct req_struct {
        int w_id;
        int o_carrier_id;
        time_t qtime;                   /* Time transaction was queued */
  } *delp = (struct req_struct *)(rqst->data);
  int       len;
  int       retries=0, err = 0;

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

        vgetdate(cr_date);
        MOVETO(w_id, delp);
        MOVETO(o_carrier_id, delp);

        tx_count++;
        sprintf(outbuf, "Starting transaction %d queued at %d\n",
                tx_count, delp->qtime);

#if ACID
        time(timep);
        userlog("ACID DELIVERY Transaction begun at %s\n", ctime(timep));
#endif

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  int hasno;
  int reread;
  char sdate[30];

  oexfet (&curd0, 1, 0, 0);
  sysdate (sdate);
  printf ("Delivery started at %s on node %s\n", sdate, dctx->inum);
#endif

retry:

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  reread = 1;
#endif

iso:
  invalid = 0;
  /* initialization for array operations */

  for (i = 0; i < NDISTS; i++) {
```

```
    dctx->del_o_id_ind[i] = TRUE;
    dctx->cons_ind[i] = TRUE;
    dctx->w_id_ind[i] = TRUE;
    dctx->d_id_ind[i] = TRUE;
    dctx->c_id_ind[i] = TRUE;
    dctx->del_date_ind[i] = TRUE;
    dctx->carrier_id_ind[i] = TRUE;
    dctx->amt_ind[i] = TRUE;
    dctx->no_rowid_ind[i] = TRUE;
    dctx->o_rowid_ind[i] = TRUE;

    dctx->del_o_id_len[i] = SIZ(dctx->del_o_id[0]);
    dctx->cons_len[i] = SIZ(dctx->cons[0]);
    dctx->w_id_len[i] = SIZ(dctx->w_id[0]);
    dctx->d_id_len[i] = SIZ(dctx->d_id[0]);
    dctx->c_id_len[i] = SIZ(dctx->c_id[0]);
    dctx->del_date_len[i] = DEL_DATE_LEN;
    dctx->carrier_id_len[i] = SIZ(dctx->carrier_id[0]);
    dctx->amt_len[i] = SIZ(dctx->amt[0]);
    dctx->no_rowid_len[i] = SIZ(dctx->no_rowid[0]);
    dctx->o_rowid_len[i] = SIZ(dctx->o_rowid[0]);

    dctx->w_id[i] = w_id;
    dctx->carrier_id[i] = o_carrier_id;
    memcpy(dctx->del_date[i],cr_date,DEL_DATE_LEN);
}

/* array select from new_order and orders tables */

if (oexfet (&curd1, NDISTS, 0, 0)) {
  if (curd1.rc == NOT_SERIALIZABLE){
    orol (&tpclda);
    retries++;
    goto retry;
  }
  else if (curd1.rc != NO_DATA_FOUND) {
    if (errrpt (&tpclda, &curd1) == RECOVERR) {
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else {
      orol (&tpclda);
      return (-1);
    }
  }
}

/* mark districts with no new order */

rpc = curd1.rpc;
invalid = NDISTS - curd1.rpc;
for (i = rpc; i < NDISTS; i++) {
  dctx->del_o_id_ind[i] = NA;
  dctx->w_id_ind[i] = NA;
  dctx->d_id_ind[i] = NA;
  dctx->c_id_ind[i] = NA;
  dctx->carrier_id_ind[i] = NA;
  dctx->no_rowid_ind[i] = NA;
  dctx->o_rowid_ind[i] = NA;
}

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  if (invalid) {
    sysdate (sdate);
    for (i = 1; i <= NDISTS; i++) {
      hasno = 0;
      for (j = 0; j < rpc; j++) {
        if (dctx->d_id[j] == i) {
          hasno = 1;
          break;
        }
      }
      if (!hasno)
        printf ("Delivery [dist %d] found no new order at %s\n", i, sdate);
    }
    if (reread) {
      sleep (60);
      sysdate (sdate);
      printf ("Delivery wake up at %s\n", sdate);
      reread = 0;
      goto iso;
    }
  }
#endif

  /* array delete of new_order table */

  if (oexn (&curd2, rpc, 0)) {
    if (curd2.rc == NOT_SERIALIZABLE) {
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else if (errrpt (&tpclda, &curd2) == RECOVERR) {
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else {
      orol (&tpclda);
      return (-1);
    }
  }

  if (curd2.rpc != rpc) {
#ifdef TPCSO
    write_log ("Del %d: %d rows selected, %d rows deleted\n",
          my_id, rpc, curd2.rpc);
#else
    userlog ("%d rows selected, %d rows deleted\n", rpc, curd2.rpc);
#endif
    orol (&tpclda);
    return (-1);
  }

  /* array update of orders table */
  if (oexn (&curd3, rpc, 0)) {
    if (curd3.rc == NOT_SERIALIZABLE) {
      orol (&tpclda);
```

```
      retries++;
      goto retry;
    }
    else if (errrpt (&tpclda, &curd3) == RECOVERR) {
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else {
      orol (&tpclda);
      return (-1);
    }
  }

  if (curd3.rpc != rpc) {
#ifdef TPCSO
    write_log ("Del %d: %d rows selected, %d ords updated\n",
          my_id, rpc, curd3.rpc);
#else
    userlog ("%d rows selected, %d ords updated\n", rpc, curd3.rpc);
#endif
    orol (&tpclda);
    return (-1);
  }

  /* array update of order_line table */

  if (oexn (&curd4, rpc, 0)) {
    if (curd4.rc == NOT_SERIALIZABLE) {
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else if (errrpt (&tpclda, &curd4) == RECOVERR) {
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else {
      orol (&tpclda);
      return (-1);
    }
  }

  /* array select from order_line table */

  if (oexfet (&curd5, rpc, 0, 0)) {
    if (curd5.rc == NOT_SERIALIZABLE){
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else if (curd5.rc != NO_DATA_FOUND) {
      if (errrpt (&tpclda, &curd5) == RECOVERR) {
        orol (&tpclda);
        retries++;
        goto retry;
      }
      else {
        orol (&tpclda);
        return (-1);
      }
    }
  }

  if (curd5.rpc != rpc) {
#ifdef TPCSO
    write_log("Del %d: %d rows selected, %d ordl selected\n",
          my_id, rpc, curd5.rpc);
#else
    userlog( "%d rows selected, %d ordl selected\n", rpc, curd5.rpc);
#endif
    orol (&tpclda);
    return (-1);
  }

  /* reorder amount selected if necessary */

  for (i = 0; i < rpc; i++) {
    if (dctx->cons[i] != dctx->d_id[i]) {
#ifdef TPCSO
      write_log ("Del %d: reordering amount\n", my_id);
#else
      userlog("reordering amount\n");
#endif
      for (j = i + 1; j < rpc; j++) {
        if (dctx->cons[j] == dctx->d_id[i]) {
          tmp_id = dctx->cons[i];
          dctx->cons[i] = dctx->cons[j];
          dctx->cons[j] = tmp_id;
          tmp_amt = dctx->amt[i];
          dctx->amt[i] = dctx->amt[j];
          dctx->amt[j] = tmp_amt;
          break;
        }
      }
      if (j >= rpc) {
#ifdef TPCSO
        write_log("Del %d: missing ordl?\n", my_id);
#else
        userlog("missing ordl?\n");
#endif
        orol (&tpclda);
        return (-1);
      }
    }
  }

#if defined(ISO5) || defined(ISO6)
  printf ("d_id:amount\n");
  for (i = 0; i < rpc; i++)
    printf ("%d:%.2f ", dctx->d_id[i], dctx->amt[i]);
  printf ("\n");
#endif

  /* array update of customer table */

  if (oexn (&curd6, rpc, 0)) {
    if (curd6.rc == NOT_SERIALIZABLE) {
      orol (&tpclda);
```

```
      retries++;
      goto retry;
    }
    else if (errrpt (&tpclda, &curd6) == RECOVERR) {
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else {
      orol (&tpclda);
      return (-1);
    }
  }
  if (curd6.rpc != rpc) {
#ifdef TPCSO
    write_log ("Del %d: %d rows selected, %d cust updated\n",
          my_id, rpc, curd6.rpc);
#else
    userlog ("%d rows selected, %d cust updated\n", rpc, curd6.rpc);
#endif
    orol (&tpclda);
    return (-1);
  }

#if defined(ISO5) || defined(ISO6)
  sysdate (sdate);
#ifdef ISO5
  printf ("Delivery sleep before commit at %s\n", sdate);
#else
  printf ("Delivery sleep before abort at %s\n", sdate);
#endif
  sleep (60);
  sysdate (sdate);
  printf ("Delivery wake up at %s\n", sdate);
#endif

#ifdef ISO6
  orol (&tpclda);
#else
  OCOM(&tpclda,&tpclda);
#endif

#if defined(ISO5) || defined(ISO6)
  sysdate (sdate);
  printf ("Delivery completed at: %s\n", sdate);
#endif

  /* return o_id's in district id order */

  for (i = 0; i < NDISTS; i++)
    del_o_id[i] = 0;
  for (i = 0; i < rpc; i++)
    del_o_id[dctx->d_id[i] - 1] = dctx->del_o_id[i];

        for (i = 0; i < 10; i++) {
            if (del_o_id[i] == 0) {
                /* No order found for this district */
                    sprintf(outbuf+strlen(outbuf),
                    "Delivery for District %d skipped\n",
i+1);
            }
            else {
            sprintf(outbuf+strlen(outbuf),
            "Delivered order %d for district %d, warehouse %d, carrier
%d\n",
                    del_o_id[i], i+1, w_id, o_carrier_id);
            }
        }

time(0));   sprintf(outbuf+strlen(outbuf), "Transaction completed at %d\n",
        fwrite(outbuf, strlen(outbuf), 1, delfile);
        fflush(delfile);

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/
#if ACID
        time(timep);
ctime(timep));  userlog("ACID DELIVERY Transaction completed at %s\n",
#endif

        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct req_struct), 0);
        return(0);
}



/* Tuxedo */
tpsvrinit(argc, argv)
char **argv;
{
        char *p;
        char filename[200];
        int proc_no=0, count;
        struct utsname name;

        if ((p = getenv("TMPDIR")) == (char *)NULL) {
                userlog("TMPDIR environment variable not set\n");
                exit(1);
        }

proc_no */  proc_no = atoi(argv[4]);          /* Needs argument which is the

        /* Get hostname of our machine  and create results file */
        uname( &name);
        strcpy(filename, p);
proc_no);   sprintf(filename+strlen(filename), "/%s.del%d", name.nodename,
        delfile = fopen(filename, "w");
        if (delfile == NULL) {
                userlog("Cannot create file %s\n", filename);
        }
        return(init_del_tx());          /* Prepare transaction */
}

void
tpsvrdone()
{
        oclose(&curd0);
        oclose(&curd1);
        oclose(&curd2);
        oclose(&curd3);
```

```
        oclose(&curd4);
        oclose(&curd5);
        oclose(&curd6);
        fclose(delfile);        /* Close results file */
}

DEL(rqst)
TPSVCINFO *rqst;
{
        delivery_tx(rqst);
}

vgetdate(unsigned char *buf)
{
  time_t tloc;
  char temp[5];
  int cen;

  if(time(&tloc) == (time_t)-1) {
    printf("Error getting date\n");
    exit(1);
  }
  cftime(temp,"%d",&tloc);
  buf[3] = (unsigned char)atoi(temp);
  cftime(temp,"%m",&tloc);
  buf[2] = (unsigned char)atoi(temp);
  cftime(temp,"%Y",&tloc);
  cen = atoi(temp);
  buf[0] = (unsigned char)((cen/100)+100);
  buf[1] = (unsigned char)((cen%100)+100);
  cftime(temp,"%H",&tloc);
  buf[4] = (unsigned char)(atoi(temp) + 1);
  cftime(temp,"%M",&tloc);
  buf[5] = (unsigned char)(atoi(temp) + 1);
  cftime(temp,"%S",&tloc);
  buf[6] = (unsigned char)(atoi(temp) + 1);
}
```

tpcc_srv_stock.pc

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcc_srv_stock.pc        1.3        94/12/07    SMI"

/*
 * File: stock.ec
 * Stock-level transaction code for Informix Tuxedo
 * Author : Shanti S
 * Date   : 84/93
 * Ported to Oracle 8/26/94 - dbt
 *
 */

#define SQLNET

EXEC SQL INCLUDE sqlca;
EXEC SQL INCLUDE sqlda;
#include <stdio.h>

static int     tx_count=0;

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

#define MOVETO(element, struct_name)        \
        element = struct_name -> element
#define MOVEBACK(element, struct_name)    \
        struct_name -> element = element

/* List of fields in stock */
/* This structure should be EXACTLY identical to the one declared in client.h */
/* List of fields in stock-level */
struct stock_inf {
        int             w_id;
        int d_id;
        int threshold;
        int low_stock;
};

struct stock_inf *stocklevel;        /* Input to stocklevel transaction */


/*
 * Initialize transaction
 */
int
init_stock_tx()
{

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

        EXEC SQL BEGIN DECLARE SECTION;
        char        *db = "tpcc";
        EXEC SQL END DECLARE SECTION;

        EXEC SQL CONNECT :db IDENTIFIED BY :db;

        if (sqlca.sqlcode != 0) {
                initerr("database tpcc");
                return(1);
        }

#ifdef TKPROF
        EXEC SQL ALTER SESSION SET SQL_TRACE = TRUE;
#endif

        /*proc_stat_msg("init_stock_tx()\n");
        proc_stat(); */

        return(0);

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/
```

```
}


/*
 * Function: do stocklevel transaction
 * Input is the stocklevel structure. Output is low_stock field
 */
stocklevel_tx(rqst)
TPSVCINFO *rqst;
{

        EXEC SQL BEGIN DECLARE SECTION;

    int next_order;
    int old_order;

    int w_id;
    int d_id;
    int threshold;
    int low_stock;
                int err;

    EXEC SQL END DECLARE SECTION;

        struct stock_inf        *stocklevel;
        stocklevel = (struct stock_inf *)(rqst->data);

    MOVETO(w_id, stocklevel);
    MOVETO(d_id, stocklevel);
    MOVETO(threshold, stocklevel);

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

        tx_count++;

        EXEC SQL EXECUTE
        BEGIN
                stocklevel.getstocklevel(:w_id, :d_id, :threshold,
:low_stock);
        END;
        END-EXEC;
        if (sqlca.sqlcode != 0) {
                sqerr("Stocklevel");
                return(1);
        }

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/

    MOVEBACK(low_stock, stocklevel);

        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct stock_inf), 0);
}

/* Tuxedo */
tpsvrinit(argc, argv)
char **argv;
{
        return(init_stock_tx());        /* Prepare transaction */
}

void
tpsvrdone()
{
}


STOCK(rqst)
TPSVCINFO *rqst;
{
        stocklevel_tx(rqst);
}


sqerr(str)
char *str;
{
%s\n", str,     userlog("SQL ERROR during %s, sqlcode = %d, ISAM code = %d,
                sqlca.sqlcode, sqlca.sqlerrd[1], sqlca.sqlerrm);
        EXEC SQL ROLLBACK WORK;
        tpreturn(TPFAIL, 0, (char *)0, 0, 0);
}


initerr(str)
char *str;
{
%s\n", str,     userlog("SQL ERROR during %s, sqlcode = %d, ISAM code = %d,
                sqlca.sqlcode, sqlca.sqlerrd[1], sqlca.sqlerrm);
        EXEC SQL ROLLBACK WORK;
}
```

tpcc_srv_newo.c

```
#define     REAL_ORACLE 1       /* Compile with connections to RDMS */
/*
 * Copyright (c) 1995 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcc_srv_newo.c          1.15        96/01/11    SMI"

/*
 * File: newo.ec
 * Neworder transaction code for Informix using Tuxedo
 * Author : Shanti S
 * Date   : 8/03/93
 * Ported to Oracle by dbt 8-30-94
 *
 */

#include "oci.h"

#include <stdio.h>
#include <errno.h>
#if REAL_ORACLE
#include "./ora_err.h"
#endif /* REAL_ORACLE */
```

```
static int     tx_count=0;

/* Tuxedo includes */
#include "atmi.h"
#include "userlog.h"


/* Macros for Struct assignment */

#define MOVETO(element, struct_name) \
                element = struct_name->element
#define MOVEBACK(element, struct_name) \
                struct_name->element = element
#define MOVECTO(element, cnt, struct_name) { \
        int     i; \
        strncpy(element, struct_name->element, cnt);    \
        element[cnt] = '\0';    \
        for(i=0; i<=cnt; i++) \
                if(isspace(element[i])) \
                { \
                        element[i] = '\0'; \
                        break; \
                } \
        }

#define MOVECBACK(element, cnt, struct_name) \
                strncpy(struct_name->element, element, cnt)
/* handle nested structure */

#define MOVESTO(element, struct_name, sub_struct, index) \
                element = struct_name->sub_struct[index].element
#define MOVESBACK(element, struct_name, sub_struct, index) \
                struct_name->sub_struct[index].element = element
#define MOVESCTO(element, cnt, struct_name, sub_struct, index) { \
        int     i; \
        strncpy(element, struct_name->sub_struct[index].element, cnt); \
        element[cnt] = '\0'; \
        for(i=0; i<=cnt; i++) \
                if(isspace(element[i])) \
                { \
                        element[i] = '\0'; \
                        break; \
                } \
        }
#define MOVESCBACK(element, cnt, struct_name, sub_struct, index) \
                strncpy(struct_name->sub_struct[index].element, element, cnt)


/*Lists of items on an order */
/* These structures should match the struct definitions for item_struct and
 no_struct
 * defined in client.h exactly.
 * Any change to those, should be reflected here
 */

struct items_inf {
                int ol_supply_w_id;
                int ol_i_id;
                char i_name[25];
                int             ol_quantity;
                int s_quantity;
                char brand[2];
                double i_price;
                double ol_amount;
} ;

/* List of fields in neworder */
struct newo_inf {
                int w_id;
                int d_id;
                int c_id;
                int o_id;
                int o_ol_cnt;
                double c_discount;
                double w_tax;
                double d_tax;
                char o_entry_d[20];
                char c_credit[3];
                char c_last[17];
                struct items_inf n_items[15];
                char status[25];
                double total;
} ;

struct newo_inf *neworder;              /* Neworder field structure */
char blank_mesg[25] = "                         ";

#if ACID
#include <sys/types.h>
#include <time.h>
time_t curtime, *timep = &curtime;
#endif

int             my_id;

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

                /* struct newo_inf */

                int w_id;
                int d_id;
                int c_id;
                int o_id;
                int o_ol_cnt;
                int c_discount;
                int w_tax;
                int d_tax;
                char o_entry_d[20];
                char c_credit[3];
                char c_last[17];
                char status[25];
                float total;

                int nol_i_id[15];
                int nol_supply_w_id[15];
                int nol_quantity[15];
                int nol_amount[15];
```

```
            char i_name[15][25];
            int s_quantity[15];
            char brand_gen[15][2];
            int i_price[15];
            int o_all_local;
            int retries;

#define NITEMS 15
#define ROWIDLEN 20

struct newctx {
  sb2 nol_i_id_ind[NITEMS];
  sb2 nol_supply_w_id_ind[NITEMS];
  sb2 nol_quantity_ind[NITEMS];
  sb2 nol_amount_ind[NITEMS];
  sb2 i_name_ind[NITEMS];
  sb2 s_quantity_ind[NITEMS];
  sb2 i_price_ind[NITEMS];
  sb2 ol_w_id_ind[NITEMS];
  sb2 ol_d_id_ind[NITEMS];
  sb2 ol_o_id_ind[NITEMS];
  sb2 ol_number_ind[NITEMS];
  sb2 cons_ind[NITEMS];
  sb2 i_id_ind[NITEMS];
  sb2 w_id_ind[NITEMS];
  sb2 s_rowid_ind[NITEMS];
  sb2 s_remote_ind[NITEMS];
  sb2 s_quant_ind[NITEMS];
  sb2 i_data_ind[NITEMS];
  sb2 s_data_ind[NITEMS];
  sb2 s_dist_info_ind[NITEMS];
  sb2 ol_dist_info_ind[NITEMS];

  ub2 nol_i_id_len[NITEMS];
  ub2 nol_supply_w_id_len[NITEMS];
  ub2 nol_quantity_len[NITEMS];
  ub2 nol_amount_len[NITEMS];
  ub2 i_name_len[NITEMS];
  ub2 s_quantity_len[NITEMS];
  ub2 i_price_len[NITEMS];
  ub2 ol_w_id_len[NITEMS];
  ub2 ol_d_id_len[NITEMS];
  ub2 ol_o_id_len[NITEMS];
  ub2 ol_number_len[NITEMS];
  ub2 cons_len[NITEMS];
  ub2 i_id_len[NITEMS];
  ub2 w_id_len[NITEMS];
  ub2 s_rowid_len[NITEMS];
  ub2 s_remote_len[NITEMS];
  ub2 s_quant_len[NITEMS];
  ub2 i_data_len[NITEMS];
  ub2 s_data_len[NITEMS];
  ub2 s_dist_info_len[NITEMS];
  ub2 ol_dist_info_len[NITEMS];

  ub2 nol_i_id_rcode[NITEMS];
  ub2 nol_supply_w_id_rcode[NITEMS];
  ub2 nol_quantity_rcode[NITEMS];
  ub2 nol_amount_rcode[NITEMS];
  ub2 i_name_rcode[NITEMS];
  ub2 s_quantity_rcode[NITEMS];
  ub2 i_price_rcode[NITEMS];
  ub2 ol_w_id_rcode[NITEMS];
  ub2 ol_d_id_rcode[NITEMS];
  ub2 ol_o_id_rcode[NITEMS];
  ub2 ol_number_rcode[NITEMS];
  ub2 cons_rcode[NITEMS];
  ub2 i_id_rcode[NITEMS];
  ub2 w_id_rcode[NITEMS];
  ub2 s_rowid_rcode[NITEMS];
  ub2 s_remote_rcode[NITEMS];
  ub2 s_quant_rcode[NITEMS];
  ub2 i_data_rcode[NITEMS];
  ub2 s_data_rcode[NITEMS];
  ub2 s_dist_info_rcode[NITEMS];
  ub2 ol_dist_info_rcode[NITEMS];

  int ol_w_id[NITEMS];
  int ol_d_id[NITEMS];
  int ol_o_id[NITEMS];
  int ol_number[NITEMS];
  int cons[NITEMS];
  int i_id[NITEMS];
  int w_id[NITEMS];
  char s_rowid[NITEMS][ROWIDLEN];
  int s_remote[NITEMS];
  char i_data[NITEMS][51];
  char s_data[NITEMS][51];
  char s_dist_info[NITEMS][25];
};

unsigned char cr_date[7];
int state;


typedef struct newctx newctx;
newctx *nctx;

#if REAL_ORACLE
ldadef tpclda;
csrdef curn1, curn2, curn3[10], curn4, curs;
unsigned long tpchda[256];

#define SQLTXT "alter session set isolation_level = serializable"

#define SQLTXT1 "BEGIN neworder.enterorder (:w_id, :d_id, :c_id, :o_ol_cnt, \
  :o_all_local, :c_discount, :c_last, :c_credit, :d_tax, :w_tax, :o_id, \
  :o_entry_d, :retry, :cr_date); END;"

#define SQLTXT2 "UPDATE stock SET s_order_cnt = s_order_cnt + 1, \
  s_ytd = s_ytd + :ol_quantity, s_remote_cnt = s_remote_cnt + :s_remote, \
  s_quantity = :s_quantity \
  WHERE rowid = :s_rowid"

#define SQLTXT3 "\
SELECT 0,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :10 AND s_w_id = :30 AND s_i_id = i_id UNION
 ALL \
SELECT 1,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :11 AND s_w_id = :31 AND s_i_id = i_id UNION
```

```
 ALL \
SELECT 2,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :12 AND s_w_id = :32 AND s_i_id = i_id UNION
 ALL \
SELECT 3,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :13 AND s_w_id = :33 AND s_i_id = i_id UNION
 ALL \
SELECT 4,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :14 AND s_w_id = :34 AND s_i_id = i_id UNION
 ALL \
SELECT 5,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :15 AND s_w_id = :35 AND s_i_id = i_id UNION
 ALL \
SELECT 6,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :16 AND s_w_id = :36 AND s_i_id = i_id UNION
 ALL \
SELECT 7,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :17 AND s_w_id = :37 AND s_i_id = i_id UNION
 ALL \
SELECT 8,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :18 AND s_w_id = :38 AND s_i_id = i_id UNION
 ALL \
SELECT 9,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :19 AND s_w_id = :39 AND s_i_id = i_id UNION
 ALL \
SELECT 10,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :20 AND s_w_id = :40 AND s_i_id = i_id UNION
 ALL \
SELECT 11,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :21 AND s_w_id = :41 AND s_i_id = i_id UNION
 ALL \
SELECT 12,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :22 AND s_w_id = :42 AND s_i_id = i_id UNION
 ALL \
SELECT 13,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :23 AND s_w_id = :43 AND s_i_id = i_id UNION
 ALL \
SELECT 14,stock.rowid,i_price,i_name,i_data,s_dist_%02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :24 AND s_w_id = :44 AND s_i_id = i_id"

#define SQLTXT4 "INSERT INTO order_line VALUES (:ol_o_id, :ol_d_id, \
  :ol_w_id, :ol_number, to_date('15-Sep-1911', 'DD-MON-YYYY'), \
  :ol_i_id, :ol_supply_w_id, :ol_quantity, :ol_amount, :ol_dist_info)"

#endif /* REAL_ORACLE */
/*
 * Initialize the neworder transaction
 */
int
init_newo_tx()
{

  int i, j;
  text stmbuf[3000];
  char id[4];
  char sd[4];

  nctx = (newctx *) malloc (sizeof(newctx));

#if REAL_ORACLE
  if (orlon(&tpclda, (ub1 *)tpchda, (text *)"tpcc/tpcc",
     -1, (text *)0, -1, 0)) {
     errrpt(&tpclda, &tpclda, "logon");
     return(1);
  }

  /* turn off auto-commit */
  if (ocof(&tpclda)) {
     errrpt(&tpclda, &tpclda, "auto-commit");
     ologof(&tpclda);
     return(1);
  }
#if VER73
  /* run in serializable mode */

  if (oopen(&curs, &tpclda)) {
                errrpt(&tpclda,&curs,"Open cursor for serializable mode");
                ologof(&tpclda);
                return(1);
                }

   sprintf((char *)stmbuf, SQLTXT);

        if (oparse(&curs, stmbuf, NA, FALSE, VER7)) {
          errrpt(&tpclda,&curs,"Parse for serializable mode");
          oclose(&curs);
          ologof(&tpclda);
          return(1);
        }

        if (oexec(&curs)) {
                errrpt(&tpclda,&curs,"oexec for serializable mode");
                orol(&tpclda);
                oclose(&curs);
                ologof(&tpclda);
                return(1);
                }

        if (oclose(&curs)) {
                errrpt(&tpclda,&curs,"oclose for serializable mode");
                return(1);
                }
#endif
  vgetdate(cr_date);
  /* open first cursor */

  OOPEN(&tpclda,&curn1);

  sprintf ((char *) stmbuf, SQLTXT1);
  OPARSE(&tpclda,&curn1,stmbuf,NA,FALSE,VER7);

  /* bind variables */

  OBNDRV(&tpclda,&curn1,":w_id",ADR(w_id),SIZ(w_id),SQLT_INT);
  OBNDRV(&tpclda,&curn1,":d_id",ADR(d_id),SIZ(d_id),SQLT_INT);
  OBNDRV(&tpclda,&curn1,":c_id",ADR(c_id),SIZ(c_id),SQLT_INT);
  OBNDRV(&tpclda,&curn1,":o_all_local",ADR(o_all_local),SIZ(o_all_local),
          SQLT_INT);
  OBNDRV(&tpclda,&curn1,":o_ol_cnt",ADR(o_ol_cnt),SIZ(o_ol_cnt),SQLT_INT);
```

```
OBNDRV(&tpclda,&curn1,":w_tax",ADR(w_tax),SIZ(w_tax),SQLT_INT);
OBNDRV(&tpclda,&curn1,":d_tax",ADR(d_tax),SIZ(d_tax),SQLT_INT);
OBNDRV(&tpclda,&curn1,":o_id",ADR(o_id),SIZ(o_id),SQLT_INT);
OBNDRV(&tpclda,&curn1,":c_discount",ADR(c_discount),SIZ(c_discount),
        SQLT_INT);
OBNDRV(&tpclda,&curn1,":c_credit",c_credit,SIZ(c_credit),SQLT_STR);
OBNDRV(&tpclda,&curn1,":c_last",c_last,SIZ(c_last),SQLT_STR);
OBNDRV(&tpclda,&curn1,":o_entry_d",o_entry_d,SIZ(o_entry_d),SQLT_STR);
OBNDRV(&tpclda,&curn1,":retry",ADR(retries),SIZ(retries),SQLT_INT);
OBNDRV(&tpclda,&curn1,":cr_date",cr_date,SIZ(cr_date),SQLT_DAT);

/* open second cursor */

OOPEN(&tpclda,&curn2);

sprintf ((char *) stmbuf, SQLTXT2);
OPARSE(&tpclda,&curn2,stmbuf,NA,FALSE,VER7);

/* bind variables */

OBNDRA(&tpclda,&curn2,":s_quantity",s_quantity,SIZ(int),SQLT_INT,
        nctx->s_quant_ind,nctx->s_quant_len, nctx->s_quant_rcode);
OBNDRA(&tpclda,&curn2,":s_rowid",nctx->s_rowid,SIZ(nctx->s_rowid[0]),
                SQLT_RID,
nctx->s_rowid_ind,nctx->s_rowid_len,nctx->s_rowid_rcode);
OBNDRA(&tpclda,&curn2,":ol_quantity",nol_quantity,SIZ(int),SQLT_INT,
        nctx->nol_quantity_ind,nctx->nol_quantity_len,
    nctx->nol_quantity_rcode);
OBNDRA(&tpclda,&curn2,":s_remote",nctx->s_remote,SIZ(int),SQLT_INT,
        nctx->s_remote_ind,nctx->s_remote_len,
    nctx->s_remote_rcode);




/* open third cursor and bind variables */

for (i = 0; i < 10; i++) {
  OOPEN(&tpclda,&curn3[i]);
  j = i + 1;
  sprintf ((char *) stmbuf, SQLTXT3, j, j, j, j, j, j, j, j, j, j,
       j, j, j);
  OPARSE(&tpclda,&curn3[i],stmbuf,NA,FALSE,VER7);
  for (j = 0; j < NITEMS; j++) {
    sprintf (id, ":%d", j + 10);
    sprintf (sd, ":%d", j + 30);
    OBNDRA(&tpclda,&curn3[i],id,ADR(nol_i_id[j]),SIZ(int),SQLT_INT,
            &nctx->nol_i_id_ind[j],&nctx->nol_i_id_len[j],
        &nctx->nol_i_id_rcode[j]);
    OBNDRA(&tpclda,&curn3[i],sd,ADR(nol_supply_w_id[j]),SIZ(int),SQLT_INT,
            &nctx->nol_supply_w_id_ind[j],&nctx->nol_supply_w_id_len[j],
            &nctx->nol_supply_w_id_rcode[j]);
    nctx->nol_i_id_ind[j] = NA;
    nctx->nol_supply_w_id_ind[j] = NA;
    nctx->nol_i_id_len[j] = NULL;
    nctx->nol_supply_w_id_len[j] = NULL;
  }

  ODEFIN(&tpclda,&curn3[i],1,nctx->cons,SIZ(nctx->cons[0]),SQLT_INT,NA,
      nctx->cons_ind,NULL,NA,NA,nctx->cons_len, nctx->cons_rcode);
ODEFIN(&tpclda,&curn3[i],2,nctx->s_rowid,SIZ(nctx->s_rowid[0]),SQLT_RID,NA,
        nctx->s_rowid_ind,NULL,NA,NA,nctx->s_rowid_len,
 nctx->s_rowid_rcode);
    ODEFIN(&tpclda,&curn3[i],3,i_price,SIZ(float),SQLT_INT,NA,
        nctx->i_price_ind,NULL,NA,NA,nctx->i_price_len,
        nctx->i_price_rcode);
    ODEFIN(&tpclda,&curn3[i],4,i_name,SIZ(i_name[0]),SQLT_STR,NA,
        nctx->i_name_ind,NULL,NA,NA,nctx->i_name_len,nctx->i_name_rcode);
    ODEFIN(&tpclda,&curn3[i],5,nctx->i_data,SIZ(nctx->i_data[0]),SQLT_STR,NA,
        nctx->i_data_ind,NULL,NA,NA,nctx->i_data_len, nctx->i_data_rcode);
    ODEFIN(&tpclda,&curn3[i],6,nctx->s_dist_info,SIZ(nctx->s_dist_info[0]),
        SQLT_STR,NA,nctx->s_dist_info_ind,NULL,NA,NA,
        nctx->s_dist_info_len, nctx->s_dist_info_rcode);

ODEFIN(&tpclda,&curn3[i],7,nctx->s_data,SIZ(nctx->s_data[0]),SQLT_STR,NA,
        nctx->s_data_ind,NULL,NA,NA,nctx->s_data_len, nctx->s_data_rcode);

    ODEFIN(&tpclda,&curn3[i],8,s_quantity,SIZ(int),SQLT_INT,NA,
        nctx->s_quantity_ind,NULL,NA,NA,nctx->s_quantity_len,
        nctx->s_quantity_rcode);
  }

  /* open fourth cursor */

  OOPEN(&tpclda,&curn4);

  sprintf ((char *) stmbuf, SQLTXT4);
  OPARSE(&tpclda,&curn4,stmbuf,NA,FALSE,VER7);

  /* bind variables */

  OBNDRA(&tpclda,&curn4,":ol_o_id",nctx->ol_o_id,SIZ(int),SQLT_INT,
          nctx->ol_o_id_ind,nctx->ol_o_id_len,nctx->ol_o_id_rcode);
  OBNDRA(&tpclda,&curn4,":ol_d_id",nctx->ol_d_id,SIZ(int),SQLT_INT,
          nctx->ol_d_id_ind,nctx->ol_d_id_len,nctx->ol_d_id_rcode);
  OBNDRA(&tpclda,&curn4,":ol_w_id",nctx->ol_w_id,SIZ(int),SQLT_INT,
          nctx->ol_w_id_ind,nctx->ol_w_id_len,nctx->ol_w_id_rcode);
  OBNDRA(&tpclda,&curn4,":ol_number",nctx->ol_number,SIZ(int),SQLT_INT,
          nctx->ol_number_ind,nctx->ol_number_len,nctx->ol_number_rcode);
  OBNDRA(&tpclda,&curn4,":ol_i_id",nol_i_id,SIZ(int),SQLT_INT,
          nctx->nol_i_id_ind,nctx->nol_i_id_len,nctx->nol_i_id_rcode);
OBNDRA(&tpclda,&curn4,":ol_supply_w_id",nol_supply_w_id,SIZ(int),SQLT_INT,
          nctx->nol_supply_w_id_ind,nctx->nol_supply_w_id_len,
          nctx->nol_supply_w_id_rcode);
  OBNDRA(&tpclda,&curn4,":ol_quantity",nol_quantity,SIZ(int),SQLT_INT,
          nctx->nol_quantity_ind,nctx->nol_quantity_len,
      nctx->nol_quantity_rcode);
  OBNDRA(&tpclda,&curn4,":ol_amount",nol_amount,SIZ(float),SQLT_INT,
nctx->nol_amount_ind,nctx->nol_amount_len,nctx->nol_amount_rcode);
  OBNDRA(&tpclda,&curn4,":ol_dist_info",nctx->s_dist_info,
      SIZ(nctx->s_dist_info[0]),SQLT_STR,nctx->ol_dist_info_ind,
      nctx->ol_dist_info_len, nctx->ol_dist_info_rcode);


#ifdef TKPROF
            EXEC SQL ALTER SESSION SET SQL_TRACE = TRUE;
#endif

#endif /* REAL_ORACLE */
```

```
        return(0);
}

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/


/*
 * This function executes the neworder transaction
 */
neworder_tx(rqst)
TPSVCINFO *rqst;
{

        struct newo_inf         *neworder_p = (struct newo_inf *)(rqst->data);

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

        int i, j, k;
        int rpc, rpc3, rowoff, iters, onepass;

#if ACID
        int reread;
        time(timep);
        userlog("ACID NEWORDER started at %s\n", ctime(timep));
#endif

  vgetdate(cr_date);
        MOVETO(w_id, neworder_p);
   MOVETO(d_id, neworder_p);
   MOVETO(c_id , neworder_p);
   MOVETO(o_ol_cnt, neworder_p);
        tx_count++;

        strcpy(neworder_p->status, blank_mesg);

retry:
#if ISO7
        reread = 1;
#endif
        state = 0;
        onepass = 1;

        o_all_local = 1;
        for (i = 0; i < o_ol_cnt; i++) {
                nol_supply_w_id[i] =
 neworder_p->n_items[i].ol_supply_w_id;
                if (nol_supply_w_id[i] == w_id) {
                        nctx->s_remote[i] = 0;
                }
                else {
                        nctx->s_remote[i] = 1;
                        o_all_local = 0;
                }
                nol_i_id[i] = neworder_p->n_items[i].ol_i_id;
                nol_quantity[i] = neworder_p->n_items[i].ol_quantity;
        }
        for (; i < 15; i++) {
                nol_supply_w_id[i] = 0;
                nol_i_id[i] = 0;
        }


  /* execute stored procedure */

#if     REAL_ORACLE
  if (oexec (&curn1)) {
    if (curn1.rc == NOT_SERIALIZABLE) {
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else if (errrpt (&tpclda, &curn1) == RECOVERR) {
      orol (&tpclda);
      retries++;
      goto retry;
    }
    else {
      orol (&tpclda);
      return (-1);
    }
  }
#endif  /*REAL_ORACLE*/

 #ifdef ISO7
 iso7:
 #endif

  /* initialization for array operations */

  for (i = 0; i < o_ol_cnt; i++) {
    nctx->ol_w_id[i] = w_id;
    nctx->ol_d_id[i] = d_id;
    nctx->ol_number[i] = i + 1;

    nctx->nol_i_id_ind[i] = TRUE;
    nctx->nol_supply_w_id_ind[i] = TRUE;
    nctx->nol_quantity_ind[i] = TRUE;
    nctx->nol_amount_ind[i] = TRUE;
    nctx->ol_w_id_ind[i] = TRUE;
    nctx->ol_d_id_ind[i] = TRUE;
    nctx->ol_o_id_ind[i] = TRUE;
    nctx->ol_number_ind[i] = TRUE;
    nctx->ol_dist_info_ind[i] = TRUE;
    nctx->s_remote_ind[i] = TRUE;
    nctx->s_quant_ind[i] = TRUE;
    nctx->s_rowid_ind[i] = TRUE;

    nctx->nol_i_id_len[i] = sizeof(int);
    nctx->nol_supply_w_id_len[i] = sizeof(int);
    nctx->nol_quantity_len[i] = sizeof(int);
    nctx->nol_amount_len[i] = sizeof(int);
    nctx->ol_w_id_len[i] = sizeof(int);
    nctx->ol_d_id_len[i] = sizeof(int);
    nctx->ol_o_id_len[i] = sizeof(int);
    nctx->ol_number_len[i] = sizeof(int);
    nctx->ol_dist_info_len[i] = sizeof(nctx->s_dist_info[0]);
```

```
    nctx->s_remote_len[i] = sizeof(int);
    nctx->s_quant_len[i] = sizeof(int);
    nctx->s_rowid_len[i] = sizeof(nctx->s_rowid[0]);
  }
  for (i = o_ol_cnt; i < NITEMS; i++) {
    nctx->nol_i_id_ind[i] = NA;
    nctx->nol_supply_w_id_ind[i] = NA;
    nctx->nol_quantity_ind[i] = NA;
    nctx->nol_amount_ind[i] = NA;
    nctx->ol_w_id_ind[i] = NA;
    nctx->ol_d_id_ind[i] = NA;
    nctx->ol_o_id_ind[i] = NA;
    nctx->ol_number_ind[i] = NA;
    nctx->ol_dist_info_ind[i] = NA;
    nctx->s_remote_ind[i] = NA;
    nctx->s_quant_ind[i] = NA;
    nctx->s_rowid_ind[i] = NA;


    nctx->nol_i_id_len[i] = 0;
    nctx->nol_supply_w_id_len[i] = 0;
    nctx->nol_quantity_len[i] = 0;
    nctx->nol_amount_len[i] = 0;
    nctx->ol_w_id_len[i] = 0;
    nctx->ol_d_id_len[i] = 0;
    nctx->ol_o_id_len[i] = 0;
    nctx->ol_number_len[i] = 0;
    nctx->ol_dist_info_len[i] = 0;
    nctx->s_remote_len[i] = 0;
    nctx->s_quant_len[i] = 0;
    nctx->s_rowid_len[i] = 0;
  }


                rpc3 = SelItemStk (&onepass);
                if (rpc3 == -2)
                        goto retry;
                else if (rpc3 == -1)
                        return(-1);

#ifdef ISO7
                time(timep);
                userlog ("    Item table read at: %s\n", ctime(timep));
                for (i = 0; i < o_ol_cnt; i++) {
                        if (nctx->nol_i_id_ind[i] != NA)
                                userlog ("    i_id = %d, i_price = %.2f\n",
nol_i_id[i], i_price[i]);
                }
                if (reread) {
                        sleep (30);
                        reread = 0;
                        goto iso7;
                }
#endif


  /* compute order line amounts, total amount and stock quantities */

  total = 0.0;
  for (i = 0; i < o_ol_cnt; i++) {
    nctx->ol_o_id[i] = o_id;
    if (nctx->nol_i_id_ind[i] != NA) {
      s_quantity[i] -= nol_quantity[i];
      if (s_quantity[i] < 10)
        s_quantity[i] += 91;
      nol_amount[i] = (nol_quantity[i] * i_price[i]);
      total += nol_amount[i];
      if (strstr (nctx->i_data[i], "ORIGINAL") &&
          strstr (nctx->s_data[i], "ORIGINAL"))
        brand_gen[i][0] = 'B';
      else
        brand_gen[i][0] = 'G';
                        brand_gen[i][1] = '\0';
    }
  }
  total *= ((float)(10000  - c_discount)/10000) * (1.0 + ((float)(d_tax/10000)) +
 (float)(w_tax/10000));


#if             REAL_ORACLE
                rpc = UpdStk2(onepass);
                if (rpc == -2)
                        goto retry;
                else if (rpc == -1)
                        return(-1);


                /* number of items selected != number of stock updated */
                if (rpc3 != rpc) {
#if TPCSO
                        write_log("Newo %d: %d rows of item read, %d stock
 updated\n", my_id, rpc3, rpc);
#else
                        userlog("Newo rpc3 != rpc");
#endif
                        orol(&tpclda);
                        return(-1);
                }

  /* array insert into order line table */

  if (onepass && ((o_ol_cnt - state) > 0)) {
    if (oexn (&curn4, o_ol_cnt - state, 0)) {
      if (curn4.rc == NOT_SERIALIZABLE) {
        orol (&tpclda);
        retries++;
        goto retry;
      }
      else if (errrpt (&tpclda, &curn4) == RECOVERR) {
        orol (&tpclda);
        retries++;
        goto retry;
      }
      else {
        orol (&tpclda);
        return (-1);
      }
    }
    if (curn4.rpc != (o_ol_cnt - state)) {
#if TPCSO
      write_log ("Newo %d: array insert failed\n", my_id);
```

```
#else
        userlog ("array insert failed\n");
#endif
        orol (&tpclda);
        return (-1);
      }
    }

    /* continue array insert into order line until whole array is processed */

    else if ((o_ol_cnt - state) > 0) {
#if TPCSO
      write_log ("Newo %d: more than 1 pass of OEXN!\n", my_id);
#else
      userlog ("more than 1 pass of OEXN!\n");
#endif
      rpc = 0;
      for (rowoff = 0; rowoff < o_ol_cnt; rowoff++)
        if (nctx->nol_i_id_ind[rowoff] != NA)
          break;
      for (iters = rowoff + 1; iters < o_ol_cnt; iters++)
        if (nctx->nol_i_id_ind[iters] == NA)
          break;
      while ((rpc < (o_ol_cnt - state)) && (iters <= o_ol_cnt)) {
        if (oexn (&curn4, iters, rowoff)) {
          if (curn4.rc == NOT_SERIALIZABLE) {
            orol (&tpclda);
            retries++;
            goto retry;
          }
          else if (errrpt (&tpclda, &curn4) == RECOVERR) {
            orol (&tpclda);
            retries++;
            goto retry;
          }
          else {
            orol (&tpclda);
            return (-1);
          }
        }
        if (curn4.rpc != (iters - rowoff)) {
#if TPCSO
          write_log ("Newo %d: array insert failed\n", my_id);
#else
          userlog ("array insert failed\n");
#endif
          orol (&tpclda);
          return (-1);
        }
        rpc += curn4.rpc;
        for (rowoff = iters + 1; rowoff < o_ol_cnt; rowoff++)
          if (nctx->nol_i_id_ind[rowoff] != NA)
            break;
        for (iters = rowoff + 1; iters < o_ol_cnt; iters++)
          if (nctx->nol_i_id_ind[iters] == NA)
            break;
      }
    }
#endif        /*REAL_ORACLE*/
    for (i = 0; i < o_ol_cnt; i++) {
                neworder_p->n_items[i].s_quantity = s_quantity[i];
                neworder_p->n_items[i].i_price = ((double)i_price[i])/100;
                neworder_p->n_items[i].ol_amount = ((double)nol_amount[i])/100;
                strcpy(neworder_p->n_items[i].i_name, i_name[i]);
                strcpy(neworder_p->n_items[i].brand, brand_gen[i]);
                }

#if ISO1234
                sleep(30);
#endif

    /* commit if no invalid item */

#if             REAL_ORACLE
    if (state) {
                                strcpy(neworder_p->status, "Item number is not valid");
      orol (&tpclda);
    }
    else {
      OCOM(&tpclda,&tpclda);
    }
#endif          /*REAL_ORACLE*/

                MOVEBACK(o_id , neworder_p);
                neworder_p->c_discount = ((double)c_discount) / 10000;
                neworder_p->w_tax = ((double)w_tax) / 10000;
                neworder_p->d_tax = ((double)d_tax) / 10000;
                MOVECBACK(o_entry_d, 20, neworder_p);
                MOVECBACK(c_credit, 2, neworder_p);
                MOVECBACK(c_last, 16, neworder_p);
                MOVEBACK(total, neworder_p);

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/

#if ACID
                time(timep);
#if !ISO1234
userlog("ACID NEWORDER w_id=%d, d_id=%d, c_id=%d, o_id=%d,
total=%f\n",
                                w_id, d_id, c_id, o_id, total/100.0);
#endif
                userlog("ACID NEWORDER completed at %s\n", ctime(timep));
#endif
                return(0);
}



swapitemstock (i, j)
int i, j;
{

  int k, tempi;
  int tempf;
  char tempstr[52];
  ub2 tempub2;
  sb2 tempsb2;

  tempsb2 = nctx->cons_ind[i];
```

```
nctx->cons_ind[i] = nctx->cons_ind[j];
nctx->cons_ind[j] = tempsb2;
tempub2 = nctx->cons_len[i];
nctx->cons_len[i] = nctx->cons_len[j];
nctx->cons_len[j] = tempub2;
tempub2 = nctx->cons_rcode[i];
nctx->cons_rcode[i] = nctx->cons_rcode[j];
nctx->cons_rcode[j] = tempub2;
tempi = nctx->cons[i];
nctx->cons[i] = nctx->cons[j];
nctx->cons[j] = tempi;

tempsb2 = nctx->s_rowid_ind[i];
nctx->s_rowid_ind[i] = nctx->s_rowid_ind[j];
nctx->s_rowid_ind[j] = tempsb2;
tempub2 = nctx->s_rowid_len[i];
nctx->s_rowid_len[i] = nctx->s_rowid_len[j];
nctx->s_rowid_len[j] = tempub2;
tempub2 = nctx->s_rowid_rcode[i];
nctx->s_rowid_rcode[i] = nctx->s_rowid_rcode[j];
nctx->s_rowid_rcode[j] = tempub2;
for (k = 0; k < ROWIDLEN; k++) {
        tempstr[0] = nctx->s_rowid[i][k];
        nctx->s_rowid[i][k] = nctx->s_rowid[j][k];
        nctx->s_rowid[j][k] = tempstr[0];
        }

tempsb2 = nctx->i_price_ind[i];
nctx->i_price_ind[i] = nctx->i_price_ind[j];
nctx->i_price_ind[j] = tempsb2;
tempub2 = nctx->i_price_len[i];
nctx->i_price_len[i] = nctx->i_price_len[j];
nctx->i_price_len[j] = tempub2;
tempub2 = nctx->i_price_rcode[i];
nctx->i_price_rcode[i] = nctx->i_price_rcode[j];
nctx->i_price_rcode[j] = tempub2;
tempf = i_price[i];
i_price[i] = i_price[j];
i_price[j] = tempf;

tempsb2 = nctx->i_name_ind[i];
nctx->i_name_ind[i] = nctx->i_name_ind[j];
nctx->i_name_ind[j] = tempsb2;
tempub2 = nctx->i_name_len[i];
nctx->i_name_len[i] = nctx->i_name_len[j];
nctx->i_name_len[j] = tempub2;
tempub2 = nctx->i_name_rcode[i];
nctx->i_name_rcode[i] = nctx->i_name_rcode[j];
nctx->i_name_rcode[j] = tempub2;
strncpy (tempstr, i_name[i], 25);
strncpy (i_name[i], i_name[j], 25);
strncpy (i_name[j], tempstr, 25);

tempsb2 = nctx->i_data_ind[i];
nctx->i_data_ind[i] = nctx->i_data_ind[j];
nctx->i_data_ind[j] = tempsb2;
tempub2 = nctx->i_data_len[i];
nctx->i_data_len[i] = nctx->i_data_len[j];
nctx->i_data_len[j] = tempub2;
tempub2 = nctx->i_data_rcode[i];
nctx->i_data_rcode[i] = nctx->i_data_rcode[j];
nctx->i_data_rcode[j] = tempub2;
strncpy (tempstr, nctx->i_data[i], 51);
strncpy (nctx->i_data[i], nctx->i_data[j], 51);
strncpy (nctx->i_data[j], tempstr, 51);

tempsb2 = nctx->s_quantity_ind[i];
nctx->s_quantity_ind[i] = nctx->s_quantity_ind[j];
nctx->s_quantity_ind[j] = tempsb2;
tempub2 = nctx->s_quantity_len[i];
nctx->s_quantity_len[i] = nctx->s_quantity_len[j];
nctx->s_quantity_len[j] = tempub2;
tempub2 = nctx->s_quantity_rcode[i];
nctx->s_quantity_rcode[i] = nctx->s_quantity_rcode[j];
nctx->s_quantity_rcode[j] = tempub2;
tempi = s_quantity[i];
s_quantity[i] = s_quantity[j];
s_quantity[j] = tempi;

tempsb2 = nctx->s_dist_info_ind[i];
nctx->s_dist_info_ind[i] = nctx->s_dist_info_ind[j];
nctx->s_dist_info_ind[j] = tempsb2;
tempub2 = nctx->s_dist_info_len[i];
nctx->s_dist_info_len[i] = nctx->s_dist_info_len[j];
nctx->s_dist_info_len[j] = tempub2;
tempub2 = nctx->s_dist_info_rcode[i];
nctx->s_dist_info_rcode[i] = nctx->s_dist_info_rcode[j];
nctx->s_dist_info_rcode[j] = tempub2;
strncpy (tempstr, nctx->s_dist_info[i], 25);
strncpy (nctx->s_dist_info[i], nctx->s_dist_info[j], 25);
strncpy (nctx->s_dist_info[j], tempstr, 25);

tempsb2 = nctx->s_data_ind[i];
nctx->s_data_ind[i] = nctx->s_data_ind[j];
nctx->s_data_ind[j] = tempsb2;
tempub2 = nctx->s_data_len[i];
nctx->s_data_len[i] = nctx->s_data_len[j];
nctx->s_data_len[j] = tempub2;
tempub2 = nctx->s_data_rcode[i];
nctx->s_data_rcode[i] = nctx->s_data_rcode[j];
nctx->s_data_rcode[j] = tempub2;
strncpy (tempstr, nctx->s_data[i], 51);
strncpy (nctx->s_data[i], nctx->s_data[j], 51);
strncpy (nctx->s_data[j], tempstr, 51);

}



SelItemStk (onepass)
int *onepass;
{
        int i, j, rpc3;

  /* array select from item and stock tables */

  if (oexfet (&curn3[d_id-1], o_ol_cnt, 0, 0)) {
    if (curn3[d_id-1].rc == NOT_SERIALIZABLE) {
      orol (&tpclda);
      retries++;
```

```
      return (-2);
    }
    else if (curn3[d_id-1].rc != NO_DATA_FOUND) {
      if (errrpt (&tpclda, &curn3[d_id-1]) == RECOVERR) {
        orol (&tpclda);
        retries++;
        return (-2);
      }
      else {
        orol (&tpclda);
        return (-1);
      }
    }
  }

  /* mark invalid items */

  rpc3 = curn3[d_id-1].rpc;
  if (curn3[d_id-1].rpc != o_ol_cnt)
    for (i = curn3[d_id-1].rpc; i < o_ol_cnt; i++) {
      nctx->cons_ind[i] = NA;
      nctx->s_rowid_ind[i] = NA;
    }

  /* check for invalid items and reorder results if necessary */

  for (i = 0; i < o_ol_cnt; i++) {
    if (nctx->cons_ind[i] != NA) {
      if (nctx->cons[i] != i) {

        /* this item is invalid or results are out of order */

#ifdef TPCSO
        write_log ("Newo %d: reordering items and stocks\n", my_id);
#else
        userlog ("reordering items and stocks\n");
#endif

        for (j = i + 1; j < o_ol_cnt; j++) {

          /* this item is valid, but results are out of order */

          if ((nctx->cons_ind[j] != NA) && (nctx->cons[j] == i)) {
            swapitemstock (i, j);
            break;
          }
        }

        /* this item (not the last one) is invalid */

        if (j >= o_ol_cnt) {
          state++;
          nctx->nol_i_id_ind[i] = NA;
          nctx->nol_supply_w_id_ind[i] = NA;
          nctx->nol_quantity_ind[i] = NA;
          nctx->nol_amount_ind[i] = NA;
          nctx->ol_w_id_ind[i] = NA;
          nctx->ol_d_id_ind[i] = NA;
          nctx->ol_o_id_ind[i] = NA;
          nctx->ol_number_ind[i] = NA;
          nctx->ol_dist_info_ind[i] = NA;
          nctx->s_remote_ind[i] = NA;
          nctx->s_quant_ind[i] = NA;

          nctx->nol_i_id_len[i] = 0;
          nctx->nol_supply_w_id_len[i] = 0;
          nctx->nol_quantity_len[i] = 0;
          nctx->nol_amount_len[i] = 0;
          nctx->ol_w_id_len[i] = 0;
          nctx->ol_d_id_len[i] = 0;
          nctx->ol_o_id_len[i] = 0;
          nctx->ol_number_len[i] = 0;
          nctx->ol_dist_info_len[i] = 0;
          nctx->s_remote_len[i] = 0;
          nctx->s_quant_len[i] = 0;

          *onepass = 0;
          for (j = i + 1; j < o_ol_cnt; j++) {
            if (nctx->cons_ind[j] == NA) {
              swapitemstock (i, j);
              break;
            }
          }
        }
      }
    }
    else {                      /* this item is invalid */
      state++;
      nctx->nol_i_id_ind[i] = NA;
      nctx->nol_supply_w_id_ind[i] = NA;
      nctx->nol_quantity_ind[i] = NA;
      nctx->nol_amount_ind[i] = NA;
      nctx->ol_w_id_ind[i] = NA;
      nctx->ol_d_id_ind[i] = NA;
      nctx->ol_o_id_ind[i] = NA;
      nctx->ol_number_ind[i] = NA;
      nctx->ol_dist_info_ind[i] = NA;
      nctx->s_remote_ind[i] = NA;
      nctx->s_quant_ind[i] = NA;

      nctx->nol_i_id_len[i] = 0;
      nctx->nol_supply_w_id_len[i] = 0;
      nctx->nol_quantity_len[i] = 0;
      nctx->nol_amount_len[i] = 0;
      nctx->ol_w_id_len[i] = 0;
      nctx->ol_d_id_len[i] = 0;
      nctx->ol_o_id_len[i] = 0;
      nctx->ol_number_len[i] = 0;
      nctx->ol_dist_info_len[i] = 0;
      nctx->s_remote_len[i] = 0;
      nctx->s_quant_len[i] = 0;
    }
  }

  /* more than 1 invalid item!!!  shouldn't happen in TPC-C */

  if (state > 1) {
#ifdef TPCSO
    write_log ("Newo %d: more than 1 invalid item?\n", my_id);
#else
```

```
    userlog ("more than 1 invalid item?\n");
#endif
  }
  return (rpc3);
}


UpdStk2 (onepass)
int onepass;
{
        int rpc, rowoff, iters;

  /* array update of stock table */
        rpc = 0;
  if (onepass && ((o_ol_cnt - state) > 0)) {
  if (oexn (&curn2, o_ol_cnt - state, 0)) {
    if (curn2.rc == NOT_SERIALIZABLE) {
      orol (&tpclda);
      retries++;
      return(-2);
    }
    else if (errrpt (&tpclda, &curn2, "upd_stock") == RECOVERR) {
      orol (&tpclda);
      retries++;
      return(-2);
    }
    else {
      orol (&tpclda);
      return (-1);
    }
  }
  rpc = curn2.rpc;
  }

  /* continue to do array update of stock until whole array is processed */

  else if ((o_ol_cnt - state) > 0) {
#if TPCSO
    write_log ("Newo %d: more than 1 pass of OEXN!\n", my_id);
#else
    userlog ("more than 1 pass of OEXN!\n");
#endif
    rpc = 0;
    for (rowoff = 0; rowoff < o_ol_cnt; rowoff++)
                if (nctx->s_rowid_ind[rowoff] != NA)
                        break;
                for (iters = rowoff + 1; iters < o_ol_cnt; iters++)
                        if (nctx->s_rowid_ind[iters] == NA)
                                break;
                while ((rpc < (o_ol_cnt - state)) && (iters <= o_ol_cnt)) {
      if (oexn (&curn2, iters, rowoff)) {
        if (curn2.rc == NOT_SERIALIZABLE) {
          orol (&tpclda);
          retries++;
          return(-2);
        }
        else if (errrpt (&tpclda, &curn2, "upd_stock") == RECOVERR) {
          orol (&tpclda);
          retries++;
          return(-2);
        }
        else {
          orol (&tpclda);
          return (-1);
        }
      }
      if (curn2.rpc != (iters - rowoff)) {
                        userlog("Newo  array update failed\n");
                        orol (&tpclda);
                        return(-1);
                }
                rpc += curn2.rpc;
                for (rowoff = iters + 1; rowoff < o_ol_cnt; rowoff++)
                        if (nctx->s_rowid_ind[rowoff] != NA)
                                break;
                for (iters = rowoff + 1; iters < o_ol_cnt; iters++)
                        if (nctx->s_rowid_ind[iters] == NA)
                                break;
    }
  }

   /* number of items selected != number of stock updated */

  if (rpc != (o_ol_cnt - state)) {
#if TPCSO
    write_log ("Newo %d: array update failed\n", my_id);
#else
    userlog ("array update failed\n");
#endif
    orol (&tpclda);
    return (-1);
  }
  return(rpc);
}


/* Start of Tuxedo code */
int
tpsvrinit(argc, argv)
char **argv;
{
        return(init_newo_tx());         /* Prepare transaction */
}

void
tpsvrdone()
{
        int i;

  oclose (&curn1);
  oclose (&curn2);
  for ( i = 0; i < 10; i++)
        oclose(&curn3[i]);
  oclose (&curn4);
  /* log off */
  ologof (&tpclda);

}

NEWO(rqst)
```

```
TPSVCINFO *rqst;
{
        if (neworder_tx(rqst))
                tpreturn(TPFAIL, 0, rqst->data, sizeof(struct newo_inf), 0);
        else
                tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct
newo_inf), 0);
}



vgetdate(unsigned char *buf)
{
  time_t tloc;
  char temp[5];
  int cen;

  if(time(&tloc) == (time_t)-1) {
    printf("Error getting date\n");
    exit(1);
  }
  cftime(temp,"%d",&tloc);
  buf[3] = (unsigned char)atoi(temp);
  cftime(temp,"%m",&tloc);
  buf[2] = (unsigned char)atoi(temp);
  cftime(temp,"%Y",&tloc);
  cen = atoi(temp);
  buf[0] = (unsigned char)((cen/100)+100);
  buf[1] = (unsigned char)((cen%100)+100);
  cftime(temp,"%H",&tloc);
  buf[4] = (unsigned char)(atoi(temp) + 1);
  cftime(temp,"%M",&tloc);
  buf[5] = (unsigned char)(atoi(temp) + 1);
  cftime(temp,"%S",&tloc);
  buf[6] = (unsigned char)(atoi(temp)+ 1);
}
```

tpcc_srv_ords.c

```
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcc_srv_ords.c              1.17            96/01/11     SMI"

/*
 * File: ords.ec
 * Order-status transaction code for Informix
 * Author : Shanti S
 * Date   : 7/06/93
 *
 */

#include "oci.h"

#include <stdio.h>

#include "./ora_err.h"

static int      tx_count=0;

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"

#define REAL_ORACLE 1     /* Compile with connections to RDMS */

/* Macros for structure operations */

#define MOVETO(element, struct_name) \
                element = struct_name->element
#define MOVEBACK(element, struct_name) \
                struct_name->element = element
#define MOVECTO(element, cnt, struct_name) { \
        int     i; \
        strncpy(element, struct_name->element, cnt);    \
        element[cnt] = '\0';    \
        for(i=0; i<=cnt; i++) \
                if(isspace(element[i])) \
                { \
                        element[i] = '\0'; \
                        break; \
                } \
        }
#define MOVECBACK(element, cnt, struct_name) \
                strncpy(struct_name->element, element, cnt)

/* handle nested structure */

#define MOVESTO(element, struct_name, sub_struct, index) \
                element = struct_name->sub_struct[index].element
#define MOVESBACK(element, struct_name, sub_struct, index) \
                struct_name->sub_struct[index].element = element
#define MOVESCTO(element, cnt, struct_name, sub_struct, index) { \
        int     i; \
        strncpy(element, struct_name->sub_struct[index].element, cnt); \
        element[cnt] = '\0'; \
        for(i=0; i<=cnt; i++) \
                if(isspace(element[i])) \
                { \
                        element[i] = '\0'; \
                        break; \
                } \
        }
#define MOVESCBACK(element, cnt, struct_name, sub_struct, index) \
                strncpy(struct_name->sub_struct[index].element, element, cnt)

/* List of fields in ordstat */
/* This structure should be EXACTLY identical to the one declared in client.h */
/* Lists of items on an order */
struct ord_itm_inf {
                int ol_supply_w_id;
                int ol_i_id;
                int             ol_quantity;
                double ol_amount;
                char ol_delivery_d[11];
} ;

struct ord_inf {
                int item_cnt;
                int w_id;
```

```
        int d_id;
        int c_id;
        int o_id;
        int o_carrier_id;
        double c_balance;
        char c_first[17];
        char c_middle[3];
        char c_last[17];
        char o_entry_d[20];
        struct ord_itm_inf o_items[15];
};

/*$struct ord_inf *ordstat; */                  /* Input structure to ordstat_tx */

#if ACID
#include <sys/types.h>
#include <time.h>
time_t curtime;
time_t *timep = &curtime;
#endif


/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

ldadef tpclda;
csrdef curs, curo0, curo1, curo2;
unsigned long tpchda[256];

#define NITEMS 15
struct ordctx {
  sb2 ol_supply_w_id_ind[NITEMS];
  sb2 ol_i_id_ind[NITEMS];
  sb2 ol_quantity_ind[NITEMS];
  sb2 ol_amount_ind[NITEMS];
  sb2 ol_delivery_d_ind[NITEMS];
  sb2 ol_w_id_ind;
  sb2 ol_d_id_ind;
  sb2 ol_o_id_ind;

  ub2 ol_supply_w_id_len[NITEMS];
  ub2 ol_i_id_len[NITEMS];
  ub2 ol_quantity_len[NITEMS];
  ub2 ol_amount_len[NITEMS];
  ub2 ol_delivery_d_len[NITEMS];
  sb2 ol_w_id_len;
  sb2 ol_d_id_len;
  sb2 ol_o_id_len;

  ub2 ol_supply_w_id_rcode[NITEMS];
  ub2 ol_i_id_rcode[NITEMS];
  ub2 ol_quantity_rcode[NITEMS];
  ub2 ol_amount_rcode[NITEMS];
  ub2 ol_delivery_d_rcode[NITEMS];
  sb2 ol_w_id_rcode;
  sb2 ol_d_id_rcode;
  sb2 ol_o_id_rcode;

  ub4 ol_supply_w_id_csize;
  ub4 ol_i_id_csize;
  ub4 ol_quantity_csize;
  ub4 ol_amount_csize;
  ub4 ol_delivery_d_csize;
  sb2 ol_w_id_csize;
  sb2 ol_d_id_csize;
  sb2 ol_o_id_csize;

};
typedef struct ordctx ordctx;

ordctx *octx;


/* struct ord_inf elements */
int w_id;
int d_id;
int c_id, bylastname;
int o_id;
int o_carrier_id;
int item_cnt;
double c_balance;
char c_first[17];
char c_middle[3];
char c_last[17];
char o_entry_d[20];

int         ol_supply_w_id[15];
int         ol_i_id[15];
int         ol_quantity[15];
int         ol_amount[15];
char        ol_delivery_d[15][11];

/*
 * Function: init ordstat transaction
 * Prepare the ordstat transaction
 */
int
init_ords_tx()
{
#if         REAL_ORACLE

#define SQLTXT1 "alter session set isolation_level = serializable"

#define SQLTXT_ZERO                 "BEGIN orderstatus.getstatus_z (
 :w_id, :d_id, \
:c_id, :bylastname, :c_last, :c_first, :c_middle, :c_balance, :o_id, \
:o_entry_d, :o_carrier_id, :item_cnt); END;"
#define SQLTXT_NONZERO              "BEGIN orderstatus.getstatus_nz (
 :w_id, :d_id, \
:c_id, :bylastname, :c_last, :c_first, :c_middle, :c_balance, :o_id, \
:o_entry_d, :o_carrier_id, :item_cnt); END;"

#define     SQLCUR "SELECT ol_i_id, ol_supply_w_id, ol_quantity, ol_amount, \
DELIVR') \
                    NVL(TO_CHAR(ol_delivery_d, 'DD-MM-YYYY'), 'NOT
                    FROM order_line \
= :o_id"
                    WHERE ol_d_id = :d_id AND ol_w_id = :w_id AND ol_o_id

  int i;
  text stmbuf[1024];
```

```
        if (orlon(&tpclda, (ub1 *)tpchda, (text *)"tpcc/tpcc",
                        -1, (text *)0, -1, 0)) {
                        errrpt(&tpclda, &tpclda, "logon");
        }

        /* turn off auto-commit */
        if (ocof(&tpclda)) {
                        errrpt(&tpclda, &tpclda);
                        ologof(&tpclda, "auto-commit");
        }

octx = (ordctx *) malloc (sizeof(ordctx));

/* run in serializable mode */

if (oopen(&curs, &tpclda, (text *) 0, NA, NA, (text *) 0, NA)) {
                        errrpt(&tpclda,&curs,"Open cursor for serializable mode");
                        ologof(&tpclda);
                        return(1);
                        }

 sprintf((char *)stmbuf, SQLTXT1);

        if (oparse(&curs, stmbuf, NA, FALSE, VER7)) {
          errrpt(&tpclda,&curs,"Parse for serializable mode");
          oclose(&curs);
          ologof(&tpclda);
          return(1);
        }

        if (oexec(&curs)) {
                        errrpt(&tpclda,&curs,"oexec for serializable mode");
                        orol(&tpclda);
                        oclose(&curs);
                        ologof(&tpclda);
                        return(1);
                        }

        if (oclose(&curs))
                        errrpt(&tpclda,&curs,"oclose for serializable mode");


OOPEN(&tpclda, &curo0);
OOPEN(&tpclda, &curo1);
OOPEN(&tpclda, &curo2);

sprintf ((char *) stmbuf, SQLTXT_ZERO);
OPARSE(&tpclda, &curo0, stmbuf, NA, FALSE, VER7);
sprintf ((char *) stmbuf, SQLTXT_NONZERO);
OPARSE(&tpclda, &curo1, stmbuf, NA, FALSE, VER7);
sprintf ((char *) stmbuf, SQLCUR);
OPARSE(&tpclda, &curo2, stmbuf, NA, FALSE, VER7);

for (i = 0; i < NITEMS; i++) {
  octx->ol_supply_w_id_ind[i] = TRUE;
  octx->ol_i_id_ind[i] = TRUE;
  octx->ol_quantity_ind[i] = TRUE;
  octx->ol_amount_ind[i] = TRUE;
  octx->ol_delivery_d_ind[i] = TRUE;
    octx->ol_supply_w_id_len[i] = sizeof(int);
    octx->ol_i_id_len[i] = sizeof(int);
    octx->ol_quantity_len[i] = sizeof(int);
    octx->ol_amount_len[i] = sizeof(int);
    octx->ol_delivery_d_len[i] = sizeof(ol_delivery_d[0]);
  }
  octx->ol_supply_w_id_csize = NITEMS;
  octx->ol_i_id_csize = NITEMS;
  octx->ol_quantity_csize = NITEMS;
  octx->ol_amount_csize = NITEMS;
  octx->ol_delivery_d_csize = NITEMS;
  octx->ol_w_id_csize = NITEMS;
  octx->ol_d_id_csize = NITEMS;
  octx->ol_o_id_csize = NITEMS;
  octx->ol_w_id_ind = TRUE;
  octx->ol_d_id_ind = TRUE;
  octx->ol_o_id_ind = TRUE;
  octx->ol_w_id_len = sizeof(int);
  octx->ol_d_id_len = sizeof(int);
  octx->ol_o_id_len = sizeof(int);

  /* bind variables */

  OBNDRV(&tpclda,&curo0,":w_id",ADR(w_id),SIZ(w_id),SQLT_INT);
  OBNDRV(&tpclda,&curo0,":d_id",ADR(d_id),SIZ(d_id),SQLT_INT);
  OBNDRV(&tpclda,&curo0,":c_id",ADR(c_id),SIZ(c_id),SQLT_INT);
OBNDRV(&tpclda,&curo0,":bylastname",ADR(bylastname),SIZ(bylastname),SQLT
_INT);
  OBNDRV(&tpclda,&curo0,":c_last",c_last,SIZ(c_last),SQLT_STR);
  OBNDRV(&tpclda,&curo0,":c_first",c_first,SIZ(c_first),SQLT_STR);
  OBNDRV(&tpclda,&curo0,":c_middle",c_middle,SIZ(c_middle),SQLT_STR);
OBNDRV(&tpclda,&curo0,":c_balance",ADR(c_balance),SIZ(c_balance),SQLT_FL
T);
  OBNDRV(&tpclda,&curo0,":o_id",ADR(o_id),SIZ(o_id),SQLT_INT);
  OBNDRV(&tpclda,&curo0,":o_entry_d",o_entry_d,SIZ(o_entry_d),SQLT_STR);
  OBNDRV(&tpclda,&curo0,":o_carrier_id",ADR(o_carrier_id),SIZ(o_carrier_id),
        SQLT_INT);
  OBNDRV(&tpclda,&curo0,":item_cnt",ADR(item_cnt),SIZ(item_cnt),SQLT_INT);
  OBNDRV(&tpclda,&curo1,":w_id",ADR(w_id),SIZ(w_id),SQLT_INT);
  OBNDRV(&tpclda,&curo1,":d_id",ADR(d_id),SIZ(d_id),SQLT_INT);
  OBNDRV(&tpclda,&curo1,":c_id",ADR(c_id),SIZ(c_id),SQLT_INT);

OBNDRV(&tpclda,&curo1,":bylastname",ADR(bylastname),SIZ(bylastname),SQLT
_INT);
  OBNDRV(&tpclda,&curo1,":c_last",c_last,SIZ(c_last),SQLT_STR);
  OBNDRV(&tpclda,&curo1,":c_first",c_first,SIZ(c_first),SQLT_STR);
  OBNDRV(&tpclda,&curo1,":c_middle",c_middle,SIZ(c_middle),SQLT_STR);

OBNDRV(&tpclda,&curo1,":c_balance",ADR(c_balance),SIZ(c_balance),SQLT_FL
T);
  OBNDRV(&tpclda,&curo1,":o_id",ADR(o_id),SIZ(o_id),SQLT_INT);
  OBNDRV(&tpclda,&curo1,":o_entry_d",o_entry_d,SIZ(o_entry_d),SQLT_STR);
  OBNDRV(&tpclda,&curo1,":o_carrier_id",ADR(o_carrier_id),SIZ(o_carrier_id),
        SQLT_INT);
  OBNDRV(&tpclda,&curo1,":item_cnt",ADR(item_cnt),SIZ(item_cnt),SQLT_INT);

  OBNDRV(&tpclda,&curo2,":w_id",ADR(w_id),SIZ(w_id),SQLT_INT);
  OBNDRV(&tpclda,&curo2,":d_id",ADR(d_id),SIZ(d_id),SQLT_INT);
  OBNDRV(&tpclda,&curo2,":o_id",ADR(o_id),SIZ(o_id),SQLT_INT);
```

```
  ODEFIN(&tpclda, &curo2, (sword)1, ol_i_id,SIZ(int),SQLT_INT,NA,
                octx->ol_i_id_ind,NA,NA,NA,octx->ol_i_id_len,
                octx->ol_i_id_rcode);
  ODEFIN(&tpclda, &curo2, (sword)2, ol_supply_w_id,SIZ(int),SQLT_INT,NA,
                octx->ol_supply_w_id_ind,NA,NA,NA,octx->ol_supply_w_id_len,
                octx->ol_supply_w_id_rcode);
  ODEFIN(&tpclda, &curo2, (sword)3, ol_quantity,SIZ(int),SQLT_INT,NA,
                octx->ol_quantity_ind,NA,NA,NA,octx->ol_quantity_len,
                octx->ol_quantity_rcode);
  ODEFIN(&tpclda, &curo2, (sword)4, ol_amount,SIZ(int),SQLT_INT,NA,
                octx->ol_amount_ind,NA,NA,NA,octx->ol_amount_len,
                octx->ol_amount_rcode);
  ODEFIN(&tpclda, &curo2, (sword)5, ol_delivery_d,11,SQLT_STR,NA,
                octx->ol_delivery_d_ind,NA,NA,NA,octx->ol_delivery_d_len,
                octx->ol_delivery_d_rcode);

#endif

  return (0);

}

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/



ordstat_tx(rqst)
TPSVCINFO *rqst;
{
        struct ord_inf *ordstat_p;
        int i;

        ordstat_p = (struct ord_inf *)(rqst->data);

        MOVETO(w_id, ordstat_p);
        MOVETO(d_id, ordstat_p);
        MOVETO(c_id, ordstat_p);
        tx_count++;

#if ACID
        time(timep);
        userlog("ACID ORDSTAT Transaction begun at %s\n", ctime(timep));
#endif


/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

        if (c_id == 0) {
                bylastname = 1;
                strcpy(c_last, ordstat_p->c_last);
        }
        else {
                bylastname = 0;
                c_last[1] = '\0';
        }

  for (i = 0; i < NITEMS; i++) {
    octx->ol_supply_w_id_ind[i] = TRUE;
    octx->ol_i_id_ind[i] = TRUE;
    octx->ol_quantity_ind[i] = TRUE;
    octx->ol_amount_ind[i] = TRUE;
    octx->ol_delivery_d_ind[i] = TRUE;
    octx->ol_supply_w_id_len[i] = sizeof(int);
    octx->ol_i_id_len[i] = sizeof(int);
    octx->ol_quantity_len[i] = sizeof(int);
    octx->ol_amount_len[i] = sizeof(int);
    octx->ol_delivery_d_len[i] = sizeof(ol_delivery_d[0]);
  }
  octx->ol_supply_w_id_csize = NITEMS;
  octx->ol_i_id_csize = NITEMS;
  octx->ol_quantity_csize = NITEMS;
  octx->ol_amount_csize = NITEMS;
  octx->ol_delivery_d_csize = NITEMS;

#if     REAL_ORACLE
        if (bylastname) {
                OEXEC(&tpclda, &curo1);
        }
        else {
                OEXEC(&tpclda,&curo0);
        }


        octx->ol_w_id_ind = TRUE;
        octx->ol_d_id_ind = TRUE;
        octx->ol_o_id_ind = TRUE;
        octx->ol_w_id_len = sizeof(int);
        octx->ol_d_id_len = sizeof(int);
        octx->ol_o_id_len = sizeof(int);

        OEXFET( &tpclda, &curo2, item_cnt, NULL, NULL);
    OCOM(&tpclda,&tpclda);
#endif  /*REAL_ORACLE*/

#if ACID
        time(timep);
        userlog("ACID ORDSTAT for w_id = %d, d_id = %d, c_id = %d, o_id =
%d\n",
                w_id, d_id, c_id, o_id);
        time(timep);
        userlog("ACID ORDSTAT Transaction completed at %s\n",
 ctime(timep));
#endif
        for (i = 0; i < item_cnt; i++) {
                ol_delivery_d[i][10] = '\0';
                ordstat_p->o_items[i].ol_supply_w_id = ol_supply_w_id[i];
                ordstat_p->o_items[i].ol_i_id = ol_i_id[i];
                ordstat_p->o_items[i].ol_quantity = ol_quantity[i];
                ordstat_p->o_items[i].ol_amount =
 ((double)ol_amount[i])/100;
                strcpy(ordstat_p->o_items[i].ol_delivery_d,
 ol_delivery_d[i]);
        }

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/
```

```
        MOVEBACK(c_id, ordstat_p);
        MOVEBACK(o_id, ordstat_p);
        MOVEBACK(o_carrier_id, ordstat_p);
        MOVEBACK(item_cnt, ordstat_p);
        ordstat_p->c_balance = c_balance/100;
        MOVECBACK(c_first, 16, ordstat_p);
        MOVECBACK(c_middle, 2, ordstat_p);
        MOVECBACK(c_last, 16, ordstat_p);
        MOVECBACK(o_entry_d, 19, ordstat_p);
        return(0);
}

tpsvrinit(argc, argv)
char **argv;
{
        if (init_ords_tx())             /* Prepare transaction */
                return(1);
        else
                return(0);
}

void
tpsvrdone()
{
#if     REAL_ORACLE
  oclose (&curo0);
  oclose (&curo1);
  oclose (&curo2);

  /* log off */
  ologof (&tpclda);
#endif  /*REAL_ORACLE*/
}

ORDS(rqst)
TPSVCINFO *rqst;
{
#if     REAL_ORACLE
        if (ordstat_tx(rqst) )
                tpreturn(TPFAIL, 0, rqst->data, sizeof(struct ord_inf), 0);
        else
                tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct ord_inf),
0);
#else   /*REAL_ORACLE */
        sleep(1);
        tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct ord_inf), 0);
#endif  /*REAL_ORACLE*/
}
```

tpcc_srv_paym.c

```
#define REAL_ORACLE 1     /* Compile with connections to RDMS */
/*
 * Copyright (c) 1994 by Sun Microsystems, Inc.
 */

#pragma ident "@(#)tpcc_srv_paym.c      1.17    96/01/11    SMI"

/*
 * File: paym.ec
 * Payment transaction code for Informix Tuxedo
 * Author : Shanti S
 * Date   : 5/21/93
 * Ported to Oracle 8/25/94 - dbt
 *
 */

#define SQLNET

#include "oci.h"

#include <stdio.h>
#if REAL_ORACLE
#include "./ora_err.h"
#endif /* REAL_ORACLE */

static int      tx_count=0;

/* Tuxedo */
#include "atmi.h"
#include "userlog.h"


#define MOVETO(element, struct_name) \
                element = struct_name -> element
#define MOVEBACK(element, struct_name) \
                struct_name -> element = element
#define MOVECTO(element, cnt, struct_name) {                    \
                int     i;                      \
                strncpy(element, struct_name -> element, cnt); \
                element[cnt] = '\0';                    \
                for(i=0; i<=cnt; i++)                   \
                {                               \
                    if(isspace(element[i]))         \
                    {                           \
                        element[i] = '\0';      \
                        break;                  \
                    }                           \
                }                               \
            }
#define MOVECBACK(element, cnt, struct_name) \
                strncpy(struct_name -> element, element, cnt)

struct pay_inf {
        int     w_id;
        int     d_id;
        int     c_id;
        int     c_w_id;
        int     c_d_id;
        double h_amount;
        double c_credit_lim;
        double c_balance;
        double c_discount;
        char h_date[20];
        char w_street_1[21];
        char w_street_2[21];
        char w_city[21];
        char w_state[3];
        char w_zip[11];
        char d_street_1[21];
```

```
        char d_street_2[21];
        char d_city[21];
        char d_state[3];
        char d_zip[11];
        char c_first[17];
        char c_middle[3];
        char c_last[17];
        char c_street_1[21];
        char c_street_2[21];
        char c_city[21];
        char c_state[3];
        char c_zip[11];
        char c_phone[17];
        char c_since[11];
        char c_credit[3];
        char c_data_1[51];
        char c_data_2[51];
        char c_data_3[51];
        char c_data_4[51];
};



#if ACID
#include <sys/types.h>
#include <time.h>
time_t curtime;
time_t *timep = &curtime;
#endif

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

#if REAL_ORACLE
unsigned char cr_date[7];
ldadef tpclda;
csrdef curs, curp0, curp1;
unsigned long tpchda[256];
#endif /* REAL_ORACLE */

/* List of fields in payment */

int retry;
char c_data[201];

int             w_id;
int             d_id;
int             c_id, bylastname;
int             c_w_id;
int             c_d_id;
int h_amount;
int c_credit_lim;
double c_balance;
int c_discount;
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[11];
char c_credit[3];
int             retries;

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/

/*
 * Function: init payment transaction
 * Prepare the payment transaction
 */


/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

int
init_paym_tx()
{

#if REAL_ORACLE
#define SQLTXT1 "alter session set isolation_level = serializable"

#define SQLTXT_ZERO "BEGIN payment.dopayment_z (:w_id, :d_id, :c_w_id,
 :c_d_id, \
   :c_id, :byln, \
   :h_amount, :c_last, :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, :c_data, \
   :h_date, :retry, :cr_date); END;"

#define SQLTXT_NONZERO "BEGIN payment.dopayment_nz (:w_id, :d_id,
 :c_w_id, :c_d_id, \
   :c_id, :byln, \
   :h_amount, :c_last, :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, :c_data, \
   :h_date, :retry, :cr_date); END;"


                text stmbuf[1024];
```

```
#ifdef TKPROF
        EXEC SQL ALTER SESSION SET SQL_TRACE = TRUE;
#endif

    if (orlon(&tpclda, (ub1 *)tpchda, (text *)"tpcc/tpcc",
        -1, (text *)0, -1, 0)) {
        errrpt(&tpclda, &tpclda, "logon");
        return(1);
    }

    /* turn off auto-commit */
    if (ocof(&tpclda)) {
        errrpt(&tpclda, &tpclda, "auto-commit");
        ologof(&tpclda);
        return(1);
    }
    vgetdate(cr_date);
#if VER73
   /* run in serializable mode */

   if (oopen(&curs, &tpclda)) {
                errrpt(&tpclda,&curs,"Open cursor for serializable mode");
                ologof(&tpclda);
                return(1);
                }

    sprintf((char *)stmbuf, SQLTXT1);

        if (oparse(&curs, stmbuf, NA, FALSE, VER7)) {
          errrpt(&tpclda,&curs,"Parse for serializable mode");
          oclose(&curs);
          ologof(&tpclda);
          return(1);
        }

        if (oexec(&curs)) {
                errrpt(&tpclda,&curs,"oexec for serializable mode");
                orol(&tpclda);
                oclose(&curs);
                ologof(&tpclda);
                return(1);
                }

        if (oclose(&curs)) {
                errrpt(&tpclda,&curs,"oclose for serializable mode");
                return(1);
                }
#endif
        OOPEN(&tpclda, &curp0);
        OOPEN(&tpclda, &curp1);
        sprintf((char *)stmbuf, SQLTXT_ZERO);
        OPARSE(&tpclda, &curp0, stmbuf, NA, FALSE, VER7);
        sprintf((char *)stmbuf, SQLTXT_NONZERO);
        OPARSE(&tpclda, &curp1, stmbuf, NA, FALSE, VER7);

   /* bind variables */


   OBNDRV(&tpclda,&curp0,":w_id",ADR(w_id),SIZ(w_id),SQLT_INT);
   OBNDRV(&tpclda,&curp0,":d_id",ADR(d_id),SIZ(d_id),SQLT_INT);
   OBNDRV(&tpclda,&curp0,":c_w_id",ADR(c_w_id),SIZ(c_w_id),SQLT_INT);
   OBNDRV(&tpclda,&curp0,":c_d_id",ADR(c_d_id),SIZ(c_d_id),SQLT_INT);
   OBNDRV(&tpclda,&curp0,":c_id",ADR(c_id),SIZ(c_id),SQLT_INT);
OBNDRV(&tpclda,&curp0,":byln",ADR(bylastname),SIZ(bylastname),SQLT_INT);
OBNDRV(&tpclda,&curp0,":h_amount",ADR(h_amount),SIZ(h_amount),SQLT_INT
);
   OBNDRV(&tpclda,&curp0,":c_last",c_last,SIZ(c_last),SQLT_STR);
OBNDRV(&tpclda,&curp0,":w_street_1",w_street_1,SIZ(w_street_1),SQLT_STR);
OBNDRV(&tpclda,&curp0,":w_street_2",w_street_2,SIZ(w_street_2),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":w_city",w_city,SIZ(w_city),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":w_state",w_state,SIZ(w_state),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":w_zip",w_zip,SIZ(w_zip),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":d_street_1",d_street_1,SIZ(d_street_1),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":d_street_2",d_street_2,SIZ(d_street_2),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":d_city",d_city,SIZ(d_city),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":d_state",d_state,SIZ(d_state),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":d_zip",d_zip,SIZ(d_zip),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_first",c_first,SIZ(c_first),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_middle",c_middle,SIZ(c_middle),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_street_1",c_street_1,SIZ(c_street_1),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_street_2",c_street_2,SIZ(c_street_2),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_city",c_city,SIZ(c_city),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_state",c_state,SIZ(c_state),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_zip",c_zip,SIZ(c_zip),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_phone",c_phone,SIZ(c_phone),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_since",c_since,SIZ(c_since),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":c_credit",c_credit,SIZ(c_credit)-1,SQLT_CHR);
   OBNDRV(&tpclda,&curp0,":c_credit_lim",ADR(c_credit_lim),SIZ(c_credit_lim),
        SQLT_INT);
   OBNDRV(&tpclda,&curp0,":c_discount",ADR(c_discount),SIZ(c_discount),
        SQLT_INT);

OBNDRV(&tpclda,&curp0,":c_balance",ADR(c_balance),SIZ(c_balance),SQLT_FL
T);
   OBNDRV(&tpclda,&curp0,":c_data",c_data,SIZ(c_data),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":h_date",h_date,SIZ(h_date),SQLT_STR);
   OBNDRV(&tpclda,&curp0,":retry",ADR(retries),SIZ(retries),SQLT_INT);
   OBNDRV(&tpclda,&curp0,":cr_date",ADR(cr_date),SIZ(cr_date),SQLT_DAT);

   OBNDRV(&tpclda,&curp1,":w_id",ADR(w_id),SIZ(w_id),SQLT_INT);
   OBNDRV(&tpclda,&curp1,":d_id",ADR(d_id),SIZ(d_id),SQLT_INT);
   OBNDRV(&tpclda,&curp1,":c_w_id",ADR(c_w_id),SIZ(c_w_id),SQLT_INT);
   OBNDRV(&tpclda,&curp1,":c_d_id",ADR(c_d_id),SIZ(c_d_id),SQLT_INT);
   OBNDRV(&tpclda,&curp1,":c_id",ADR(c_id),SIZ(c_id),SQLT_INT);

OBNDRV(&tpclda,&curp1,":byln",ADR(bylastname),SIZ(bylastname),SQLT_INT);

OBNDRV(&tpclda,&curp1,":h_amount",ADR(h_amount),SIZ(h_amount),SQLT_INT
);
   OBNDRV(&tpclda,&curp1,":c_last",c_last,SIZ(c_last),SQLT_STR);

OBNDRV(&tpclda,&curp1,":w_street_1",w_street_1,SIZ(w_street_1),SQLT_STR);

OBNDRV(&tpclda,&curp1,":w_street_2",w_street_2,SIZ(w_street_2),SQLT_STR);
   OBNDRV(&tpclda,&curp1,":w_city",w_city,SIZ(w_city),SQLT_STR);
   OBNDRV(&tpclda,&curp1,":w_state",w_state,SIZ(w_state),SQLT_STR);
   OBNDRV(&tpclda,&curp1,":w_zip",w_zip,SIZ(w_zip),SQLT_STR);
   OBNDRV(&tpclda,&curp1,":d_street_1",d_street_1,SIZ(d_street_1),SQLT_STR);
   OBNDRV(&tpclda,&curp1,":d_street_2",d_street_2,SIZ(d_street_2),SQLT_STR);
   OBNDRV(&tpclda,&curp1,":d_city",d_city,SIZ(d_city),SQLT_STR);
```

```
  OBNDRV(&tpclda,&curp1,":d_state",d_state,SIZ(d_state),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":d_zip",d_zip,SIZ(d_zip),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_first",c_first,SIZ(c_first),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_middle",c_middle,SIZ(c_middle),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_street_1",c_street_1,SIZ(c_street_1),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_street_2",c_street_2,SIZ(c_street_2),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_city",c_city,SIZ(c_city),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_state",c_state,SIZ(c_state),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_zip",c_zip,SIZ(c_zip),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_phone",c_phone,SIZ(c_phone),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_since",c_since,SIZ(c_since),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":c_credit",c_credit,SIZ(c_credit)-1,SQLT_CHR);
  OBNDRV(&tpclda,&curp1,":c_credit_lim",ADR(c_credit_lim),SIZ(c_credit_lim),
      SQLT_INT);
  OBNDRV(&tpclda,&curp1,":c_discount",ADR(c_discount),SIZ(c_discount),
      SQLT_INT);
OBNDRV(&tpclda,&curp1,":c_balance",ADR(c_balance),SIZ(c_balance),SQLT_FL
T);
  OBNDRV(&tpclda,&curp1,":c_data",c_data,SIZ(c_data),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":h_date",h_date,SIZ(h_date),SQLT_STR);
  OBNDRV(&tpclda,&curp1,":retry",ADR(retries),SIZ(retries),SQLT_INT);
  OBNDRV(&tpclda,&curp1,":cr_date",ADR(cr_date),SIZ(cr_date),SQLT_DAT);

#endif /* REAL_ORACLE */

        return(0);
}


/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/


payment_tx(rqst)
TPSVCINFO *rqst;
{

        struct pay_inf *payment_p;
        payment_p = (struct pay_inf *)(rqst->data);

   vgetdate(cr_date);
      MOVETO(w_id, payment_p);
      MOVETO(d_id, payment_p);
      MOVETO(c_id, payment_p);
      MOVETO(c_w_id, payment_p);
      MOVETO(c_d_id, payment_p);
        h_amount = (int)(payment_p->h_amount * 100);
        MOVECTO(c_last, 16, payment_p);
        tx_count++;

#if ACID
        time(timep);
        userlog("ACID PAYMENT Transaction Begun at %s\n", ctime(timep));
#endif

/*****************************
 * BEGIN BLOCK OF COMMON CODE
 *****************************/

#if     REAL_ORACLE
        if (c_id == 0) {
                bylastname = 1;
                OEXEC(&tpclda, &curp1);
        }
        else {
                bylastname = 0;
                OEXEC(&tpclda, &curp0);
        }
        retries=0;
#endif  /*REAL_ORACLE*/

/*****************************
 * END BLOCK OF COMMON CODE
 *****************************/

#if ACID
        time(timep);
        userlog("w_id %d, d_id %d, c_id %d, h_amount = %f, c_balance =
%f\n",
                w_id, d_id, c_id, (float)h_amount/100, c_balance);
        userlog("ACID PAYMENT Transaction completed at %s\n",
 ctime(timep));
#endif
      MOVEBACK(c_id, payment_p);
        payment_p->c_credit_lim = ((double)c_credit_lim) / 100;
        payment_p->c_balance = ((double)c_balance) / 100;
        payment_p->c_discount = ((double)c_discount) / 10000;
      MOVECBACK(h_date, 19, payment_p);
      MOVECBACK(w_street_1, 20, payment_p);
      MOVECBACK(w_street_2, 20, payment_p);
      MOVECBACK(w_city, 20, payment_p);
      MOVECBACK(w_state, 2, payment_p);
      MOVECBACK(w_zip, 10, payment_p);
      MOVECBACK(d_street_1, 20, payment_p);
      MOVECBACK(d_street_2, 20, payment_p);
      MOVECBACK(d_city, 20, payment_p);
      MOVECBACK(d_state, 2, payment_p);
      MOVECBACK(d_zip, 10, payment_p);
      MOVECBACK(c_first, 16, payment_p);
      MOVECBACK(c_middle, 2, payment_p);
      MOVECBACK(c_last, 16, payment_p);
      MOVECBACK(c_street_1, 20, payment_p);
      MOVECBACK(c_street_2, 20, payment_p);
      MOVECBACK(c_city, 20, payment_p);
      MOVECBACK(c_state, 2, payment_p);
      MOVECBACK(c_zip, 10, payment_p);
      MOVECBACK(c_phone, 16, payment_p);
      MOVECBACK(c_since, 10, payment_p);
      MOVECBACK(c_credit, 2, payment_p);
                strncpy(payment_p->c_data_1, c_data, 50);
                strncpy(payment_p->c_data_2, c_data+50, 50);
                strncpy(payment_p->c_data_3, c_data+100, 50);
                strncpy(payment_p->c_data_4, c_data+150, 50);
                return(0);
}

/* Tuxedo code */
tpsvrinit(argc, argv)
char **argv;
```

```
{
        return(init_paym_tx());         /* Prepare transaction */
}

void
tpsvrdone()
{
  oclose (&curp0);
  oclose (&curp1);
  /* log off */
  ologof (&tpclda);
}

PAYM(rqst)
TPSVCINFO *rqst;
{
        if (payment_tx(rqst) )
                tpreturn(TPFAIL, 0, rqst->data, sizeof(struct pay_inf), 0);
        else
                tpreturn(TPSUCCESS, 0, rqst->data, sizeof(struct pay_inf),
0);
}


vgetdate(unsigned char *buf)
{
 time_t tloc;
 char temp[5];
 int cen;

 if(time(&tloc) == (time_t)-1) {
  printf("Error getting date\n");
  exit(1);
 }
 cftime(temp,"%d",&tloc);
 buf[3] = (unsigned char)atoi(temp);
 cftime(temp,"%m",&tloc);
 buf[2] = (unsigned char)atoi(temp);
 cftime(temp,"%Y",&tloc);
 cen = atoi(temp);
 buf[0] = (unsigned char)((cen/100)+100);
 buf[1] = (unsigned char)((cen%100)+100);
 cftime(temp,"%H",&tloc);
 buf[4] = (unsigned char)(atoi(temp) + 1);
 cftime(temp,"%M",&tloc);
 buf[5] = (unsigned char)(atoi(temp) + 1);
 cftime(temp,"%S",&tloc);
 buf[6] = (unsigned char)(atoi(temp) + 1);
}
```

Stored Procedures

new.sql

```
rem
rem
================================================================
====+
rem        Copyright (c) 1993  Oracle Corp, Belmont, CA         |
rem             OPEN SYSTEMS PERFORMANCE GROUP                  |
rem                    All Rights Reserved                      |
rem
 ================================================================
====+
rem  FILENAME
rem       new.sql
rem  DESCRIPTION
rem       SQL script to create a stored package for new order
rem       transactions.
rem
================================================================
====
rem

CREATE OR REPLACE PACKAGE neworder
IS
  PROCEDURE enterorder
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id               INTEGER,
    ord_ol_cnt            INTEGER,
    ord_all_local         INTEGER,
    cust_discount       OUT NUMBER,
    cust_last           OUT VARCHAR2,
    cust_credit         OUT VARCHAR2,
    dist_tax            OUT NUMBER,
    ware_tax            OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d      IN OUT VARCHAR2,
    retry            IN OUT INTEGER,
    cur_date                                    IN    DATE
  );
END neworder;
/

CREATE OR REPLACE PACKAGE BODY neworder
IS
  PROCEDURE enterorder
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id               INTEGER,
    ord_ol_cnt            INTEGER,
    ord_all_local         INTEGER,
    cust_discount       OUT NUMBER,
    cust_last           OUT VARCHAR2,
    cust_credit         OUT VARCHAR2,
    dist_tax            OUT NUMBER,
    ware_tax            OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d      IN OUT VARCHAR2,
    retry            IN OUT INTEGER,
                cur_date                        IN    DATE
  )
  IS
    timestamp             DATE;
                dist_rowid rowid;
    not_serializable      EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock              EXCEPTION;
```

```
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old        EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  BEGIN
    LOOP BEGIN
      SELECT district.rowid, d_tax, d_next_o_id, w_tax
        INTO dist_rowid, dist_tax, ord_id, ware_tax
        FROM district, warehouse
        WHERE d_id = dist_id AND d_w_id = ware_id
                                    AND w_id = ware_id;
      UPDATE district SET d_next_o_id = ord_id + 1
        WHERE rowid = dist_rowid;
      SELECT c_discount, c_last, c_credit
        INTO cust_discount, cust_last, cust_credit
        FROM customer
        WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id = ware_id;
      timestamp := cur_date;
      ord_entry_d := TO_CHAR(timestamp,'DD-MM-YYYY.HH24:MI:SS');
      INSERT INTO new_order VALUES (ord_id, dist_id, ware_id);
      INSERT INTO orders VALUES (ord_id, dist_id, ware_id, cust_id,
                          timestamp, 11, ord_ol_cnt, ord_all_local);
      EXIT;

      EXCEPTION
        WHEN not_serializable OR deadlock OR snapshot_too_old THEN
          ROLLBACK;
          retry := retry + 1;
      END;
    END LOOP;
END enterorder;

END neworder;
/
show errors;
/
```

ord.sql

```
rem
rem
=============================================================
====+
rem          Copyright (c) 1993  Oracle Corp, Belmont, CA          |
rem               OPEN SYSTEMS PERFORMANCE GROUP                |
rem                   All Rights Reserved                       |
rem
=============================================================
====+
rem  FILENAME
rem      ord.sql
rem  DESCRIPTION
rem      SQL script to create a stored package for order status
rem      transactions.
rem
=============================================================
====
rem

CREATE OR REPLACE PACKAGE orderstatus
IS
  TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
  PROCEDURE getstatus_z
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id          IN OUT INTEGER,
    bylastname            INTEGER,
    cust_last        IN OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_balance        OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d         OUT VARCHAR2,
    ord_carrier_id      OUT INTEGER,
    ord_ol_cnt          OUT INTEGER
  );
  PROCEDURE getstatus_nz
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id          IN OUT INTEGER,
    bylastname            INTEGER,
    cust_last        IN OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_balance        OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d         OUT VARCHAR2,
    ord_carrier_id      OUT INTEGER,
    ord_ol_cnt          OUT INTEGER
  );
END;
/

CREATE OR REPLACE PACKAGE BODY orderstatus
IS
  PROCEDURE getstatus_z
  (
    ware_id               INTEGER,
    dist_id               INTEGER,
    cust_id          IN OUT INTEGER,
    bylastname            INTEGER,
    cust_last        IN OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_balance        OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d         OUT VARCHAR2,
    ord_carrier_id      OUT INTEGER,
    ord_ol_cnt          OUT INTEGER
  )
  IS
    cust_rowid            ROWID;
    ol                  BINARY_INTEGER;
    c_num               BINARY_INTEGER;
    row_id              rowidarray;
          not_serializable                    EXCEPTION;
          PRAGMA EXCEPTION_INIT (not_serializable, -8177);
          deadlock                EXCEPTION;
          PRAGMA EXCEPTION_INIT (deadlock, -60);
```

```
            snapshot_too_old                EXCEPTION;
            PRAGMA EXCEPTION_INIT (snapshot_too_old, -1555);
    CURSOR mo_cur IS
      SELECT o_id, to_char(o_entry_d,'DD-MM-YYYY.HH24:MI:SS'),
                                    o_carrier_id, o_ol_cnt
      FROM orders
      WHERE o_d_id = dist_id AND o_w_id = ware_id AND o_c_id = cust_id
                        ORDER BY  o_w_id, o_d_id, o_c_id, o_id DESC;
  BEGIN

            LOOP BEGIN
      SELECT c_balance, c_first, c_middle, c_last
        INTO cust_balance, cust_first, cust_middle, cust_last
        FROM customer
        WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id = ware_id;
                        OPEN mo_cur;
                        FETCH mo_cur INTO ord_id, ord_entry_d, ord_carrier_id,
ord_ol_cnt;
                        CLOSE mo_cur;
            EXIT;
            EXCEPTION
                        WHEN not_serializable OR deadlock OR snapshot_too_old
THEN
                                    ROLLBACK;
            END;
            END LOOP;
  END;

  PROCEDURE getstatus_nz
  (
    ware_id             INTEGER,
    dist_id             INTEGER,
    cust_id          IN OUT INTEGER,
    bylastname          INTEGER,
    cust_last        IN OUT VARCHAR2,
    cust_first       OUT VARCHAR2,
    cust_middle      OUT VARCHAR2,
    cust_balance     OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d      OUT VARCHAR2,
    ord_carrier_id   OUT INTEGER,
    ord_ol_cnt       OUT INTEGER
  )
  IS
    cust_rowid          ROWID;
    ol                  BINARY_INTEGER;
    c_num               BINARY_INTEGER;
    row_id              rowidarray;
            not_serializable                EXCEPTION;
            PRAGMA EXCEPTION_INIT (not_serializable, -8177);
            deadlock                EXCEPTION;
            PRAGMA EXCEPTION_INIT (deadlock, -60);
            snapshot_too_old                EXCEPTION;
            PRAGMA EXCEPTION_INIT (snapshot_too_old, -1555);
            CURSOR c_cur IS
                        SELECT rowid FROM customer
                        WHERE c_d_id = dist_id AND c_w_id = ware_id AND
c_last = cust_last
                        ORDER BY c_w_id, c_d_id, c_last, c_first;
    CURSOR mo_cur IS
      SELECT o_id, to_char(o_entry_d,'DD-MM-YYYY.HH24:MI:SS'),
                                    o_carrier_id, o_ol_cnt
      FROM orders
      WHERE o_d_id = dist_id AND o_w_id = ware_id AND o_c_id = cust_id
                        ORDER BY  o_w_id, o_d_id, o_c_id, o_id DESC;
  BEGIN

            LOOP BEGIN
      c_num := 0;
      FOR c_id_rec IN c_cur LOOP
        c_num := c_num + 1;
        row_id(c_num) := c_id_rec.rowid;
      END LOOP;
      cust_rowid := row_id ((c_num + 1) / 2);

      SELECT c_id, c_balance, c_first, c_middle, c_last
        INTO cust_id, cust_balance, cust_first, cust_middle, cust_last
        FROM customer
        WHERE rowid = cust_rowid;

                        OPEN mo_cur;
                        FETCH mo_cur INTO ord_id, ord_entry_d, ord_carrier_id,
ord_ol_cnt;
                        CLOSE mo_cur;
            EXIT;
            EXCEPTION
                        WHEN not_serializable OR deadlock OR snapshot_too_old
THEN
                                    ROLLBACK;
            END;
            END LOOP;
    END;
END;
/
show errors;
```

pay.sql

```
rem
rem
================================================================
====+
rem           Copyright (c) 1993  Oracle Corp, Belmont, CA          |
rem                OPEN SYSTEMS PERFORMANCE GROUP                   |
rem                   All Rights Reserved                           |
rem
================================================================
===+
rem  FILENAME
rem       pay.sql
rem  DESCRIPTION
rem       SQL script to create a stored procedure for payment
rem       transactions.
rem
================================================================
====
rem

CREATE OR REPLACE PACKAGE payment
```

```
IS
  PROCEDURE dopayment_z
  (
    ware_id              INTEGER,
    dist_id              INTEGER,
    cust_w_id            INTEGER,
    cust_d_id            INTEGER,
    cust_id         IN OUT INTEGER,
                bylastname                          INTEGER,
    hist_amount          PLS_INTEGER,
    cust_last       IN OUT VARCHAR2,
    ware_street_1       OUT VARCHAR2,
    ware_street_2       OUT VARCHAR2,
    ware_city           OUT VARCHAR2,
    ware_state          OUT VARCHAR2,
    ware_zip            OUT VARCHAR2,
    dist_street_1       OUT VARCHAR2,
    dist_street_2       OUT VARCHAR2,
    dist_city           OUT VARCHAR2,
    dist_state          OUT VARCHAR2,
    dist_zip            OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_street_1       OUT VARCHAR2,
    cust_street_2       OUT VARCHAR2,
    cust_city           OUT VARCHAR2,
    cust_state          OUT VARCHAR2,
    cust_zip            OUT VARCHAR2,
    cust_phone          OUT VARCHAR2,
    cust_since          OUT VARCHAR2,
    cust_credit      IN OUT VARCHAR2,
    cust_credit_lim     OUT NUMBER,
    cust_discount       OUT NUMBER,
    cust_balance     IN OUT NUMBER,
    cust_data           OUT VARCHAR2,
    hist_date           OUT VARCHAR2,
    retry            IN OUT INTEGER,
                cur_date                        IN     DATE
  );

  PROCEDURE dopayment_nz
  (
    ware_id              INTEGER,
    dist_id              INTEGER,
    cust_w_id            INTEGER,
    cust_d_id            INTEGER,
    cust_id         IN OUT INTEGER,
                bylastname                          INTEGER,
    hist_amount          PLS_INTEGER,
    cust_last       IN OUT VARCHAR2,
    ware_street_1       OUT VARCHAR2,
    ware_street_2       OUT VARCHAR2,
    ware_city           OUT VARCHAR2,
    ware_state          OUT VARCHAR2,
    ware_zip            OUT VARCHAR2,
    dist_street_1       OUT VARCHAR2,
    dist_street_2       OUT VARCHAR2,
    dist_city           OUT VARCHAR2,
    dist_state          OUT VARCHAR2,
    dist_zip            OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_street_1       OUT VARCHAR2,
    cust_street_2       OUT VARCHAR2,
    cust_city           OUT VARCHAR2,
    cust_state          OUT VARCHAR2,
    cust_zip            OUT VARCHAR2,
    cust_phone          OUT VARCHAR2,
    cust_since          OUT VARCHAR2,
    cust_credit      IN OUT VARCHAR2,
    cust_credit_lim     OUT NUMBER,
    cust_discount       OUT NUMBER,
    cust_balance     IN OUT NUMBER,
    cust_data           OUT VARCHAR2,
    hist_date           OUT VARCHAR2,
    retry            IN OUT INTEGER,
                cur_date                        IN     DATE
  );
END;
/

CREATE OR REPLACE PACKAGE BODY payment
IS
  PROCEDURE dopayment_z
  (
    ware_id              INTEGER,
    dist_id              INTEGER,
    cust_w_id            INTEGER,
    cust_d_id            INTEGER,
    cust_id         IN OUT INTEGER,
                bylastname                          INTEGER,
    hist_amount          PLS_INTEGER,
    cust_last       IN OUT VARCHAR2,
    ware_street_1       OUT VARCHAR2,
    ware_street_2       OUT VARCHAR2,
    ware_city           OUT VARCHAR2,
    ware_state          OUT VARCHAR2,
    ware_zip            OUT VARCHAR2,
    dist_street_1       OUT VARCHAR2,
    dist_street_2       OUT VARCHAR2,
    dist_city           OUT VARCHAR2,
    dist_state          OUT VARCHAR2,
    dist_zip            OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_street_1       OUT VARCHAR2,
    cust_street_2       OUT VARCHAR2,
    cust_city           OUT VARCHAR2,
    cust_state          OUT VARCHAR2,
    cust_zip            OUT VARCHAR2,
    cust_phone          OUT VARCHAR2,
    cust_since          OUT VARCHAR2,
    cust_credit      IN OUT VARCHAR2,
    cust_credit_lim     OUT NUMBER,
    cust_discount       OUT NUMBER,
    cust_balance     IN OUT NUMBER,
    cust_data           OUT VARCHAR2,
    hist_date           OUT VARCHAR2,
    retry            IN OUT INTEGER,
                cur_date                        IN             DATE
```

```
)
IS
  TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
  cust_rowid            ROWID;
  ware_rowid            ROWID;
  dist_ytd              NUMBER(12);
  dist_name             VARCHAR2(11);
  ware_ytd              NUMBER(12);
  ware_name             VARCHAR2(11);
  history_date          DATE;
  c_num                 BINARY_INTEGER;
  row_id                rowidarray;
          cust_payments                                 PLS_INTEGER;
          cust_ytd                                      NUMBER(12);
          cust_data_temp                        VARCHAR2(500);
        not_serializable      EXCEPTION;
        PRAGMA EXCEPTION_INIT(not_serializable,-8177);
        deadlock              EXCEPTION;
 PRAGMA EXCEPTION_INIT(deadlock,-60);
        snapshot_too_old      EXCEPTION;
 PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);

 BEGIN
  LOOP BEGIN
        SELECT rowid, c_first, c_middle, c_last, c_street_1, c_street_2,
        c_city, c_state, c_zip, c_phone,
        to_char (c_since, 'DD-MM-YYYY'), c_credit, c_credit_lim,
        c_discount, c_balance - hist_amount, c_payment_cnt,
                                              c_ytd_payment + hist_amount,
decode(c_credit, 'BC', c_data, ' ')
      INTO cust_rowid, cust_first, cust_middle, cust_last,
        cust_street_1, cust_street_2, cust_city, cust_state,
        cust_zip, cust_phone, cust_since, cust_credit,
        cust_credit_lim, cust_discount, cust_balance ,
                                                      cust_payments,
cust_ytd, cust_data_temp
      FROM customer
                                WHERE c_id = cust_id AND c_d_id =
cust_d_id AND
                                                      c_w_id = cust_w_id;
                              cust_payments := cust_payments + 1;

    IF cust_credit = 'BC' THEN
                              cust_data_temp := substr ((to_char (cust_id) || '
' ||
                    to_char (cust_d_id) || ' ' ||
                    to_char (cust_w_id) || ' ' ||
                    to_char (dist_id) || ' ' ||
                    to_char (ware_id) || ' ' ||
                    to_char (hist_amount/100, '9999.99') || ' | ')
                   || cust_data_temp, 1, 500);
      UPDATE customer
        SET c_balance = cust_balance,
          c_ytd_payment = cust_ytd,
          c_payment_cnt = cust_payments,
                                              c_data = cust_data_temp
        WHERE rowid = cust_rowid;
      cust_data := substr(cust_data_temp, 1, 200);
    ELSE
      UPDATE customer
        SET c_balance = cust_balance,
          c_ytd_payment = cust_ytd,
          c_payment_cnt = cust_payments
        WHERE rowid = cust_rowid;

      cust_data := cust_data_temp;
    END IF;

    SELECT district.rowid, d_name, d_street_1, d_street_2, d_city,
                                    d_state, d_zip, d_ytd + hist_amount,
        warehouse.rowid, w_name, w_street_1, w_street_2, w_city,
                                    w_state, w_zip, w_ytd + hist_amount
      INTO cust_rowid, dist_name, dist_street_1, dist_street_2,
                                        dist_city, dist_state, dist_zip,
dist_ytd,
        ware_rowid, ware_name, ware_street_1, ware_street_2,
                                        ware_city, ware_state, ware_zip,
ware_ytd
      FROM district, warehouse
      WHERE d_id = dist_id AND d_w_id = ware_id AND w_id = ware_id;
    UPDATE district
      SET d_ytd = dist_ytd WHERE rowid = cust_rowid;

    UPDATE warehouse
      SET w_ytd = ware_ytd WHERE rowid = ware_rowid;

                        history_date := cur_date;

    INSERT INTO history VALUES
      (cust_id, cust_d_id, cust_w_id, dist_id, ware_id, history_date,
       hist_amount, ware_name || '    ' || dist_name);

    COMMIT;
    hist_date := to_char (history_date, 'DD-MM-YYYY.HH24:MI:SS');
    EXIT;

                        EXCEPTION
                            WHEN not_serializable OR deadlock OR
snapshot_too_old THEN
                                ROLLBACK;
                                retry := retry + 1;
                        END;

  END LOOP;
 END;

 PROCEDURE dopayment_nz
 (
  ware_id               INTEGER,
  dist_id               INTEGER,
  cust_w_id             INTEGER,
  cust_d_id             INTEGER,
  cust_id         IN OUT INTEGER,
            bylastname                                  INTEGER,
  hist_amount           PLS_INTEGER,
  cust_last       IN OUT VARCHAR2,
  ware_street_1       OUT VARCHAR2,
  ware_street_2       OUT VARCHAR2,
  ware_city           OUT VARCHAR2,
  ware_state          OUT VARCHAR2,
  ware_zip            OUT VARCHAR2,
```

```
  dist_street_1       OUT VARCHAR2,
  dist_street_2       OUT VARCHAR2,
  dist_city           OUT VARCHAR2,
  dist_state          OUT VARCHAR2,
  dist_zip            OUT VARCHAR2,
  cust_first          OUT VARCHAR2,
  cust_middle         OUT VARCHAR2,
  cust_street_1       OUT VARCHAR2,
  cust_street_2       OUT VARCHAR2,
  cust_city           OUT VARCHAR2,
  cust_state          OUT VARCHAR2,
  cust_zip            OUT VARCHAR2,
  cust_phone          OUT VARCHAR2,
  cust_since          OUT VARCHAR2,
  cust_credit      IN OUT VARCHAR2,
  cust_credit_lim     OUT NUMBER,
  cust_discount       OUT NUMBER,
  cust_balance     IN OUT NUMBER,
  cust_data           OUT VARCHAR2,
  hist_date           OUT VARCHAR2,
  retry            IN OUT INTEGER,
            cur_date                          IN            DATE
)
IS
  TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
  cust_rowid            ROWID;
  ware_rowid            ROWID;
  dist_ytd              NUMBER(12);
  dist_name             VARCHAR2(11);
  ware_ytd              NUMBER(12);
  ware_name             VARCHAR2(11);
  history_date          DATE;
  c_num                 BINARY_INTEGER;
  row_id                rowidarray;
            cust_payments                                 PLS_INTEGER;
            cust_ytd                                      NUMBER(12);
            cust_data_temp                      VARCHAR2(500);
          not_serializable      EXCEPTION;
          PRAGMA EXCEPTION_INIT(not_serializable,-8177);
          deadlock              EXCEPTION;
 PRAGMA EXCEPTION_INIT(deadlock,-60);
          snapshot_too_old      EXCEPTION;
 PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  CURSOR c_cur IS
    SELECT rowid
    FROM customer
    WHERE c_d_id = cust_d_id AND c_w_id = cust_w_id AND c_last = cust_last
    ORDER BY c_w_id, c_d_id, c_last, c_first;
BEGIN
  LOOP BEGIN
                                c_num := 0;
                                FOR c_id_rec IN c_cur LOOP
                                        c_num := c_num + 1;
                                        row_id(c_num) := c_id_rec.rowid;
                        END LOOP;
                        cust_rowid := row_id ((c_num + 1) / 2);

    SELECT c_id, c_first, c_middle, c_last, c_street_1, c_street_2,
        c_city, c_state, c_zip, c_phone,
        to_char (c_since, 'DD-MM-YYYY'), c_credit, c_credit_lim,
        c_discount, c_balance - hist_amount, c_payment_cnt,
                        c_ytd_payment + hist_amount, decode(c_credit, 'BC',
c_data, ' ')
      INTO cust_id, cust_first, cust_middle, cust_last,
        cust_street_1, cust_street_2, cust_city, cust_state,
        cust_zip, cust_phone, cust_since, cust_credit,
        cust_credit_lim, cust_discount, cust_balance ,
                                cust_payments, cust_ytd, cust_data_temp
        FROM customer
                                WHERE rowid = cust_rowid;
                        cust_payments := cust_payments + 1;

      IF cust_credit = 'BC' THEN
                                cust_data_temp := substr ((to_char (cust_id) || '
' ||
                        to_char (cust_d_id) || ' ' ||
                        to_char (cust_w_id) || ' ' ||
                        to_char (dist_id) || ' ' ||
                        to_char (ware_id) || ' ' ||
                        to_char (hist_amount/100, '9999.99') || ' | ')
                       || cust_data_temp, 1, 500);
        UPDATE customer
          SET c_balance = cust_balance,
            c_ytd_payment = cust_ytd,
            c_payment_cnt = cust_payments,
                                        c_data = cust_data_temp
          WHERE rowid = cust_rowid;
        cust_data := substr(cust_data_temp, 1, 200);
      ELSE
        UPDATE customer
          SET c_balance = cust_balance,
            c_ytd_payment = cust_ytd,
            c_payment_cnt = cust_payments
          WHERE rowid = cust_rowid;

        cust_data := cust_data_temp;
      END IF;

      SELECT district.rowid, d_name, d_street_1, d_street_2, d_city,
                                d_state, d_zip, d_ytd + hist_amount,
            warehouse.rowid, w_name, w_street_1, w_street_2, w_city,
                                w_state, w_zip, w_ytd + hist_amount
        INTO cust_rowid, dist_name, dist_street_1, dist_street_2,
                                        dist_city, dist_state, dist_zip,
dist_ytd,
            ware_rowid, ware_name, ware_street_1, ware_street_2,
                                        ware_city, ware_state, ware_zip,
ware_ytd
        FROM district, warehouse
        WHERE d_id = dist_id AND d_w_id = ware_id AND w_id = ware_id;
      UPDATE district
        SET d_ytd = dist_ytd WHERE rowid = cust_rowid;

      UPDATE warehouse
        SET w_ytd = ware_ytd WHERE rowid = ware_rowid;

                        history_date := cur_date;

      INSERT INTO history VALUES
        (cust_id, cust_d_id, cust_w_id, dist_id, ware_id, history_date,
         hist_amount, ware_name || '    ' || dist_name);
```

```
      COMMIT;
      hist_date := to_char (history_date, 'DD-MM-YYYY.HH24:MI:SS');
      EXIT;

                        EXCEPTION
                                WHEN not_serializable OR deadlock OR
snapshot_too_old THEN
                                        ROLLBACK;
                                        retry := retry + 1;
                        END;

    END LOOP;
  END;
END;
/
show errors;
```

sto.sql

```
rem
rem
================================================================
====+
rem       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
rem                 OPEN SYSTEMS PERFORMANCE GROUP                  |
rem                     All Rights Reserved                         |
rem
================================================================
====+
rem  FILENAME
rem      sto.sql
rem  DESCRIPTION
rem      SQL script to create a stored procedure for stock level
rem      transactions.
rem
================================================================
====
rem

CREATE OR REPLACE PACKAGE stocklevel
IS
  PROCEDURE getstocklevel
  (
    ware_id      INTEGER,
    dist_id      INTEGER,
    threshold    INTEGER,
    low_stock OUT INTEGER
  );
END;
/

CREATE OR REPLACE PACKAGE BODY stocklevel
IS
  PROCEDURE getstocklevel
  (
    ware_id      INTEGER,
    dist_id      INTEGER,
    threshold    INTEGER,
    low_stock OUT INTEGER
  )
  IS
    not_serializable        EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock                EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old        EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  BEGIN

    LOOP BEGIN

      SELECT count (DISTINCT s_i_id)
        INTO low_stock
        FROM order_line, stock, district
        WHERE d_id = dist_id AND d_w_id = ware_id AND
            d_id = ol_d_id AND d_w_id = ol_w_id AND
            ol_i_id = s_i_id AND ol_w_id = s_w_id AND
            s_quantity < threshold AND
            ol_o_id BETWEEN (d_next_o_id - 20) AND (d_next_o_id - 1);
      COMMIT;
      EXIT;

      EXCEPTION
        WHEN not_serializable OR deadlock OR snapshot_too_old THEN
          ROLLBACK;
      END;
    END LOOP;
  END;
END;
/

quit;
```

Appendix B: Database Parameters

```
NAME                           TYPE    VALUE
------------------------------------ ------- ------------------------------
always_anti_join                 string  NESTED_LOOPS
async_read                       boolean TRUE
async_write                      boolean TRUE
audit_file_dest                  string  ?/rdbms/audit
audit_trail                      string  NONE
background_core_dump               string  full
background_dump_dest               string  ?/rdbms/log
bitmap_merge_area_size             integer 1048576
blank_trimming                   boolean FALSE
cache_size_threshold               integer 81000
ccf_io_size                      integer 524288
checkpoint_process               boolean FALSE
cleanup_rollback_entries           integer 20
close_cached_open_cursors          boolean FALSE
commit_point_strength              integer 1
compatible                       string  7.3.1.0.0
compatible_no_recovery             string
control_files                    string  ?/dbs/cntrl@.dbf
core_dump_dest                     string  ?/dbs
cpu_count                        integer 4
```

```
create_bitmap_area_size            integer 8388608
cursor_space_for_time              boolean TRUE
db_block_buffers                   integer 870000
db_block_checkpoint_batch          integer 64
db_block_checksum                  boolean FALSE
db_block_lru_extended_statistics   integer 0
db_block_lru_latches               integer 4
db_block_lru_statistics            boolean FALSE
db_block_size                      integer 2048
db_domain                          string  WORLD
db_file_multiblock_read_count      integer 32
db_file_simultaneous_writes        integer 8
db_file_standby_name_convert       string
db_files                           integer 70
db_name                            string  tpcc
dblink_encrypt_login               boolean FALSE
delayed_logging_block_cleanouts    boolean TRUE
discrete_transactions_enabled      boolean FALSE
distributed_lock_timeout           integer 60
distributed_recovery_connection_hol integer 200
distributed_transactions           integer 0
dml_locks                          integer 0
enqueue_resources                  integer 1020
event                              string
fixed_date                         string
gc_db_locks                        integer 100
gc_files_to_locks                  string
gc_freelist_groups                 integer 50
gc_lck_procs                       integer 1
gc_releasable_locks                integer 100
gc_rollback_locks                  integer 20
gc_rollback_segments               integer 220
gc_save_rollback_locks             integer 20
gc_segments                        integer 10
gc_tablespaces                     integer 5
global_names                       boolean FALSE
hash_area_size                     integer 50000
hash_join_enabled                  boolean TRUE
hash_multiblock_io_count           integer 8
ifile                              file
instance_number                    integer 0
job_queue_interval                 integer 60
job_queue_keep_connections         boolean FALSE
job_queue_processes                integer 0
lgwr_use_async_io                  boolean FALSE
license_max_sessions               integer 0
license_max_users                  integer 0
license_sessions_warning           integer 0
log_archive_buffer_size            integer 32
log_archive_buffers                integer 4
log_archive_dest                   string  ?/dbs/arch
log_archive_format                 string  %t_%s.dbf
log_archive_start                  boolean FALSE
log_block_checksum                 boolean FALSE
log_buffer                         integer 1048576
log_checkpoint_interval            integer 1500000000
log_checkpoint_timeout             integer 0
log_checkpoints_to_alert           boolean TRUE
log_file_standby_name_convert      string
log_files                          integer 255
log_simultaneous_copies            integer 8
log_small_entry_max_size           integer 800
max_commit_propagation_delay       integer 90000
max_dump_file_size                 integer 3000
max_enabled_roles                  integer 20
max_rollback_segments              integer 200
max_transaction_branches           integer 8
mts_dispatchers                    string
mts_listener_address               string  (address=(protocol=ipc)(key=%s
mts_max_dispatchers                integer 0
mts_max_servers                    integer 0
mts_multiple_listeners             boolean FALSE
mts_servers                        integer 0
mts_service                        string  tpcc
nail_shadow                        boolean TRUE
nls_currency                       string
nls_date_format                    string
nls_date_language                  string
nls_iso_currency                   string
nls_language                       string  AMERICAN
nls_numeric_characters             string
nls_sort                           string
nls_territory                      string  AMERICA
open_cursors                       integer 300
open_links                         integer 4
optimizer_mode                     string  CHOOSE
optimizer_percent_parallel         integer 0
oracle_trace_collection_name       string  oracle7
oracle_trace_collection_path       string  ?/rdbms/log
oracle_trace_collection_size       integer 5242880
oracle_trace_enable                boolean FALSE
oracle_trace_facility_name         string  oracle7
oracle_trace_facility_path         string  ?/rdbms/admin
os_authent_prefix                  string  ops$
os_roles                           boolean FALSE
parallel_default_max_instances     integer 0
parallel_max_servers               integer 30
parallel_min_percent               integer 0
parallel_min_servers               integer 0
parallel_server_idle_time          integer 5
partition_view_enabled             boolean FALSE
post_wait_device                   string  /dev/pw
pre_page_sga                       boolean FALSE
processes                          integer 300
recovery_parallelism               integer 30
reduce_alarm                       boolean TRUE
remote_dependencies_mode           string  timestamp
remote_login_passwordfile          string  NONE
remote_os_authent                  boolean FALSE
remote_os_roles                    boolean FALSE
resource_limit                     boolean FALSE
rollback_segments                  string  t1, t2, t3, t4, t5, t6, t7, t8
row_cache_cursors                  integer 10
row_locking                        string  always
sequence_cache_entries             integer 10
sequence_cache_hash_buckets        integer 7
serializable                       boolean FALSE
session_cached_cursors             integer 0
sessions                           integer 600
shadow_core_dump                   string  full
```

```
shared_pool_reserved_min_alloc      integer 5000
shared_pool_reserved_size           integer 0
shared_pool_size                    integer 14000000
snapshot_refresh_interval           integer 60
snapshot_refresh_keep_connections   boolean FALSE
snapshot_refresh_processes          integer 0
sort_area_retained_size             integer 65536
sort_area_size                      integer 65536
sort_direct_writes                  string  AUTO
sort_read_fac                       integer 5
sort_spacemap_size                  integer 512
sort_write_buffer_size              integer 32768
sort_write_buffers                  integer 2
spin_count                          integer 2000
sql92_security                      boolean FALSE
sql_trace                           boolean FALSE
temporary_table_locks               integer 400
text_enable                         boolean FALSE
thread                              integer 0
timed_statistics                    boolean FALSE
transactions                        integer 400
transactions_per_rollback_segment   integer 1
use_async_io                        boolean FALSE
use_internode_pqo                   boolean FALSE
use_ism                             boolean TRUE
use_list_io_extension               boolean FALSE
use_pdo_device                      string
use_post_wait_driver                boolean FALSE
use_post_wait_extension             boolean FALSE
use_readv                           boolean FALSE
use_shared_memory_nailing           boolean TRUE
user_dump_dest                      string  ?/rdbms/log
utl_file_dir                        string
```

Solaris Client Parameters

```
*ident  "@(#)system  1.15  92/11/14 SMI" /* SVR4 1.5 */
*
* SYSTEM SPECIFICATION FILE
*

* moddir:
*
*       Set the search path for modules.  This has a format similar to the
*       csh path variable. If the module isn't found in the first directory
*       it tries the second and so on. The default is /kernel /usr/kernel
*
*       Example:
*               moddir: /kernel /usr/kernel /other/modules



* root device and root filesystem configuration:
*
*       The following may be used to override the defaults provided by
*       the boot program:
*
*       rootfs:         Set the filesystem type of the root.
*
*       rootdev:    Set the root device.  This should be a fully
*                   expanded physical pathname.  The default is
 the
*                   physical pathname of the device where the boot
*                   program resides.  The physical pathname is
*                   highly platform and configuration dependent.
*
*       Example:
*               rootfs:ufs
*               rootdev:/sbus@1,f8000000/esp@0,800000/sd@3,0:a
*
*       (Swap device configuration should be specified in /etc/vfstab.)

* exclude:
*
*       Modules appearing in the moddir path which are NOT to be loaded,
*       even if referenced. Note that `exclude' accepts either a module name,
*       or a filename which includes the directory.
*
*       Examples:
*               exclude: win
*               exclude: sys/shmsys

* forceload:
*
*       Cause these modules to be loaded at boot time, (just before mounting
*       the root filesystem) rather than at first reference. Note that
*       forceload expects a filename which includes the directory. Also
*       note that loading a module does not necessarily imply that it will
*       be installed.
*
*       Example:
*               forceload: drv/foo

* set:
*
*       Set an integer variable in the kernel or a module to a new value.
*       This facility should be used with caution.  See system(4).
*
*       Examples:
*
*       To set variables in 'unix':
*
*               set nautopush=32
*               set maxusers=40
*
*       To set a variable named 'debug' in the module named 'test_module'
*
*               set test_module:debug = 0x13

set maxusers=10
set max_nprocs=2280
set maxuprc=2270
set pt_cnt=1050
set shmsys:shminfo_shmmax=2684354560
set msgsys:msginfo_msgmax=8192
set msgsys:msginfo_msgmni=1300
set msgsys:msginfo_msgtql=1300
set msgsys:msginfo_msgseg=19200
set msgsys:msginfo_msgssz=128
*set msgsys:msginfo_msgmap=2048
```

```
set semsys:seminfo_semmap=20
set semsys:seminfo_semmsl=1300
set semsys:seminfo_semmns=1300
set semsys:seminfo_semmnu=1300
set semsys:seminfo_semume=1300
set semsys:seminfo_semmni=1300
set i86mmu_hwptepages=2200
set i86mmu_maps_per_page=4
set lotsfree=0x100
#set fastscan=0x800
#set handspreadpages=0x800
# set IGNORE_KERNEL_PREEMPTION=1
# set only_intr_kpreempt=1
```

Solaris Server Parameters

```
*ident	"@(#)system	1.15	92/11/14 SMI" /* SVR4 1.5 */
*
* SYSTEM SPECIFICATION FILE
*

* moddir:
*
*		Set the search path for modules.  This has a format similar to the
*		csh path variable. If the module isn't found in the first directory
*		it tries the second and so on. The default is /kernel /usr/kernel
*
*		Example:
*			moddir: /kernel /usr/kernel /other/modules

* root device and root filesystem configuration:
*
*		The following may be used to override the defaults provided by
*		the boot program:
*
*		rootfs:		Set the filesystem type of the root.
*
*		rootdev:	Set the root device.  This should be a fully
*				expanded physical pathname.  The default is the
*				physical pathname of the device where the boot
*				program resides.  The physical pathname is
*				highly platform and configuration dependent.
*
*		Example:
*			rootfs:ufs
*			rootdev:/sbus@1,f8000000/esp@0,800000/sd@3,0:a
*
*		(Swap device configuration should be specified in /etc/vfstab.)

* exclude:
*
*		Modules appearing in the moddir path which are NOT to be loaded,
*		even if referenced. Note that `exclude' accepts either a module name,
*		or a filename which includes the directory.
*
*		Examples:
*			exclude: win
*			exclude: sys/shmsys

* forceload:
*
*		Cause these modules to be loaded at boot time, (just before mounting
*		the root filesystem) rather than at first reference. Note that
*		forceload expects a filename which includes the directory. Also
*		note that loading a module does not necessarily imply that it will
*		be installed.
*
*		Example:
*			forceload: drv/foo

* set:
*
*		Set an integer variable in the kernel or a module to a new value.
*		This facility should be used with caution.  See system(4).
*
*		Examples:
*
*		To set variables in 'unix':
*
*			set nautopush=32
*			set maxusers=40
*
*		To set a variable named 'debug' in the module named 'test_module'
*
*			set test_module:debug = 0x13


set shmsys:shminfo_shmmni=200
set semsys:seminfo_semmnu=3032
set semsys:seminfo_semmap=6064
set rlim_fd_cur = 200
set tune_t_fsflushr = 36000
set maxusers=10
set max_nprocs=400
set maxuprc=300
set spt_max=99
set i86mmu_hwptepages=1024
set i86mmu_maps_per_page=1
set strategy:dmult_maxcnt=30
set only_intr_kpreempt=1
set rechoose_interval=0x200
set pcplusmp:apic_intr_distribute=0
set pcplusmp:apic_int0_bind_cpu2= 0x4000
set pcplusmp:apic_coarse_hrtime=1
set enable_rdwrmsr=1
set dpt:gsc_options=1
set shmsys: shminfo_shmmax = 3500000000
set shmsys: shminfo_shmseg = 32
set semsys: seminfo_semmns = 6064
set semsys: seminfo_semmni = 70
set lotsfree = 1000
```

Tuxedo Configuration Parameters

```
*RESOURCES
IPCKEY		40001
MASTER		client1
PERM		0666
MODEL		SHM
LDBAL		Y
```

```
MAXACCESSERS    1200
MAXSERVERS      35
MAXSERVICES     35
SCANUNIT30
SANITYSCAN      5
BLOCKTIME       10
BBLQUERY        60

*MACHINES
client1     LMID=client1
            TUXCONFIG="/dbbench/tuxedo/tuxconfig.client1"
            ROOTDIR="/dbbench/tuxedo/tuxedo"
            APPDIR="/dbbench/tuxedo"
            ULOGPFX="/dbbench/tuxedo/ULOGclient1"

*GROUPS
group1      LMID=client1        GRPNO=1
group2      LMID=client1        GRPNO=2
group3      LMID=client1        GRPNO=3
group4      LMID=client1        GRPNO=4
group5      LMID=client1        GRPNO=5

*SERVERS
tpcc_srv_newo       SRVGRP=group1       SRVID=101 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=102 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=103 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=104 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=105 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=106 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=107 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=108 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=109 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=110 RQADDR=newoq1
 REPLYQ=Y
tpcc_srv_newo       SRVGRP=group1       SRVID=111 RQADDR=newoq1
 REPLYQ=Y

tpcc_srv_paym       SRVGRP=group2       SRVID=201 RQADDR=paymq1
 REPLYQ=Y
tpcc_srv_paym       SRVGRP=group2       SRVID=202 RQADDR=paymq1
 REPLYQ=Y
tpcc_srv_paym       SRVGRP=group2       SRVID=203 RQADDR=paymq1
 REPLYQ=Y
tpcc_srv_paym       SRVGRP=group2       SRVID=204 RQADDR=paymq1
 REPLYQ=Y

tpcc_srv_ords       SRVGRP=group3       SRVID=301 RQADDR=ordsq1
 REPLYQ=Y

tpcc_srv_stock      SRVGRP=group4       SRVID=401 RQADDR=stockq1
 REPLYQ=Y
tpcc_srv_stock      SRVGRP=group4       SRVID=402 RQADDR=stockq1
 REPLYQ=Y
tpcc_srv_stock      SRVGRP=group4       SRVID=403 RQADDR=stockq1
 REPLYQ=Y
tpcc_srv_stock      SRVGRP=group4       SRVID=404 RQADDR=stockq1
 REPLYQ=Y
tpcc_srv_stock      SRVGRP=group4       SRVID=405 RQADDR=stockq1
 REPLYQ=Y
tpcc_srv_del        SRVGRP=group5 SRVID=501 CLOPT="-A -- 1"
 RQADDR=delq1 REPLYQ=Y
tpcc_srv_del        SRVGRP=group5 SRVID=502 CLOPT="-A -- 2"
 RQADDR=delq1 REPLYQ=Y

*SERVICES
DEL
NEWO
ORDS
PAYM
STOCK
```

Appendix C: Database Creation Scripts

alter.sh

```
#
# $Header: alter.sh 7030100.1 95/07/17 14:36:20 plai Generic<base> $ Copyr (c)
 1995 Oracle
#

#==============================================================
=====+
#       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
#                  OPEN SYSTEMS PERFORMANCE GROUP                |
#                       All Rights Reserved                      |
#==============================================================
=====+
# FILENAME
#     alter.sh
# DESCRIPTION
#     Change next extent size for TPC-C tables and indexes.
# USAGE
#     alter.sh
#==============================================================
=====*/

sqlplus tpcc/tpcc <<!
  alter table history     storage (next 750M);
  alter cluster ccluster  storage (next  50M);
  alter cluster scluster  storage (next  50M);
  alter table orders      storage (next 750M);
  alter table order_line  storage (next 800M);
  alter table new_order   storage (next 350M);
  alter index iorders     storage (next 800M);
  alter index iorders2    storage (next 800M);
  alter index inew_order  storage (next 325M);
  alter index iorder_line storage (next 850M);
```

```
  alter index istock      storage (next  50M);
  alter index icustomer   storage (next  50M);
  alter index icustomer2  storage (next  50M);
  quit;
!
```

benchdb.sh

```
#
# $Header: benchdb.sh 7030100.1 96/05/02 19:05:22 plai Generic<base> $ Copyr
(c) 1995 Oracle
#

#
#============================================================
=====+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
#                  OPEN SYSTEMS PERFORMANCE GROUP                 |
#                       All Rights Reserved                       |
#============================================================
=====+
# FILENAME
#      benchdb.sh
# DESCRIPTION
#      Usage: benchdb.sh [options]
#             -n          do not create new tpcc database
#             -c          do not run catalog scripts
#============================================================
=====
#



BENCH_HOME=$ORACLE_HOME/bench/tpc
TPCC_ADMIN=admin

while [ "$#" != "0" ]
do
  case $1 in
     -n) shift
        NO_CREATE="y"
        ;;
     -c) shift
        NO_CAT="y"
        ;;
      *) echo "Bag arg: $1"
        exit 1;
        ;;
  esac
done

#
# Create database if NO_CREATE unset
#

if [ "$NO_CREATE" = "" ]
then
sqldba <<!
              set echo on
              connect internal
              startup  pfile=$TPCC_ADMIN/p_create.ora  nomount
              create database tpcc controlfile reuse maxdatafiles 70
                 datafile '/dev/orac/sys1' size  299M reuse
                 logfile  '/dev/orac/log1' size 2023M reuse,
                          '/dev/orac/log2' size 2023M reuse;
              exit
!

#
# Create more rollback segments
#

sqldba <<!
  connect system/manager
  create rollback segment s1  storage (initial 200k minextents 2 next 200k);
  create rollback segment s2  storage (initial 200k minextents 2 next 200k);
  create rollback segment s3  storage (initial 200k minextents 2 next 200k);
  create rollback segment s4  storage (initial 200k minextents 2 next 200k);
  create rollback segment s5  storage (initial 200k minextents 2 next 200k);
  create rollback segment s6  storage (initial 200k minextents 2 next 200k);
  create rollback segment s7  storage (initial 200k minextents 2 next 200k);
  create rollback segment s8  storage (initial 200k minextents 2 next 200k);
  create rollback segment s9  storage (initial 200k minextents 2 next 200k);
  create rollback segment s10 storage (initial 200k minextents 2 next 200k);
  create rollback segment s11 storage (initial 200k minextents 2 next 200k);
  create rollback segment s12 storage (initial 200k minextents 2 next 200k);
  create rollback segment s13 storage (initial 200k minextents 2 next 200k);
  create rollback segment s14 storage (initial 200k minextents 2 next 200k);
  create rollback segment s15 storage (initial 200k minextents 2 next 200k);
  create rollback segment s16 storage (initial 200k minextents 2 next 200k);
  create rollback segment s17 storage (initial 200k minextents 2 next 200k);
  create rollback segment s18 storage (initial 200k minextents 2 next 200k);
  create rollback segment s19 storage (initial 200k minextents 2 next 200k);
  create rollback segment s20 storage (initial 200k minextents 2 next 200k);
  create rollback segment s21 storage (initial 200k minextents 2 next 200k);
  create rollback segment s22 storage (initial 200k minextents 2 next 200k);
  create rollback segment s23 storage (initial 200k minextents 2 next 200k);
  create rollback segment s24 storage (initial 200k minextents 2 next 200k);
  create rollback segment s25 storage (initial 200k minextents 2 next 200k);
  create rollback segment s26 storage (initial 200k minextents 2 next 200k);
  create rollback segment s27 storage (initial 200k minextents 2 next 200k);
  create rollback segment s28 storage (initial 200k minextents 2 next 200k);
  create rollback segment s29 storage (initial 200k minextents 2 next 200k);
  create rollback segment s30 storage (initial 200k minextents 2 next 200k);
  disconnect;
  connect internal;
  shutdown;
  exit;
!
fi

#
#  Startup database with params file that includes new rollback segments
#

sqldba <<!
      connect internal
      startup pfile=$TPCC_ADMIN/p_build.ora;
      connect system/manager
      create tablespace roll datafile
        '/dev/orac/roll1'   size  399M reuse;
```

```
    create tablespace hist datafile
      '/dev/orac/hist1'   size 1899M reuse;
    create tablespace ware datafile
      '/dev/orac/ware1'   size   19M reuse;
    create tablespace cust datafile
      '/dev/orac/cust0'   size  100K reuse;
    create tablespace items datafile
      '/dev/orac/item1'   size   19M reuse;
    create tablespace ord datafile
      '/dev/orac/ord1'    size 1599M reuse;
    create tablespace nord datafile
      '/dev/orac/nord1'   size  499M reuse;
    create tablespace ordl datafile
      '/dev/orac/ordl0'   size  100K reuse;
    create tablespace stocks datafile
      '/dev/orac/stk0'    size  100K reuse;
    create tablespace icust1 datafile
      '/dev/orac/icust1'  size  999M reuse;
    create tablespace icust2 datafile
      '/dev/orac/icust2'  size 1599M reuse;
    create tablespace istk datafile
      '/dev/orac/istk1'   size 1699M reuse;
    create tablespace iord1 datafile
      '/dev/orac/iord1'   size  1399M reuse;
    create tablespace iord2 datafile
      '/dev/orac/iord2'   size 1699M reuse;
    create tablespace inord datafile
      '/dev/orac/inord1'  size  499M reuse;
    create tablespace iordl datafile
      '/dev/orac/iordl0'  size  100K reuse;
    create tablespace temp datafile
      '/dev/orac/temp0'   size  100K reuse;
    exit;
!

#
#  Add datafiles to tablespaces in parallel
#

addfile.sh cust /dev/orac/cust1 1699M &
addfile.sh cust /dev/orac/cust2 1699M &
addfile.sh cust /dev/orac/cust3 1699M &
addfile.sh cust /dev/orac/cust4 1699M &
addfile.sh cust /dev/orac/cust5 1699M &
addfile.sh cust /dev/orac/cust6 1699M &
addfile.sh cust /dev/orac/cust7 1699M &
addfile.sh cust /dev/orac/cust8 1699M &
addfile.sh cust /dev/orac/cust9 1699M &
addfile.sh cust /dev/orac/cust10 1699M &
addfile.sh cust /dev/orac/cust11 1699M &
addfile.sh cust /dev/orac/cust12 1699M &

addfile.sh ordl /dev/orac/ordl1 1999M &
addfile.sh ordl /dev/orac/ordl2 1999M &
addfile.sh ordl /dev/orac/ordl3 1999M &
addfile.sh ordl /dev/orac/ordl4 1999M &
addfile.sh ordl /dev/orac/ordl5 1999M &
addfile.sh ordl /dev/orac/ordl6 1999M &
addfile.sh ordl /dev/orac/ordl7 1999M &
addfile.sh ordl /dev/orac/ordl8 1999M &
addfile.sh ordl /dev/orac/ordl9 1999M &
addfile.sh ordl /dev/orac/ordl10 1999M &
addfile.sh ordl /dev/orac/ordl11 1999M &
addfile.sh ordl /dev/orac/ordl12 1999M &

addfile.sh stocks /dev/orac/stk1 1699M &
addfile.sh stocks /dev/orac/stk2 1699M &
addfile.sh stocks /dev/orac/stk3 1699M &
addfile.sh stocks /dev/orac/stk4 1699M &
addfile.sh stocks /dev/orac/stk5 1699M &
addfile.sh stocks /dev/orac/stk6 1699M &
addfile.sh stocks /dev/orac/stk7 1699M &
addfile.sh stocks /dev/orac/stk8 1699M &
addfile.sh stocks /dev/orac/stk9 1699M &
addfile.sh stocks /dev/orac/stk10 1699M &
addfile.sh stocks /dev/orac/stk11 1699M &
addfile.sh stocks /dev/orac/stk12 1699M &
addfile.sh stocks /dev/orac/stk13 1699M &
addfile.sh stocks /dev/orac/stk14 1699M &
addfile.sh stocks /dev/orac/stk15 1699M &
addfile.sh stocks /dev/orac/stk16 1699M &

addfile.sh iordl  /dev/orac/iordl1 1999M &
addfile.sh iordl  /dev/orac/iordl2 1999M &
addfile.sh iordl  /dev/orac/iordl3 1999M &
addfile.sh iordl  /dev/orac/iordl4 1999M &

addfile.sh temp   /dev/orac/temp1 1999M &
addfile.sh temp   /dev/orac/temp2 1999M &
addfile.sh temp   /dev/orac/temp3 1999M &
addfile.sh temp   /dev/orac/temp4 1999M &

wait

#
# run catalog if NO_CAT unset
#

if [ "$NO_CAT" = "" ]
then
sqldba <<!
      set echo off;
      connect sys/change_on_install;
      @?/rdbms/admin/catalog;
      @?/rdbms/admin/catproc;
      @?/rdbms/admin/catparr;
      exit;
!
fi
```

benchsetup.sh

```
#
# $Header: benchsetup.sh 7030100.2 96/05/16 17:59:17 plai Generic<base> $
Copyr (c) 1995 Oracle
#

#
#=============================================================
=====+
```

```
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA       |
#                  OPEN SYSTEMS PERFORMANCE GROUP                 |
#                       All Rights Reserved                       |
#=================================================================
=====+
# FILENAME
#      benchsetup.sh
# DESCRIPTION
#      Usage: benchsetup.sh [options]
#           -mu <multiplier>     (# of warehouses)
#           -nd                  do not run benchdb.sh
#           -nt                  do not create tpcc tables
#           -nx                  do not create index for tpcc tables
#=================================================================
=====
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
BENCH_GEN=$ORACLE_HOME/bench/gen
GEN_SQL=$BENCH_GEN/sql
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_SQL=$BENCH_HOME/tpcc/sql
TPCC_STORE=$BENCH_HOME/tpcc/stored_proc
TPCC_SCRIPTS=$BENCH_HOME/tpcc/scripts
TPCC_UTILS=$TPCC_SCRIPTS/utils
AUDIT_SQL=$BENCH_HOME/tpcc/audit/sql
BUILD_SQL=sql
OUTDIR=outdir
MULT=564

PATH=${PATH}:${TPCC_SOURCE}:${TPCC_UTILS}
export PATH

if echo "\c" | grep c >/dev/null 2>&1; then
   N='-n'
else
   C='\c'
fi
export N C

while [ "$#" != "0" ]
do
  case $1 in
     -mu) shift
        if [ "$1" != "" ]
        then
          MULT=$1
          shift
        fi
        ;;
     -nd) shift
        NO_DB="y"
        ;;
     -nt) shift
        NO_TAB="y"
        ;;
     -nx) shift
        NO_IND="y"
         ;;
      *) echo "Bag arg: $1"
        exit 1;
         ;;
  esac
done

if [ "$MULT" = "" ]
then
  echo $N "Database multiplier (# of warehouses)? [1]" $C
  read MULT
  if [ "$MULT" = "" ]
  then
     MULT=1
  fi
fi

if [ ! -d $OUTDIR ]
then
  mkdir $OUTDIR
fi

#
# Create database.
#

if [ "$NO_DB" = "" ]
then
  benchdb.sh
fi

switchlog.sh

#
#  Create tables.
#

if [ "$NO_TAB" = "" ]
then
  sqlplus system/manager @$BUILD_SQL/tpcc_tab
  sqlplus system/manager @$BUILD_SQL/tpcc_rol
fi

#
#  Load history, new-order, order, order-line tables
#

pload.sh > ${OUTDIR}/pload.out 2>&1

switchlog.sh

#
#  Create customer and stock tables.
#

if [ "$NO_TAB" = "" ]
then
  sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_tab2 > ${OUTDIR}/tab2.out 2>&1 &
  sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_tab3 > ${OUTDIR}/tab3.out 2>&1 &
fi

wait
```

```
switchlog.sh

#
#  Load warehouse, district, item tables
#

tpccload -M $MULT -w
tpccload -M $MULT -d
tpccload -M $MULT -i

#
#  Load customer table
#

I=1
SW=1
EW=12
INC=12
while [ $I -le 47 ]
do
  tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${I}.out 2>&1 &
  I=`expr $I + 1`
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

#
#  Load stock table
#

I=1
SI=1
EI=2500
INC=2500
while [ $I -le 40 ]
do
  tpccload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out 2>&1 &
  I=`expr $I + 1`
  SI=`expr $SI + $INC`
  EI=`expr $EI + $INC`
done

wait

switchlog.sh

#
#  Create indexes
#

if [ "$NO_IND" = "" ]
then

sqlplus system/manager <<!
  alter user tpcc temporary tablespace temp;
  quit;
!

sqldba <<!
  connect internal
  alter tablespace temp
    default storage (initial 100M next 100M pctincrease 0);
  exit;
!

sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_ix1
sqlplus tpcc/tpcc @$BUILD_SQL/tpcc_ix2

sqldba <<!
  connect internal
  alter tablespace temp
    default storage (initial 20K next 20K pctincrease 50);
  exit;
!

fi

#
#  Analyze tables and indexes
#

sqlplus tpcc/tpcc @$TPCC_SQL/tpcc_ana

#
#  Create table for processing benchmark results
#

sqlplus sys/change_on_install @$GEN_SQL/orst_cre
sqlplus sys/change_on_install @$TPCC_SQL/c_stat
sqlplus sys/change_on_install @$GEN_SQL/pst_c

#
#  Create stored procedures
#

sqlplus tpcc/tpcc @$TPCC_STORE/new
sqlplus tpcc/tpcc @$TPCC_STORE/pay
sqlplus tpcc/tpcc @$TPCC_STORE/ord
sqlplus tpcc/tpcc @$TPCC_STORE/sto

#
#  Get some statistics
#

$TPCC_SCRIPTS/utils/ext_all.sh > ${OUTDIR}/ext_all.out 2>&1

$TPCC_SCRIPTS/utils/space_init.sh
$TPCC_SCRIPTS/utils/space_get.sh 6650 564
$TPCC_SCRIPTS/utils/space_rpt.sh ${OUTDIR}/space.rpt

sqlplus system/manager <<!
  alter user tpcc temporary tablespace system;
  quit;
!

sqlplus sys/change_on_install <<!
  grant execute on dbms_lock to public;
  grant execute on dbms_pipe to public;
  grant select on v_\$parameter to public;
  quit;
```

```
!

sqlplus tpcc/tpcc @$AUDIT_SQL/plsql_mon
sqlplus tpcc/tpcc @$AUDIT_SQL/cre_tab

sh alter.sh

dml.sh

sqldba <<!
  set echo off;
  connect sys/change_on_install;
  @?/rdbms/admin/catparr;
  exit;
!

#
#  Shutdown database
#

sqldba <<!
  connect internal;
  alter system switch logfile;
  alter system switch logfile;
  shutdown;
  exit;
!
```

dml.sh

```
#
# $Header: dml.sh 7030100.1 96/05/02 10:22:52 plai Generic<base> $ Copyr (c)
1995 Oracle
#

#============================================================
=====+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                  OPEN SYSTEMS PERFORMANCE GROUP                |
#                     All Rights Reserved                        |
#============================================================
=====+
# FILENAME
#     dml.sh
# DESCRIPTION
#     Disable table locks for TPC-C tables.
# USAGE
#     dml.sh
#============================================================
=====*/

sqlplus tpcc/tpcc <<!
  alter table warehouse  disable table lock;
  alter table district   disable table lock;
  alter table customer   disable table lock;
  alter table history    disable table lock;
  alter table item       disable table lock;
  alter table stock      disable table lock;
  alter table orders     disable table lock;
  alter table new_order  disable table lock;
  alter table order_line disable table lock;
  quit;
!
```

pload.sh

```
#
# $Header: pload.sh 7030100.1 96/05/02 19:06:06 plai Generic<base> $ Copyr (c)
 1995 Oracle
#

#
#============================================================
=====+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                  OPEN SYSTEMS PERFORMANCE GROUP                |
#                     All Rights Reserved                        |
#============================================================
=====+
# FILENAME
#      pload.sh
# DESCRIPTION
#      Usage: pload.sh [options]
#             -mu <multiplier>     (# of warehouses)
#============================================================
=====
#

BENCH_HOME=$ORACLE_HOME/bench/tpc
TPCC_SOURCE=$BENCH_HOME/tpcc/source
TPCC_LOADER=$BENCH_HOME/tpcc/loader
LDIR=data
OUTDIR=outdir
MULT=564

PATH=${PATH}:$TPCC_SOURCE
export PATH

if echo "\c" | grep c >/dev/null 2>&1; then
   N='-n'
else
   C='\c'
fi
export N C

while [ "$#" != "0" ]
do
  case $1 in
     -mu) shift
        if [ "$1" != "" ]
        then
          MULT=$1
          shift
        fi
        ;;
     -nd) shift
        NO_DB="y"
        ;;
     -nt) shift
        NO_TAB="y"
```

```
        ;;
    -nx) shift
        NO_IND="y"
        ;;
     *) echo "Bag arg: $1"
        exit 1;
        ;;
  esac
done

if [ "$MULT" = "" ]
then
  echo $N "Database multiplier (# of warehouses)? [1]" $C
  read MULT
  if [ "$MULT" = "" ]
  then
     MULT=1
  fi
fi

if [ ! -d $LDIR ]
then
  mkdir $LDIR
fi

if [ ! -d $OUTDIR ]
then
  mkdir $OUTDIR
fi

#
#  Load history table
#

I=1
while [ $I -le 12 ]
do
  mknod ${LDIR}/hist${I}.dat p
  I=`expr $I + 1`
done

I=1
SW=1
EW=47
INC=47
while [ $I -le 12 ]
do
  tpccload -M $MULT -h -g -b $SW -e $EW > ${LDIR}/hist${I}.dat 2> \
    ${OUTDIR}/hist${I}.out &
  I=`expr $I + 1`
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done

sleep 30

I=1
while [ $I -le 12 ]
do
  sqlldr tpcc/tpcc control=$TPCC_LOADER/hist.ctl \
        log=${OUTDIR}/hist${I}.log \
        bad=${OUTDIR}/hist${I}.bad data=${LDIR}/hist${I}.dat \
        discard=${OUTDIR}/hist${I}.dsc \
        file=/dev/orac/hist1 &
  I=`expr $I + 1`
done

wait

I=1
while [ $I -le 12 ]
do
  rm -f ${LDIR}/hist${I}.dat
  I=`expr $I + 1`
done

#
#  Load new-order table
#

mknod ${LDIR}/neword1.dat p
tpccload -M $MULT -n -g > ${LDIR}/neword1.dat 2> \
  ${OUTDIR}/neword1.out &
sleep 30
sqlldr tpcc/tpcc control=$TPCC_LOADER/neword.ctl \
      log=${OUTDIR}/neword1.log \
      bad=${OUTDIR}/neword1.bad data=${LDIR}/neword1.dat \
      discard=${OUTDIR}/neword1.dsc \
      file=/dev/orac/nord1 &
wait
rm -f ${LDIR}/neword1.dat

#
#  Load order and order-line table
#

I=1
while [ $I -le 24 ]
do
  mknod ${LDIR}/order${I}.dat p
  mknod ${LDIR}/ordline${I}.dat p
  I=`expr $I + 1`
done

I=1
SW=1
EW=24
INC=24
while [ $I -le 24 ]
do
  if [ $EW = 576 ]
  then
              EW=564
  fi
  tpccload -M $MULT -o ${LDIR}/ordline${I}.dat -g -b $SW -e $EW > \
    ${LDIR}/order${I}.dat 2> ${OUTDIR}/order${I}.out &
  I=`expr $I + 1`
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
done
```

```
sleep 30

I=1
while [ $I -le 24 ]
do
  J=`expr $I - 1`
  J=`expr $J / 2`
  J=`expr $J + 1`
  sqlldr tpcc/tpcc control=$TPCC_LOADER/order.ctl \
       log=${OUTDIR}/order${I}.log \
       bad=${OUTDIR}/order${I}.bad data=${LDIR}/order${I}.dat \
       discard=${OUTDIR}/order${I}.dsc \
       file=/dev/orac/ord1 &
  sqlldr tpcc/tpcc control=$TPCC_LOADER/ordline.ctl \
       log=${OUTDIR}/ordline${I}.log \
       bad=${OUTDIR}/ordline${I}.bad data=${LDIR}/ordline${I}.dat \
       discard=${OUTDIR}/ordline${I}.dsc \
       file=/dev/orac/ordl${J} &
  I=`expr $I + 1`
done

wait

I=1
while [ $I -le 24 ]
do
  rm -f ${LDIR}/order${I}.dat
  rm -f ${LDIR}/ordline${I}.dat
  I=`expr $I + 1`
done
```

SQL Scripts for DB Build

tpcc_ix1.sql

```
rem
rem
============================================================
====+
rem        Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
rem                  OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                       All Rights Reserved                       |
rem
============================================================
====+
rem  FILENAME
rem       tpcc_ix1.sql
rem  DESCRIPTION
rem       Create indexes for TPC-C database.
rem
============================================================
====
rem

drop index iwarehouse;
drop index idistrict;
drop index icustomer;
drop index icustomer2;
drop index istock;
drop index iitem;

set timing on
create unique index iwarehouse on warehouse(w_id)
       tablespace ware
       initrans 3
       storage (initial 200K next 20K pctincrease 0) pctfree 1;

create unique index idistrict on district(d_w_id, d_id)
       tablespace ware
       initrans 3
       storage (initial 2000K next 60K pctincrease 0) pctfree 1;

create unique index iitem on item(i_id)
       tablespace items
       storage (initial 2000K next 100K pctincrease 0) pctfree 1;

create unique index icustomer on customer(c_w_id, c_d_id, c_id)
       tablespace icust1
       initrans 3
       parallel 10
       storage (initial 8M next 8M pctincrease 0) pctfree 1;

create unique index icustomer2 on customer(c_last, c_w_id, c_d_id, c_first, c_id)
       tablespace icust2
       initrans 3
       parallel 10
       storage (initial 16M next 16M pctincrease 0) pctfree 1;

create unique index istock on stock(s_i_id, s_w_id)
       tablespace istk
       initrans 3
       parallel 10
       storage (initial 20M next 20M pctincrease 0) pctfree 1;

exit;
```

tpcc_ix2.sql

```
rem
rem
 ============================================================
====+
rem        Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
rem                  OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                       All Rights Reserved                       |
rem
============================================================
====+
rem  FILENAME
rem       tpcc_ix2.sql
rem  DESCRIPTION
rem       Create indexes for TPC-C database.
rem
============================================================
====
rem
```

```
drop index iorders;
drop index iorders2;
drop index inew_order;
drop index iorder_line;

set timing on

create unique index iorders on orders(o_w_id, o_d_id, o_id)
      tablespace iord1
      initrans 3
      parallel 10
      pctfree  1
      storage (initial 20M next 20M pctincrease 0
            freelist groups 13 freelists 24);

create unique index iorders2 on orders(o_w_id, o_d_id, o_c_id, o_id)
      tablespace iord2
      initrans 3
      parallel 10
      pctfree  25
      storage (initial 30M next 30M pctincrease 0
            freelist groups 13 freelists 24);

create unique index inew_order on new_order(no_w_id, no_d_id, no_o_id)
      tablespace inord
      initrans 4
      parallel 10
      pctfree  5
      storage (initial 7M next 7M pctincrease 0
            freelist groups 13 freelists 24);

create unique index iorder_line on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
      tablespace iordl
      initrans 4
      parallel 10
      pctfree  1
      storage (initial 50M next 50M pctincrease 0
            freelist groups 13 freelists 24);

exit;
```

tpcc_rol.sql

```
rem
rem
============================================================
====+
rem        Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
rem                  OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                       All Rights Reserved                       |
rem
============================================================
====+
rem  FILENAME
rem       tpcc_rol.sql
rem  DESCRIPTION
rem       Create rollback segments for TPCC database.
rem
============================================================
====
rem

  set timing on;

  host date;

  CREATE ROLLBACK SEGMENT t1
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t2
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t3
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t4
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t5
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t6
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t7
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t8
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t9
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t10
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t11
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t12
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t13
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t14
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t15
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t16
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t17
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
  CREATE ROLLBACK SEGMENT t18
    TABLESPACE roll
    STORAGE (initial 50K next 50K minextents 2);
```

```
CREATE ROLLBACK SEGMENT t19
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t20
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t21
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t22
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t23
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t24
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t25
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t26
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t27
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t28
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t29
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t30
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t31
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t32
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t33
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t34
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t35
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t36
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t37
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t38
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t39
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t40
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t41
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t42
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t43
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t44
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t45
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t46
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t47
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t48
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t49
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t50
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t51
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t52
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t53
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t54
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t55
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t56
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t57
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t58
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
```

```
CREATE ROLLBACK SEGMENT t59
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t60
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t61
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t62
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t63
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t64
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t65
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t66
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t67
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t68
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t69
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t70
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t71
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t72
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t73
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t74
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t75
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t76
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t77
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t78
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t79
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t80
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t81
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t82
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t83
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t84
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t85
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t86
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t87
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t88
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t89
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t90
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t91
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t92
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t93
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t94
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t95
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t96
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t97
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t98
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
```

```
CREATE ROLLBACK SEGMENT t99
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t100
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t101
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t102
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t103
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t104
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t105
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t106
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t107
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t108
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t109
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t110
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t111
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t112
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t113
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t114
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t115
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t116
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t117
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t118
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t119
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t120
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t121
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t122
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t123
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t124
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t125
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t126
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t127
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t128
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t129
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t130
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t131
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t132
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t133
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t134
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t135
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t136
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t137
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t138
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
```

```
CREATE ROLLBACK SEGMENT t139
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t140
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t141
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t142
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t143
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t144
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t145
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t146
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t147
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t148
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t149
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t150
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t151
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t152
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t153
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t154
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t155
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t156
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t157
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t158
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t159
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t160
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t161
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t162
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t163
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t164
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t165
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t166
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t167
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t168
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t169
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t170
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t171
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t172
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t173
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t174
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t175
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t176
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t177
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t178
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
```

```
CREATE ROLLBACK SEGMENT t179
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t180
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t181
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t182
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t183
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t184
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t185
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t186
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t187
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t188
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t189
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t190
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t191
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t192
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t193
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t194
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t195
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t196
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t197
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t198
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t199
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);
CREATE ROLLBACK SEGMENT t200
  TABLESPACE roll
  STORAGE (initial 50K next 50K minextents 2);

host date;

exit;
```

tpcc_tab.sql

```
rem
rem
============================================================
====+
rem        Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
rem                OPEN SYSTEMS PERFORMANCE GROUP                |
rem                    All Rights Reserved                       |
rem
============================================================
====+
rem  FILENAME
rem      tpcc_tab.sql
rem  DESCRIPTION
rem      Create tables for TPC-C database.
rem
============================================================
====
rem

rem
rem FIRST, create TPCC userid and connect to it.
rem
      grant connect,resource,unlimited tablespace to tpcc identified by tpcc;
      alter user tpcc temporary tablespace temp;
            connect tpcc/tpcc

rem
rem NEXT, DROP all first
rem
      drop cluster icluster including tables;
            drop table warehouse;
            drop table district;
            drop table history;
            drop table orders;
            drop table new_order;
            drop table order_line;
            drop table item;

set timing on

rem
rem LAST, CREATE all tables
rem

rem
rem WAREHOUSE table
rem
```

```
create table warehouse (
        w_id        number,
        w_ytd       number(12),
        w_tax       number(4),
        w_name      varchar2(10),
        w_street_1  varchar2(20),
        w_street_2  varchar2(20),
        w_city      varchar2(20),
        w_state     char(2),
        w_zip       char(9)
    )
  tablespace ware
  initrans 4
  pctfree  95 pctused 4
  storage (initial 1000K next 40K pctincrease 0);


rem
rem DISTRICT table
rem

    create table district (
  d_id        number,
  d_w_id      number,
  d_ytd       number(12),
  d_tax       number(4),
  d_next_o_id number,
  d_name      varchar2(10),
  d_street_1  varchar2(20),
  d_street_2  varchar2(20),
  d_city      varchar2(20),
  d_state     char(2),
  d_zip       char(9)
  )
  tablespace ware
  initrans 4
  pctfree  95 pctused 4
  storage (initial 10000K next 100K pctincrease 0);


rem
rem HISTORY table
rem

    create table history (
  h_c_id      number,
  h_c_d_id    number,
  h_c_w_id    number,
  h_d_id      number,
  h_w_id      number,
  h_date      date,
  h_amount    number(6),
  h_data      varchar2(24)
  )
  tablespace hist
  initrans 3
  pctfree  1
  storage (initial 20K next 85M pctincrease 0
      freelist groups 13 freelists 24);


rem
rem ORDER table
rem

    create table orders (
  o_id        number,
  o_d_id      number,
  o_w_id      number,
  o_c_id      number,
  o_entry_d   date,
  o_carrier_id number,
  o_ol_cnt    number,
  o_all_local number
  )
  tablespace ord
  initrans 3
  pctfree  5
  storage (initial 20K next 30M pctincrease 0
      freelist groups 13 freelists 24);


rem
rem NEW_ORDER table
rem

    create table new_order (
  no_o_id     number,
  no_d_id     number,
  no_w_id     number
  )
  tablespace nord
  initrans 4
  pctfree  5
  storage (initial 20K next 80M pctincrease 0
      freelist groups 13 freelists 24);


rem
rem ORDER_LINE table
rem

    create table order_line (
  ol_o_id     number,
  ol_d_id     number,
  ol_w_id     number,
  ol_number   number,
  ol_delivery_d date,
  ol_i_id     number,
  ol_supply_w_id number,
  ol_quantity number,
  ol_amount   number(6),
  ol_dist_info char(24)
  )
  tablespace ordl
  initrans 4
  pctfree  5
  storage (initial 20K next 530M pctincrease 0
      freelist groups 13 freelists 24);
```

```
rem
rem ITEM table
rem

        create cluster icluster (
        i_id            number(6,0)
        )
        hashkeys        100000
        hash is         i_id
        size            120
        initrans        3
        pctfree         0
        tablespace      items
        storage (initial 14M next 720K pctincrease 0);

        create table item (
        i_id            number(6,0),
        i_im_id         number,
        i_name          varchar2(24),
        i_price         number(5,0),
        i_data          varchar2(50)
        )
        cluster icluster(i_id);


rem
rem done
rem

        exit;
```

tpcc_tab2.sql

```
rem
rem
================================================================
====+
rem        Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
rem                  OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                      All Rights Reserved                        |
rem
================================================================
====+
rem  FILENAME
rem       tpcc_tab2.sql
rem  DESCRIPTION
rem       Create customer table for TPC-C database.
rem
================================================================
====
rem

rem
rem DROP all first
rem
        drop cluster ccluster including tables;
                drop table customer;
set timing on
rem
rem CUSTOMER table
rem

        create cluster ccluster (
          c_id          number(5,0),
          c_d_id        number(2,0),
          c_w_id        number(4,0)
        )
        hashkeys        16920000
        hash is         (c_w_id * 30000 + c_d_id * 3000 + c_id)
        size            850
        initrans        3
        pctfree         0
        tablespace      cust
        storage (initial 1425M next 1425M pctincrease 0 minextents 12);

                create table customer (
        c_id            number(5,0),
        c_d_id          number(2,0),
        c_w_id          number(4,0),
        c_first         varchar2(16),
        c_middle        char(2),
        c_last          varchar2(16),
        c_street_1      varchar2(20),
        c_street_2      varchar2(20),
        c_city          varchar2(20),
        c_state         char(2),
        c_zip           char(9),
        c_phone         char(16),
        c_since         date,
        c_credit        char(2),
        c_credit_lim    number(12),
        c_discount      number(4),
        c_balance       number(12),
        c_ytd_payment   number(12),
        c_payment_cnt   number(8),
        c_delivery_cnt  number(8),
        c_data          varchar2(500)
        )
        cluster ccluster (c_id, c_d_id, c_w_id);

rem
rem done
rem

        exit;
```

tpcc_tab3.sql

```
rem
rem
================================================================
====+
rem        Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
rem                  OPEN SYSTEMS PERFORMANCE GROUP                 |
rem                      All Rights Reserved                        |
rem
================================================================
```

```
====+
rem  FILENAME
rem      tpcc_tab3.sql
rem  DESCRIPTION
rem      Create stock table for TPC-C database.
rem
============================================================
====
rem
rem
rem DROP all first
rem
      drop cluster scluster including tables;
            drop table stock;

set timing on

rem
rem STOCK table
rem

      create cluster scluster (
        s_i_id      number(6,0),
        s_w_id      number(4,0)
      )
      hashkeys       56400000
      hash is        (s_i_id * 564 + s_w_id)
      size           350
      initrans       3
      pctfree        0
      tablespace     stocks
      storage (initial 1425M next 1425M pctincrease 0 minextents 16);

            create table stock (
      s_i_id         number(6,0),
      s_w_id         number(4,0),
      s_quantity     number(6,0),
      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_ytd          number(10,0),
      s_order_cnt    number(6,0),
      s_remote_cnt   number(6,0),
      s_data         varchar2(50)
      )
      cluster scluster (s_i_id, s_w_id);

rem
rem done
rem

      exit;
```

Data Generation Code

tpccload.c

```
\
#ifdef RCSID
static char *RCSid =
  "$Header: tpccload.c 7030100.1 96/05/13 16:20:36 plai Generic<base> $ Copyr
 (c) 1993 Oracle";
#endif /* RCSID */

/*============================================================
=====+
 |       Copyright (c) 1994  Oracle Corp, Redwood Shores, CA       |
 |                  OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                      All Rights Reserved                        |
 +============================================================
=====+
 | FILENAME
 |    tpccload.c
 | DESCRIPTION
 |    Load or generate TPC-C database tables.
 |    Usage: tpccload -M <# of warehouses> [options]
 |              options: -A  load all tables
 |                       -w  load warehouse table
 |                       -d  load district table
 |                       -c  load customer table
 |                       -i  load item table
 |                       -s  load stock table (cluster around s_w_id)
 |                       -S  load stock table (cluster around s_i_id)
 |                       -h  load history table
 |                       -n  load new-order table
 |                       -o <oline file> load order and order-line table
 |                       -b <ware#>  beginning warehouse number
 |                       -e <ware#>  ending warehouse number
 |                       -j <item#>  beginning item number (with -S)
 |                       -k <item#>  ending item number (with -S)
 |                       -g  generate rows to standard output
 +============================================================
=====*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include "tpcc.h"

#define DISTARR 10                  /* district insert array size              */
#define CUSTARR 100                 /* customer insert array size          */
#define STOCARR 100                             /* stock insert array size
*/
#define ITEMARR 100                 /* item insert array size */
#define HISTARR 100             /* history insert array size    */
#define ORDEARR 100             /* order insert array size      */
#define NEWOARR 100             /* new order insert array size  */

#define DISTFAC 10                  /* max. disctrict id        */
```

```
#define CUSTFAC 3000                    /* max. customer id      */
#define STOCFAC 100000                  /* max. stock id         */
#define ITEMFAC 100000                  /* max. item id
*/
#define HISTFAC 30000          /* history / warehouse  */
#define ORDEFAC 3000            /* order / district     */
#define NEWOFAC 900             /* new order / district */

#define C       0               /* constant in non-uniform dist. eqt. */
#define CNUM1   1               /* first constant in non-uniform dist. eqt. */
#define CNUM2   2               /* second constant in non-uniform dist. eqt. */
#define CNUM3   3               /* third constant in non-uniform dist. eqt. */

#define SEED    2               /* seed for random functions */

#define SQLTXTW "INSERT INTO warehouse VALUES (:w_id, 30000000, :w_tax,
:w_name, :w_street_1, \
  :w_street_2, :w_city, :w_state, :w_zip)"

#define SQLTXTD "INSERT INTO district VALUES (:d_id, :d_w_id,3000000,
:d_tax, \
    3001, :d_name, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip)"

#define SQLTXTC "INSERT INTO customer VALUES (:c_id, :c_d_id, :c_w_id, \
  :c_first, 'OE', :c_last, :c_street_1, :c_street_2, :c_city, :c_state, \
  :c_zip, :c_phone, SYSDATE, :c_credit, 5000000, :c_discount, -1000, 1000, 1, \
  0, :c_data)"

#define SQLTXTH "INSERT INTO history VALUES (:h_c_id, :h_c_d_id, :h_c_w_id,
\
  :h_d_id, :h_w_id, SYSDATE, 1000, :h_data)"

#define SQLTXTS "INSERT INTO stock VALUES (:s_i_id, :s_w_id, :s_quantity, \
  :s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05, :s_dist_06, \
  :s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10, 0, 0, 0, :s_data)" \

#define SQLTXTI "INSERT INTO item VALUES (:i_id, :i_im_id, :i_name, :i_price, \
  :i_data)"

#define SQLTXTO1 "INSERT INTO orders VALUES (:o_id, :o_d_id, :o_w_id,
:o_c_id, \
  SYSDATE, :o_carrier_id, :o_ol_cnt, 1)"

#define SQLTXTO2 "INSERT INTO orders VALUES (:o_id, :o_d_id, :o_w_id,
:o_c_id, \
  SYSDATE, 11, :o_ol_cnt, 1)"

#define SQLTXTOL1 "INSERT INTO order_line VALUES (:ol_o_id, :ol_d_id, \
  :ol_w_id, :ol_number, SYSDATE, :ol_i_id, :ol_supply_w_id, 5, 0, \
  :ol_dist_info)"

#define SQLTXTOL2 "INSERT INTO order_line VALUES (:ol_o_id, :ol_d_id, \
  :ol_w_id, :ol_number, to_date('15-Sep-11'), :ol_i_id, :ol_supply_w_id, 5,
:ol_amount, \
  :ol_dist_info)"

#define SQLTXTNO "INSERT INTO new_order VALUES (:no_o_id, :no_d_id,
:no_w_id)"

ldadef tpclda;
csrdef curw, curd, curc, curh, curs, curi, curo1, curo2, curol1, curol2, curno;
unsigned long tpchda[256];

static char *lastname[] = {
  "BAR",
  "OUGHT",
  "ABLE",
  "PRI",
  "PRES",
  "ESE",
  "ANTI",
  "CALLY",
  "ATION",
  "EING"
};

char num9[10];
char num16[17];
char str2[3];
char str24[15][25];
int randperm3000[3000];

myusage()

{

  fprintf (stderr, "\n");
  fprintf (stderr, "Usage:\ttpccload -M <multiplier> [options]\n");
  fprintf (stderr, "options:\n");
  fprintf (stderr, "\t-A :\tload all tables\n");
  fprintf (stderr, "\t-w :\tload warehouse table\n");
  fprintf (stderr, "\t-d :\tload district table\n");
  fprintf (stderr, "\t-c :\tload customer table\n");
  fprintf (stderr, "\t-i :\tload item table\n");
  fprintf (stderr, "\t-s :\tload stock table (cluster around s_w_id)\n");
  fprintf (stderr, "\t-S :\tload stock table (cluster around s_i_id)\n");
  fprintf (stderr, "\t-h :\tload history table\n");
  fprintf (stderr, "\t-n :\tload new-order table\n");
  fprintf (stderr, "\t-o <oline file> :\tload order and order-line table\n");
  fprintf (stderr, "\t-b <ware#> :\tbeginning warehouse number\n");
  fprintf (stderr, "\t-e <ware#> :\tending warehouse number\n");
  fprintf (stderr, "\t-j <item#> :\tbeginning item number (with -S)\n");
  fprintf (stderr, "\t-k <item#> :\tending item number (with -S)\n");
  fprintf (stderr, "\t-g :\tgenerate rows to standard output\n");
  fprintf (stderr, "\n");
  exit(1);

}

errrpt (lda, cur)

csrdef *lda;
csrdef *cur;

{

  text msg[2048];

  if (cur->rc) {
    oerhms (lda, cur->rc, msg, 2048);
```

```
    fprintf (stderr, "TPC-C load error: %s\n", msg);
  }

}

quit ()

{

  if (oclose (&curw))
    errrpt (&tpclda, &curw);

  if (oclose (&curd))
    errrpt (&tpclda, &curd);

  if (oclose (&curc))
    errrpt (&tpclda, &curc);

  if (oclose (&curh))
    errrpt (&tpclda, &curh);

  if (oclose (&curs))
    errrpt (&tpclda, &curs);

  if (oclose (&curi))
    errrpt (&tpclda, &curi);

  if (oclose (&curo1))
    errrpt (&tpclda, &curo1);

  if (oclose (&curo2))
    errrpt (&tpclda, &curo2);

  if (oclose (&curol1))
    errrpt (&tpclda, &curol1);

  if (oclose (&curol2))
    errrpt (&tpclda, &curol2);

  if (oclose (&curno))
    errrpt (&tpclda, &curno);

  if (ologof (&tpclda))
    fprintf (stderr, "TPC-C load error: Error in logging off\n");

}

main (argc, argv)

int argc;
char *argv[];

{

  char *uid="tpcc/tpcc";
  text sqlbuf[1024];
  int scale=0;
  int i, j;
  int loop;
  int loopcount;
  int cid;
  int dwid;
  int cdid;
  int cwid;
  int sid;
  int swid;
  int olcnt;
  int nrows;
  int row;

  int w_id;
  char w_name[11];
  char w_street_1[21];
  char w_street_2[21];
  char w_city[21];
  char w_state[2];
  char w_zip[9];
  int w_tax;

  int d_id[10];
  int d_w_id[10];
  char d_name[10][11];
  char d_street_1[10][21];
  char d_street_2[10][21];
  char d_city[10][21];
  char d_state[10][2];
  char d_zip[10][9];
  int d_tax[10];

  int c_id[100];
  int c_d_id[100];
  int c_w_id[100];
  char c_first[100][17];
  char c_last[100][17];
  char c_street_1[100][21];
  char c_street_2[100][21];
  char c_city[100][21];
  char c_state[100][2];
  char c_zip[100][9];
  char c_phone[100][16];
  char c_credit[100][2];
  int c_discount[100];
  char c_data[100][501];

  int i_id[100];
  int i_im_id[100];
  int i_price[100];
  char i_name[100][25];
  char i_data[100][51];

  int s_i_id[100];
  int s_w_id[100];
  int s_quantity[100];
  char s_dist_01[100][24];
  char s_dist_02[100][24];
  char s_dist_03[100][24];
  char s_dist_04[100][24];
  char s_dist_05[100][24];
  char s_dist_06[100][24];
  char s_dist_07[100][24];
```

```
char s_dist_08[100][24];
char s_dist_09[100][24];
char s_dist_10[100][24];
char s_data[100][51];

int h_w_id[100];
int h_d_id[100];
int h_c_id[100];
char h_data[100][25];

int o_id[100];
int o_d_id[100];
int o_w_id[100];
int o_c_id[100];
int o_carrier_id[100];
int o_ol_cnt[100];

int ol_o_id[15];
int ol_d_id[15];
int ol_w_id[15];
int ol_number[15];
int ol_i_id[15];
int ol_supply_w_id[15];
int ol_amount[15];
char ol_dist_info[15][24];

int no_o_id[100];
int no_d_id[100];
int no_w_id[100];

char sdate[30];

double begin_time, end_time;
double begin_cpu, end_cpu;
double gettime(), getcpu();

extern int getopt();
extern char *optarg;
extern int optind, opterr;

char        *argstr="M:AwdcisShno:b:e:j:k:g";
int opt;
int do_A=0;
int do_w=0;
int do_d=0;
int do_i=0;
int do_c=0;
int do_s=0;
int do_S=0;
int do_h=0;
int do_o=0;
int do_n=0;
int gen=0;
int bware=1;
int eware=0;
int bitem=1;
int eitem=0;

FILE *olfp=NULL;
char olfname[100];

/*---------------------------------------------------------------+
 | Parse command line -- look for scale factor.                  |
 +---------------------------------------------------------------*/

  if (argc == 1) {
    myusage ();
  }

  while ((opt = getopt (argc, argv, argstr)) != -1) {
    switch (opt) {
      case '?': myusage ();
                break;
      case 'M': scale = atoi (optarg);
                break;
      case 'A': do_A = 1;
                break;
      case 'w': do_w = 1;
                break;
      case 'd': do_d = 1;
                break;
      case 'c': do_c = 1;
                break;
      case 'i': do_i = 1;
                break;
      case 's': do_s = 1;
                break;
      case 'S': do_S = 1;
                break;
      case 'h': do_h = 1;
                break;
      case 'n': do_n = 1;
                break;
      case 'o': do_o = 1;
                strcpy (olfname, optarg);
                break;
      case 'b': bware = atoi (optarg);
                break;
      case 'e': eware = atoi (optarg);
                break;
      case 'j': bitem = atoi (optarg);
                break;
      case 'k': eitem = atoi (optarg);
                break;
      case 'g': gen = 1;
                break;
      default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
               fprintf (stderr, "(reached default case in getopt ())\n");
               myusage ();
    }
  }

/*---------------------------------------------------------------*|
|                Rudimentary error checking                     |
|*---------------------------------------------------------------*/

  if (scale  < 1) {
    fprintf (stderr, "Invalid scale factor: '%d'\n", scale);
    myusage ();
  }
```

```
  if (!(do_A || do_w || do_d || do_c || do_i || do_s || do_S || do_h || do_o ||
        do_n)) {
    fprintf (stderr, "What should I load???\n");
    myusage ();
  }

  if (gen && (do_A || (do_w + do_d + do_c + do_i + do_s + do_S + do_h + do_o +
                       do_n > 1))) {
    fprintf (stderr, "Can only generate table one at a time\n");
    myusage ();
  }

  if (do_S && (do_A || do_s)) {
    fprintf (stderr, "Cluster stock table around s_w_id or s_i_id?\n");
    myusage ();
  }

  if (eware <= 0)
    eware = scale;
  if (eitem <= 0)
    eitem = STOCFAC;

  if (do_S) {
    if ((bitem  < 1) || (bitem > STOCFAC)) {
      fprintf (stderr, "Invalid beginning item number: '%d'\n", bitem);
      myusage ();
    }

    if ((eitem  < bitem) || (eitem > STOCFAC)) {
      fprintf (stderr, "Invalid ending item number: '%d'\n", eitem);
      myusage ();
    }
  }

  if ((bware  < 1) || (bware > scale)) {
    fprintf (stderr, "Invalid beginning warehouse number: '%d'\n", bware);
    myusage ();
  }

  if ((eware  < bware) || (eware > scale)) {
    fprintf (stderr, "Invalid ending warehouse number: '%d'\n", eware);
    myusage ();
  }

  if (gen && do_o) {
    if ((olfp = fopen (olfname, "w")) == NULL) {
      fprintf (stderr, "Can't open '%s' for writing order lines\n", olfname);
      myusage ();
    }
  }

/*---------------------------------------------------------------------+
 | Prepare to insert into database.                                    |
 +---------------------------------------------------------------------*/

  sysdate (sdate);
  if (!gen) {

    /* log on to Oracle */

  if (orlon (&tpclda, (ub1 *) tpchda, (text *) uid, -1, (text *) 0, -1, 0)) {
    fprintf (stderr, "TPC-C load error: Error in logging on\n");
    errrpt (&tpclda, &tpclda);
    exit (1);
  }

  fprintf (stderr, "\nConnected to Oracle userid '%s'.\n", uid);

  /* turn off auto-commit */

  if (ocof (&tpclda)) {
    errrpt (&tpclda, &tpclda);
    ologof (&tpclda);
    exit (1);
  }

  /* open cursors */

  if (oopen (&curw, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curw);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curd, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curd);
    oclose (&curw);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curc, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curc);
    oclose (&curw);
    oclose (&curd);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curh, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curh);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curs, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curs);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curi, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
```

```
    errrpt (&tpclda, &curi);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curo1, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curo1);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curo2, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curo2);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curo1);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curol1, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curol1);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curo1);
    oclose (&curo2);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curol2, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curol2);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curo1);
    oclose (&curo2);
    oclose (&curol1);
    ologof (&tpclda);
    exit (1);
  }

  if (oopen (&curno, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curno);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curo1);
    oclose (&curo2);
    oclose (&curol1);
    oclose (&curol2);
    ologof (&tpclda);
    exit (1);
  }

  /* parse statements */

  sprintf ((char *) sqlbuf, SQLTXTW);
  if (oparse (&curw, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTD);
  if (oparse (&curd, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTC);
  if (oparse (&curc, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTH);
  if (oparse (&curh, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curh);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTS);
  if (oparse (&curs, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
  }

  sprintf ((char *) sqlbuf, SQLTXTI);
  if (oparse (&curi, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curi);
```

```
  quit ();
  exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTO1);
if (oparse (&curo1, sqlbuf, -1, 0, 1)) {
  errrpt (&tpclda, &curo1);
  quit ();
  exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTO2);
if (oparse (&curo2, sqlbuf, -1, 0, 1)) {
  errrpt (&tpclda, &curo2);
  quit ();
  exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTOL1);
if (oparse (&curol1, sqlbuf, -1, 0, 1)) {
  errrpt (&tpclda, &curol1);
  quit ();
  exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTOL2);
if (oparse (&curol2, sqlbuf, -1, 0, 1)) {
  errrpt (&tpclda, &curol2);
  quit ();
  exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTNO);
if (oparse (&curno, sqlbuf, -1, 0, 1)) {
  errrpt (&tpclda, &curno);
  quit ();
  exit (1);
}

/* bind variables */

/* warehouse */

if (obndrv (&curw, (text *) ":w_id", -1, (ub1 *) &w_id, sizeof (w_id),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curw);
  quit ();
  exit (1);
}

if (obndrv (&curw, (text *) ":w_name", -1, (ub1 *) w_name, 11,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curw);
  quit ();
  exit (1);
}

if (obndrv (&curw, (text *) ":w_street_1", -1, (ub1 *) w_street_1, 21,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curw);
  quit ();
  exit (1);
}

if (obndrv (&curw, (text *) ":w_street_2", -1, (ub1 *) w_street_2, 21,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curw);
  quit ();
  exit (1);
}

if (obndrv (&curw, (text *) ":w_city", -1, (ub1 *) w_city, 21,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curw);
  quit ();
  exit (1);
}

if (obndrv (&curw, (text *) ":w_state", -1, (ub1 *) w_state, 2,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curw);
  quit ();
  exit (1);
}

if (obndrv (&curw, (text *) ":w_zip", -1, (ub1 *) w_zip, 9,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curw);
  quit ();
  exit (1);
}

if (obndrv (&curw, (text *) ":w_tax", -1, (ub1 *) &w_tax, sizeof (w_tax),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curw);
  quit ();
  exit (1);
}

/* district */

if (obndrv (&curd, (text *) ":d_id", -1, (ub1 *) d_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curd);
  quit ();
  exit (1);
}

if (obndrv (&curd, (text *) ":d_w_id", -1, (ub1 *) d_w_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curd);
  quit ();
  exit (1);
}

if (obndrv (&curd, (text *) ":d_name", -1, (ub1 *) d_name, 11,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curd);
  quit ();
  exit (1);
```

```
}

if (obndrv (&curd, (text *) ":d_street_1", -1, (ub1 *) d_street_1, 21,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curd);
  quit ();
  exit (1);
}

if (obndrv (&curd, (text *) ":d_street_2", -1, (ub1 *) d_street_2, 21,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curd);
  quit ();
  exit (1);
}

if (obndrv (&curd, (text *) ":d_city", -1, (ub1 *) d_city, 21,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curd);
  quit ();
  exit (1);
}

if (obndrv (&curd, (text *) ":d_state", -1, (ub1 *) d_state, 2,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curd);
  quit ();
  exit (1);
}

if (obndrv (&curd, (text *) ":d_zip", -1, (ub1 *) d_zip, 9,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curd);
  quit ();
  exit (1);
}

if (obndrv (&curd, (text *) ":d_tax", -1, (ub1 *) d_tax, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curd);
  quit ();
  exit (1);
}

/* customer */

if (obndrv (&curc, (text *) ":c_id", -1, (ub1 *) c_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_d_id", -1, (ub1 *) c_d_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_w_id", -1, (ub1 *) c_w_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_first", -1, (ub1 *) c_first, 17,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_last", -1, (ub1 *) c_last, 17,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_street_1", -1, (ub1 *) c_street_1, 21,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_street_2", -1, (ub1 *) c_street_2, 21,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_city", -1, (ub1 *) c_city, 21,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_state", -1, (ub1 *) c_state, 2,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_zip", -1, (ub1 *) c_zip, 9,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_phone", -1, (ub1 *) c_phone, 16,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
```

```
  exit (1);
}

if (obndrv (&curc, (text *) ":c_credit", -1, (ub1 *) c_credit, 2,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_discount", -1, (ub1 *) c_discount,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1,
        -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

if (obndrv (&curc, (text *) ":c_data", -1, (ub1 *) c_data, 501,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curc);
  quit ();
  exit (1);
}

/* item */

if (obndrv (&curi, (text *) ":i_id", -1, (ub1 *) i_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curi);
  quit ();
  exit (1);
}

if (obndrv (&curi, (text *) ":i_im_id", -1, (ub1 *) i_im_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curi);
  quit ();
  exit (1);
}

if (obndrv (&curi, (text *) ":i_name", -1, (ub1 *) i_name, 25,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curi);
  quit ();
  exit (1);
}

if (obndrv (&curi, (text *) ":i_price", -1, (ub1 *) i_price,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1,
        -1)) {
  errrpt (&tpclda, &curi);
  quit ();
  exit (1);
}

if (obndrv (&curi, (text *) ":i_data", -1, (ub1 *) i_data, 51,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curi);
  quit ();
  exit (1);
}

/* stock */

if (obndrv (&curs, (text *) ":s_i_id", -1, (ub1 *) s_i_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_w_id", -1, (ub1 *) s_w_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_quantity", -1, (ub1 *) s_quantity,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_01", -1, (ub1 *) s_dist_01, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_02", -1, (ub1 *) s_dist_02, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_03", -1, (ub1 *) s_dist_03, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_04", -1, (ub1 *) s_dist_04, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_05", -1, (ub1 *) s_dist_05, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}
```

```
if (obndrv (&curs, (text *) ":s_dist_06", -1, (ub1 *) s_dist_06, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_07", -1, (ub1 *) s_dist_07, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_08", -1, (ub1 *) s_dist_08, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_09", -1, (ub1 *) s_dist_09, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_10", -1, (ub1 *) s_dist_10, 24,
        SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

if (obndrv (&curs, (text *) ":s_data", -1, (ub1 *) s_data, 51,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curs);
  quit ();
  exit (1);
}

/* history */

if (obndrv (&curh, (text *) ":h_c_id", -1, (ub1 *) h_c_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curh);
  quit ();
  exit (1);
}

if (obndrv (&curh, (text *) ":h_c_d_id", -1, (ub1 *) h_d_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curh);
  quit ();
  exit (1);
}

if (obndrv (&curh, (text *) ":h_c_w_id", -1, (ub1 *) h_w_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curh);
  quit ();
  exit (1);
}

if (obndrv (&curh, (text *) ":h_d_id", -1, (ub1 *) h_d_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curh);
  quit ();
  exit (1);
}

if (obndrv (&curh, (text *) ":h_w_id", -1, (ub1 *) h_w_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curh);
  quit ();
  exit (1);
}

if (obndrv (&curh, (text *) ":h_data", -1, (ub1 *) h_data, 25,
        SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curh);
  quit ();
  exit (1);
}

/* order_line (delivered) */

if (obndrv (&curol1, (text *) ":ol_o_id", -1, (ub1 *) ol_o_id,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol1);
  quit ();
  exit (1);
}

if (obndrv (&curol1, (text *) ":ol_d_id", -1, (ub1 *) ol_d_id,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol1);
  quit ();
  exit (1);
}

if (obndrv (&curol1, (text *) ":ol_w_id", -1, (ub1 *) ol_w_id,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol1);
  quit ();
  exit (1);
}

if (obndrv (&curol1, (text *) ":ol_number", -1, (ub1 *) ol_number,
    sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol1);
  quit ();
  exit (1);
}

if (obndrv (&curol1, (text *) ":ol_i_id", -1, (ub1 *) ol_i_id,
    sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol1);
```

```
  quit ();
  exit (1);
}

if (obndrv (&curol1, (text *) ":ol_supply_w_id", -1,
        (ub1 *) ol_supply_w_id, sizeof (int), SQLT_INT, -1,
        (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol1);
  quit ();
  exit (1);
}

if (obndrv (&curol1, (text *) ":ol_dist_info", -1, (ub1 *) ol_dist_info,
        24, SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol1);
  quit ();
  exit (1);
}

/* order_line (not delivered) */

if (obndrv (&curol2, (text *) ":ol_o_id", -1, (ub1 *) ol_o_id,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol2);
  quit ();
  exit (1);
}

if (obndrv (&curol2, (text *) ":ol_d_id", -1, (ub1 *) ol_d_id,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol2);
  quit ();
  exit (1);
}

if (obndrv (&curol2, (text *) ":ol_w_id", -1, (ub1 *) ol_w_id,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol2);
  quit ();
  exit (1);
}

if (obndrv (&curol2, (text *) ":ol_number", -1, (ub1 *) ol_number,
    sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol2);
  quit ();
  exit (1);
}

if (obndrv (&curol2, (text *) ":ol_i_id", -1, (ub1 *) ol_i_id,
    sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol2);
  quit ();
  exit (1);
}

if (obndrv (&curol2, (text *) ":ol_supply_w_id", -1,
        (ub1 *) ol_supply_w_id, sizeof (int), SQLT_INT, -1,
        (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol2);
  quit ();
  exit (1);
}

if (obndrv (&curol2, (text *) ":ol_amount", -1, (ub1 *) ol_amount,
    sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol2);
  quit ();
  exit (1);
}

if (obndrv (&curol2, (text *) ":ol_dist_info", -1, (ub1 *) ol_dist_info,
        24, SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curol2);
  quit ();
  exit (1);
}

/* orders (delivered) */

if (obndrv (&curo1, (text *) ":o_id", -1, (ub1 *) o_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo1);
  quit ();
  exit (1);
}

if (obndrv (&curo1, (text *) ":o_d_id", -1, (ub1 *) o_d_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo1);
  quit ();
  exit (1);
}

if (obndrv (&curo1, (text *) ":o_w_id", -1, (ub1 *) o_w_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo1);
  quit ();
  exit (1);
}

if (obndrv (&curo1, (text *) ":o_c_id", -1, (ub1 *) o_c_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo1);
  quit ();
  exit (1);
}

if (obndrv (&curo1, (text *) ":o_carrier_id", -1, (ub1 *) o_carrier_id,
    sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo1);
  quit ();
  exit (1);
}

if (obndrv (&curo1, (text *) ":o_ol_cnt", -1, (ub1 *) o_ol_cnt,
    sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo1);
  quit ();
  exit (1);
```

```
}

/* orders (not delivered) */

if (obndrv (&curo2, (text *) ":o_id", -1, (ub1 *) o_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo2);
  quit ();
  exit (1);
}

if (obndrv (&curo2, (text *) ":o_d_id", -1, (ub1 *) o_d_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo2);
  quit ();
  exit (1);
}

if (obndrv (&curo2, (text *) ":o_w_id", -1, (ub1 *) o_w_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo2);
  quit ();
  exit (1);
}

if (obndrv (&curo2, (text *) ":o_c_id", -1, (ub1 *) o_c_id, sizeof (int),
        SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo2);
  quit ();
  exit (1);
}

if (obndrv (&curo2, (text *) ":o_ol_cnt", -1, (ub1 *) o_ol_cnt,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curo2);
  quit ();
  exit (1);
}

/* new order */

if (obndrv (&curno, (text *) ":no_o_id", -1, (ub1 *) no_o_id,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curno);
  quit ();
  exit (1);
}

if (obndrv (&curno, (text *) ":no_d_id", -1, (ub1 *) no_d_id,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curno);
  quit ();
  exit (1);
}

if (obndrv (&curno, (text *) ":no_w_id", -1, (ub1 *) no_w_id,
        sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
  errrpt (&tpclda, &curno);
  quit ();
  exit (1);
    }
  }

/*---------------------------------------------------------------+
 | Initialize random number generator                            |
 +---------------------------------------------------------------*/

  srand (getpid ());
  srand48 (getpid ());
  initperm ();


/*---------------------------------------------------------------+
 | Load the WAREHOUSE table.                                     |
 +---------------------------------------------------------------*/

  if (do_A || do_w) {
    nrows = eware - bware + 1;

    fprintf (stderr, "Loading/generating warehouse: w%d - w%d (%d rows)\n",
          bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    for (loop = bware; loop <= eware; loop++) {

      w_tax = (rand () % 2001);
      randstr (w_name, 6, 10);
      randstr (w_street_1, 10, 20);
      randstr (w_street_2, 10, 20);
      randstr (w_city, 10, 20);
      randstr (str2, 2, 2);
      randnum (num9, 9);
      num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

      if (gen) {
        printf ("%d 30000000 %d %s %s %s %s %s %s\n", loop, w_tax,
              w_name, w_street_1, w_street_2, w_city, str2, num9);
        fflush (stdout);
      }
      else {
        w_id = loop;
        strncpy (w_state, str2, 2);
        strncpy (w_zip, num9, 9);

        if (oexec (&curw)) {
          errrpt (&tpclda, &curw);
          orol (&tpclda);
          fprintf (stderr, "Aborted at warehouse %d\n", loop);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at warehouse %d\n", loop);
          quit ();
          exit (1);
        }
```

```
      }
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
            nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the DISTRICT table.                                      |
 +---------------------------------------------------------------*/

  if (do_A || do_d) {
    nrows = (eware - bware + 1) * DISTFAC;

    fprintf (stderr, "Loading/generating district: w%d - w%d (%d rows)\n",
            bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    dwid = bware - 1;

    for (row = 0; row < nrows; ) {
      dwid++;

      for (i = 0; i < DISTARR; i++, row++) {
        d_tax[i] = (rand () % 2001);
        randstr (d_name[i], 6, 10);
        randstr (d_street_1[i], 10, 20);
        randstr (d_street_2[i], 10, 20);
        randstr (d_city[i], 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

        if (gen) {
          /* printf ("%d %d %s %s %s %s %s %s %d 30000.0 3001\n",
                i + 1, dwid, d_name[i], d_street_1[i], d_street_2[i],
                d_city[i], str2, num9, d_tax[i]); */
                                /* Reordered columns */
          printf ("%d %d 3000000 %d 3001 %s %s %s %s %s %s\n",
                i + 1, dwid, d_tax[i], d_name[i], d_street_1[i],
                                d_street_2[i], d_city[i], str2, num9 );
        }
        else {
          d_id[i] = i + 1;
          d_w_id[i] = dwid;
          strncpy (d_state[i], str2, 2);
          strncpy (d_zip[i], num9, 9);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curd, DISTARR, 0)) {
          errrpt (&tpclda, &curd);
          orol (&tpclda);
          fprintf (stderr, "Aborted at warehouse %d, district 1\n", dwid);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at warehouse %d, district 1\n", dwid);
          quit ();
          exit (1);
        }
      }
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
            nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the CUSTOMER table.                                      |
 +---------------------------------------------------------------*/

  if (do_A || do_c) {
    nrows = (eware - bware + 1) * CUSTFAC * DISTFAC;

    fprintf (stderr, "Loading/generating customer: w%d - w%d (%d rows)\n   ",
            bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < CUSTARR; i++, row++) {
        cid++;
        if (cid > CUSTFAC) {        /* cycle cust id */
          cid = 1;                  /* cheap mod */
          cdid++;                   /* shift district cycle */
          if (cdid > DISTFAC) {
            cdid = 1;
            cwid++;                 /* shift warehouse cycle */
          }
        }
        c_id[i] = cid;
        c_d_id[i] = cdid;
        c_w_id[i] = cwid;
        if (cid <= 1000)
          randlastname (c_last[i], cid - 1);
        else
          randlastname (c_last[i], NURand (255, 0, 999, CNUM1));
        c_credit[i][1] = 'C';
```

```
        if (rand () % 10)
          c_credit[i][0] = 'G';
        else
          c_credit[i][0] = 'B';
        c_discount[i] = (rand () % 5001);
        randstr (c_first[i], 8, 16);
        randstr (c_street_1[i], 10, 20);
        randstr (c_street_2[i], 10, 20);
        randstr (c_city[i], 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
        randnum (num16, 16);
        randstr (c_data[i], 300, 500);

        if (gen) {
          printf ("%d %d %d %s OE %s %s %s %s %s %s %s %cC 5000000
%d -1000 1000 1 0 %s\n",
                  cid, cdid, cwid, c_first[i], c_last[i],
                  c_street_1[i], c_street_2[i], c_city[i], str2, num9,
                  num16, sdate, c_credit[i][0], c_discount[i], c_data[i]);
        }
        else {
          strncpy (c_state[i], str2, 2);
          strncpy (c_zip[i], num9, 9);
          strncpy (c_phone[i], num16, 16);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curc, CUSTARR, 0)) {
          errrpt (&tpclda, &curc);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                   c_w_id[0], c_d_id[0], c_id[0]);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                   c_w_id[0], c_d_id[0], c_id[0]);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
             nrows, end_time - begin_time, end_cpu - begin_cpu);
    }


/*-------------------------------------------------------------+
 | Load the ITEM table.                                        |
 +-------------------------------------------------------------*/

    if (do_A || do_i) {
      nrows = ITEMFAC;

      fprintf (stderr, "Loading/generating item: (%d rows)\n   ", nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      loopcount = 0;

      for (row = 0; row < nrows; ) {
        for (i = 0; i < ITEMARR; i++, row++) {
          i_im_id[i] = (rand () % 10000) + 1;
          i_price[i] = ((rand () % 9901) + 100);
          randstr (i_name[i], 14, 24);
          randdatastr (i_data[i], 26, 50);

          if (gen) {
            printf ("%d %d %s %d %s\n", row + 1, i_im_id[i], i_name[i],
                    i_price[i], i_data[i]);
          }
          else {
            i_id[i] = row + 1;
          }
        }

        if (gen) {
          fflush (stdout);
        }
        else {
          if (oexn (&curi, ITEMARR, 0)) {
            errrpt (&tpclda, &curi);
            orol (&tpclda);
            fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
            quit ();
            exit (1);
          }
          else if (ocom (&tpclda)) {
            errrpt (&tpclda, &tpclda);
            orol (&tpclda);
            fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
            quit ();
            exit (1);
          }
        }

        if ((++loopcount) % 50)
          fprintf (stderr, ".");
        else
          fprintf (stderr, " %d rows committed\n   ", row);
      }
```

```
    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
            nrows, end_time - begin_time, end_cpu - begin_cpu);
  }


/*---------------------------------------------------------------+
 | Load the STOCK table.                                         |
 +---------------------------------------------------------------*/

  if (do_A || do_s) {

    nrows = (eware - bware + 1) * STOCFAC;

    fprintf (stderr, "Loading/generating stock: w%d - w%d (%d rows)\n   ",
            bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    sid = 0;
    swid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < STOCARR; i++, row++) {
        if (++sid > STOCFAC) {        /* cheap mod */
          sid = 1;
          swid++;
        }
        s_quantity[i] = (rand () % 91) + 10;
        randstr (str24[0], 24, 24);
        randstr (str24[1], 24, 24);
        randstr (str24[2], 24, 24);
        randstr (str24[3], 24, 24);
        randstr (str24[4], 24, 24);
        randstr (str24[5], 24, 24);
        randstr (str24[6], 24, 24);
        randstr (str24[7], 24, 24);
        randstr (str24[8], 24, 24);
        randstr (str24[9], 24, 24);
        randdatastr (s_data[i], 26, 50);

        if (gen) {
          printf ("%d %d %d %s %s %s %s %s %s %s %s %s 0 0 0 %s\n",
                sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
                str24[3], str24[4], str24[5], str24[6], str24[7],
                str24[8], str24[9], s_data[i]);
        }
        else {
          s_i_id[i] = sid;
          s_w_id[i] = swid;
          strncpy (s_dist_01[i], str24[0], 24);
          strncpy (s_dist_02[i], str24[1], 24);
          strncpy (s_dist_03[i], str24[2], 24);
          strncpy (s_dist_04[i], str24[3], 24);
          strncpy (s_dist_05[i], str24[4], 24);
          strncpy (s_dist_06[i], str24[5], 24);
          strncpy (s_dist_07[i], str24[6], 24);
          strncpy (s_dist_08[i], str24[7], 24);
          strncpy (s_dist_09[i], str24[8], 24);
          strncpy (s_dist_10[i], str24[9], 24);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curs, STOCARR, 0)) {
          errrpt (&tpclda, &curs);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
            nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the STOCK table (cluster around s_i_id).                 |
 +---------------------------------------------------------------*/

  if (do_S) {

    nrows = (eitem - bitem + 1) * (eware - bware + 1);

",  fprintf (stderr, "Loading/generating stock: i%d - i%d, w%d - w%d (%d rows)\n
            bitem, eitem, bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    sid = bitem;
    swid = bware - 1;
    loopcount = 0;
```

```
for (row = 0; row < nrows; ) {
  for (i = 0; i < STOCARR; i++, row++) {
    if (++swid > eware) {          /* cheap mod */
      swid = bware;
      sid++;
    }
    s_quantity[i] = (rand () % 91) + 10;
    randstr (str24[0], 24, 24);
    randstr (str24[1], 24, 24);
    randstr (str24[2], 24, 24);
    randstr (str24[3], 24, 24);
    randstr (str24[4], 24, 24);
    randstr (str24[5], 24, 24);
    randstr (str24[6], 24, 24);
    randstr (str24[7], 24, 24);
    randstr (str24[8], 24, 24);
    randstr (str24[9], 24, 24);
    randdatastr (s_data[i], 26, 50);

    if (gen) {
      printf ("%d %d %d %s %s %s %s %s %s %s %s %s 0 0 0 %s\n",
            sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
            str24[3], str24[4], str24[5], str24[6], str24[7],
            str24[8], str24[9], s_data[i]);
    }
    else {
      s_i_id[i] = sid;
      s_w_id[i] = swid;
      strncpy (s_dist_01[i], str24[0], 24);
      strncpy (s_dist_02[i], str24[1], 24);
      strncpy (s_dist_03[i], str24[2], 24);
      strncpy (s_dist_04[i], str24[3], 24);
      strncpy (s_dist_05[i], str24[4], 24);
      strncpy (s_dist_06[i], str24[5], 24);
      strncpy (s_dist_07[i], str24[6], 24);
      strncpy (s_dist_08[i], str24[7], 24);
      strncpy (s_dist_09[i], str24[8], 24);
      strncpy (s_dist_10[i], str24[9], 24);
    }
  }

  if (gen) {
    fflush (stdout);
  }
  else {
    if (oexn (&curs, STOCARR, 0)) {
      errrpt (&tpclda, &curs);
      orol (&tpclda);
      fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
            s_i_id[0]);
      quit ();
      exit (1);
    }
    else if (ocom (&tpclda)) {
      errrpt (&tpclda, &tpclda);
      orol (&tpclda);
      fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
            s_i_id[0]);
      quit ();
      exit (1);
    }
  }

  if ((++loopcount) % 50)
    fprintf (stderr, ".");
  else
    fprintf (stderr, " %d rows committed\n   ", row);
 }

 end_time = gettime ();
 end_cpu = getcpu ();
 fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-------------------------------------------------------------+
| Load the HISTORY table.                                      |
+-------------------------------------------------------------*/

if (do_A || do_h) {
 nrows = (eware - bware + 1) * HISTFAC;

 fprintf (stderr, "Loading/generating history: w%d - w%d (%d rows)\n   ",
          bware, eware, nrows);

 begin_time = gettime ();
 begin_cpu = getcpu ();

 cid = 0;
 cdid = 1;
 cwid = bware;
 loopcount = 0;

 for (row = 0; row < nrows; ) {
   for (i = 0; i < HISTARR; i++, row++) {
     cid++;
     if (cid > CUSTFAC) {       /* cycle cust id */
       cid = 1;                 /* cheap mod */
       cdid++;                  /* shift district cycle */
       if (cdid > DISTFAC) {
         cdid = 1;
         cwid++;                /* shift warehouse cycle */
       }
     }
     h_c_id[i] = cid;
     h_d_id[i] = cdid;
     h_w_id[i] = cwid;
     randstr (h_data[i], 12, 24);
     if (gen) {
       printf ("%d %d %d %d %d %s 1000 %s\n", cid, cdid, cwid, cdid,
             cwid, sdate, h_data[i]);
     }
   }

   if (gen) {
     fflush (stdout);
   }
   else {
```

```
        if (oexn (&curh, HISTARR, 0)) {
          errrpt (&tpclda, &curh);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                h_w_id[0], h_d_id[0], h_c_id[0]);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                h_w_id[0], h_d_id[0], h_c_id[0]);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the ORDERS and ORDER-LINE table.                         |
 +---------------------------------------------------------------*/

  if (do_A || do_o) {
    nrows = (eware - bware + 1) * ORDEFAC * DISTFAC;

    fprintf (stderr, "Loading/generating orders and order-line: w%d - w%d (%d ord,
~%d ordl)\n   ",
          bware, eware, nrows, nrows * 10);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < ORDEARR; i++, row++) {
        cid++;
        if (cid > ORDEFAC) {       /* cycle cust id */
          cid = 1;                 /* cheap mod */
          cdid++;                  /* shift district cycle */
          if (cdid > DISTFAC) {
            cdid = 1;
            cwid++;                /* shift warehouse cycle */
          }
        }
        o_carrier_id[i] = rand () % 10 + 1;
        o_ol_cnt[i] = olcnt = rand () % 11 + 5;

        if (gen) {
          if (cid < 2101) {
            printf ("%d %d %d %d %s %d %d 1\n", cid, cdid, cwid,
                  randperm3000[cid - 1], sdate,o_carrier_id[i],
                  o_ol_cnt[i]);
          }
          else {
                    /* set carrierid to 11 instead of null 9/24 Saar  */
            printf ("%d %d %d %d %s 11 %d 1\n", cid, cdid, cwid,
                  randperm3000[cid - 1], sdate, o_ol_cnt[i]);
          }
        }
        else {
          o_id[i] = cid;
          o_d_id[i] = cdid;
          o_w_id[i] = cwid;
          o_c_id[i] = randperm3000[cid - 1];
        }

        for (j = 0; j < o_ol_cnt[i]; j++) {
          ol_i_id[j] = sid = lrand48 () % 100000 + 1;
          if (cid < 2101)
            ol_amount[j] = 0;
          else
            ol_amount[j] = (lrand48 () % 999999 + 1) ;
          randstr (str24[j], 24, 24);

          if (gen) {
            if (cid < 2101) {
              fprintf (olfp, "%d %d %d %d %s %d %d 5 %ld %s\n", cid,
                    cdid, cwid, j + 1, sdate, ol_i_id[j], cwid,
                    ol_amount[j], str24[j]);
            }
            else {
                     /* change from null date, reorg columns  9/24 Saar  */
              fprintf (olfp, "%d %d %d %d 15-Sep-11 %d %d 5 %ld %s\n", cid,
                    cdid, cwid, j + 1, ol_i_id[j], cwid,
                    ol_amount[j], str24[j]);
            }
          }
          else {
            ol_o_id[j] = cid;
            ol_d_id[j] = cdid;
            ol_w_id[j] = cwid;
            ol_number[j] = j + 1;
            ol_supply_w_id[j] = cwid;
            strncpy (ol_dist_info[j], str24[j], 24);
          }
        }

        if (gen) {
          fflush (olfp);
        }
        else {
          if (cid < 2101) {
            if (oexn (&curol1, olcnt, 0)) {
```

```
          errrpt (&tpclda, &curol1);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                cwid, cdid, cid);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                cwid, cdid, cid);
          quit ();
          exit (1);
        }
      }
      else {
        if (oexn (&curol2, olcnt, 0)) {
          errrpt (&tpclda, &curol2);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                cwid, cdid, cid);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
                cwid, cdid, cid);
          quit ();
          exit (1);
        }
      }
    }
  }

  if (gen) {
    fflush (stdout);
  }
  else {
    if (cid < 2101) {
      if (oexn (&curo1, ORDEARR, 0)) {
        errrpt (&tpclda, &curo1);
        orol (&tpclda);
        fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
              cwid, cdid, cid);
        quit ();
        exit (1);
      }
      else if (ocom (&tpclda)) {
        errrpt (&tpclda, &tpclda);
        orol (&tpclda);
        fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
              cwid, cdid, cid);
        quit ();
        exit (1);
      }
    }
    else {
      if (oexn (&curo2, ORDEARR, 0)) {
        errrpt (&tpclda, &curo2);
        orol (&tpclda);
        fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
              cwid, cdid, cid);
        quit ();
        exit (1);
      }
      else if (ocom (&tpclda)) {
        errrpt (&tpclda, &tpclda);
        orol (&tpclda);
        fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
              cwid, cdid, cid);
        quit ();
        exit (1);
      }
    }
  }

  if ((++loopcount) % 50)
    fprintf (stderr, ".");
  else
    fprintf (stderr, " %d orders committed\n   ", row);
 }

 end_time = gettime ();
 end_cpu = getcpu ();
 fprintf (stderr, "Done.  %d orders loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
        nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-------------------------------------------------------------+
| Load the NEW-ORDER table.                                    |
+-------------------------------------------------------------*/

if (do_A || do_n) {
  nrows = (eware - bware + 1) * NEWOFAC * DISTFAC;

  fprintf (stderr, "Loading/generating new-order: w%d - w%d (%d rows)\n   ",
        bware, eware, nrows);

  begin_time = gettime ();
  begin_cpu = getcpu ();

  cid = 0;
  cdid = 1;
  cwid = bware;
  loopcount = 0;

  for (row = 0; row < nrows; ) {
    for (i = 0; i < NEWOARR; i++, row++) {
      cid++;
      if (cid > NEWOFAC) {
        cid = 1;
        cdid++;
        if (cdid > DISTFAC) {
          cdid = 1;
          cwid++;
        }
```

```
        }

        if (gen) {
          printf ("%d %d %d\n", cid + 2100, cdid, cwid);
        }
        else {
          no_o_id[i] = cid + 2100;
          no_d_id[i] = cdid;
          no_w_id[i] = cwid;
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        if (oexn (&curno, NEWOARR, 0)) {
          errrpt (&tpclda, &curno);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
                cwid, cdid, cid + 2100);
          quit ();
          exit (1);
        }
        else if (ocom (&tpclda)) {
          errrpt (&tpclda, &tpclda);
          orol (&tpclda);
          fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n   ",
                cwid, cdid, cid + 2100);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 45)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
 cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*-------------------------------------------------------------+
 | clean up and exit.                                          |
 +-------------------------------------------------------------*/

  if (olfp)
    fclose (olfp);
  if (!gen)
    quit ();
  exit (0);

}
initperm ()
{

  int i;
  int pos;
  int temp;

  /* init randperm3000 */

  for (i = 0; i < 3000; i++)
    randperm3000[i] = i + 1;
  for (i = 3000; i > 0; i--) {
    pos = rand () % i;
    temp = randperm3000[i - 1];
    randperm3000[i - 1] = randperm3000[pos];
    randperm3000[pos] = temp;
  }

}

randstr (str, x, y)

char *str;
int x;
int y;

{

  int i, j;
  int len;

  len = (rand () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = rand () % 62;
    if (j < 26)
      str[i] = (char) (j + 'a');
    else if (j < 52)
      str[i] = (char) (j - 26 + 'A');
    else
      str[i] = (char) (j - 52 + '0');
  }
  str[len] = '\0';

}

randdatastr (str, x, y)

char *str;
int x;
int y;

{

  int i, j;
  int len;
  int pos;

  len = (rand () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = rand () % 62;
    if (j < 26)
```

```
      str[i] = (char) (j + 'a');
    else if (j < 52)
      str[i] = (char) (j - 26 + 'A');
    else
      str[i] = (char) (j - 52 + '0');
  }
  str[len] = '\0';
  if ((rand () % 10) == 0) {
    pos = (rand () % (len - 8));
    str[pos] = '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';
  }

}

randnum (str, len)

char *str;
int len;

{

  int i;

  for (i = 0; i < len; i++)
    str[i] = (char) (rand () % 10 + '0');
  str[len] = '\0';

}

randlastname (str, id)

char *str;
int id;

{

  id = id % 1000;
  strcpy (str, lastname[id / 100]);
  strcat (str, lastname[(id / 10) % 10]);
  strcat (str, lastname[id % 10]);

}

NURand (A, x, y, cnum)

int A, x, y, cnum;

{

  int a, b;

  a = lrand48 () % (A + 1);
  b = (lrand48 () % (y - x + 1)) + x;
  return ((((a | b) + cnum) % (y - x + 1)) + x);

}

sysdate (sdate)

char *sdate;

{

  time_t tp;
  struct tm *tmptr;

  time (&tp);
  tmptr = localtime (&tp);
  strftime (sdate, 29, "%d-%b-%y", tmptr);

}
```

Appendix D: RTE Driver Scripts

The following code sections show how the transactions are generated and how statistics are gathered. Each of the transaction functions generates the input data for that transaction, sends it to the client, reads the output form and computes keying, response and think time statistics.

This is the main loop of the RTE:

```
/* run for ramp up without capturing the stats */
                i=0;
                in_ramp = 1;
                while (1)
                {
                        tx_type = do_menu();   /* Select transaction */
                        switch (tx_type) {
                        case NEWORDER:
                                do_neworder();
                                break;
                        case PAYMENT:
                                do_payment();
                                break;
                        case DELIVERY:
                                do_delivery();
                                break;
                        case ORDSTAT:
                                do_ordstat();
                                break;
                        case STOCKLEVEL:
                                do_stocklevel();
                                break;
                        default:
                                fprintf(stderr, "%s: Slave %d: Internal error.
 Tx-type = %d\n",
                                hostname, slave_num, tx_type);
                                cleanup(-1);
                        }
                        end_time = gettime();
                        if ( end_time >= control->end_rampup &&
                                        end_time < control->end_stdystate
)
                                        in_ramp = 0;
```

```
                        else
                                in_ramp = 1;
                        if (end_time >= control->end_rampdown)
                                break;
        }
The do_menu function selects the transaction to execute based on the weighted
 distribution
algorithm.
int
do_menu()
{
        int val, result, menu_start, menu_end, menu_resp;
        char ch;
        /* Read menu line from client */
        /* Choose tx. type*/
        /* Now select menu and compute menu response time */
        menu_start = gettime();
        /* Write menu selection to client */
        /* Read input form for this transaction type */
        menu_end = gettime();
        menu_resp = menu_end - menu_start;
        if ( ! in_ramp) {
                statsp->menu_resp += menu_resp;
                /* Post in histogram bucket */
                if ((menu_resp / MENU_BUCKET) < MENU_MAX)
                        statsp->menu_hist[menu_resp /
 MENU_BUCKET]++;
                else
                        statsp->menu_hist[MENU_MAX - 1]++;
                if (menu_resp > statsp->menu_max)
                        statsp->menu_max = menu_resp;
        }
        return(result);
}
/*
 * Function: do_neworder
 * This function executes the neworder transaction
 * It generates all the input fields, sends it to the
 * client over the keying time, measures the response
 * time, reads the results and delays for the think time.
 */
 /* The code for the other transactions is similar */
do_neworder()
{
        struct newo_fld no;
        struct items_fld *itemp = no.items;
        int ol_cnt, rbk, remote = 0, i, x;
        char *bufp = fldbuf;
        int start_time, end_time, key_time, resp_time, elapse_time, del;
        start_time = gettime();
        /* Now wait for keying time */
        poll (0, 0, NEWO_KEY);
        /* Generate all input data */
        no.d_id = random(1, 10);
        no.c_id = NURand(1023, 1, 3000, CONST_CID);
        ol_cnt = random(5, 15);
        rbk = random(1, 100);   /* trans. to be rolledback */
        sprintf(bufp, "%02d%04d", no.d_id, no.c_id);
        bufp += strlen(bufp);
        /* Generate all the item fields */
        for (i=0; i < ol_cnt; i++, itemp++) {
                itemp->ol_i_id = NURand(8191, 1, 100000, CONST_IID);
                /* If last item and rbk, select unused item */
                if (i == ol_cnt - 1 && rbk == 1) {
                        itemp->ol_i_id = 100001;
                }
                x = random(1, 100);
                if ( x > 1)
                        itemp->ol_supply_w_id = W_ID;
                else {
                        /* Select a warehouse other than w_id */
                        do {
                                x = random(1, control->scale);
                        } while (x == W_ID);
                        itemp->ol_supply_w_id = x;
                        remote++;
                }
                itemp->ol_quantity = random(1, 10);
                sprintf(bufp, "%04d%06d%02d",itemp->ol_supply_w_id,
                itemp->ol_i_id, itemp->ol_quantity);
                bufp += strlen(bufp);
        }
        strcpy(bufp, leave_key);
        bufp += 2;
        /* Compute keying time info */
        end_time = gettime();
        key_time = end_time - start_time;
        start_time = end_time;

        /* Now send fields to client */
        /* Read output screen from client */
        end_time = gettime();
        /* Store elapse time info for thruput */
        elapse_time = end_time - control->start_time;
        /* compute the how long it took to run the tx */
        resp_time = end_time - start_time + control->newo_delta;
        /* Wait think time */
        del = delay(control->newo_think, 5*control->newo_think);
        poll(0, 0, del + control->newo_delta);
        end_time = gettime();
        /* Now post all stats */
        if ( ! in_ramp && end_time <= control->end_stdystate) {
                statsp->newo_cnt++; /* another one bytes the dust */
                if ( rbk == 1 )
                        statsp->newo_rbkcnt++;
                statsp->newo_remote += remote;
                statsp->newo_olcnt += ol_cnt;
                statsp->newo_key += key_time;
                /* Save keying time in histogram bucket */
                statsp->newo_resp += (double) resp_time;      /* sum up
 the response time */
                /* Save response time in histogram bucket */
                statsp->newo_think += (double) del;
                /* Save think time in histogram bucket */
        }
}
```

Appendix E: Third-Party Letters

GIBRALTAR
COMPUTER
LAN-WAN-IBM-MIDRANGE-MAINFRAME

WEDNESDAY, DECEMBER 18, 1996

ATTN: CHRIS KING
IBM CORP
3039 CORNWALLIS ROAD
RTP, NC 27709

DEAR CHRIS,

THANK YOU FOR YOUR INTEREST IN HAVING GIBRALTAR COMPUTER GROUP RESPOND TO YOUR BID. I HOPE THE INFORMATION ENCLOSED ON THIS BID WILL FULFILL YOUR NEEDS. IF YOU HAVE ANY FURTHER QUESTIONS PLEASE FEEL FREE TO CALL ME AT (770) 232-7744 X118.

ALL PRICES WILL BE VALID FOR 60 (SIXTY) DAYS.

SINCERELY,

CARL WILLIAMS
V.P. OF SALES

1000 SATELLITE BOULEVARD, SUITE 104, SUWANEE, GEORGIA 30174 770-232-7744 FAX 770-232-7733

GIBRALTAR
COMPUTER
LAN-WAN-IBM-MIDRANGE-MAINFRAME

CUSTOMER: IBM ATTENTION CHRIS KING-

SALESMAN: CARL WILLIAMS DATE: 12/18/96 QUOTE #

ITEM DESCRIPTION	PART #	UNIT PRICE	QTY	TOTAL	5 YR. MAINT	TOTAL PRICE
IBM PC SERVER 704	8650-4MO	$13,696.00	1	$13,696.00	$4,800.00	$18,496.00
2-PENT. PRO 200 BOARDS				$0.00		$0.00
1-PENT. PRO 200 PROCES.				$0.00		$0.00
128MB ECC MEMORY				$0.00		$0.00
704 SMP 200 PROC. UPG.	94G6678	$2,544.00	3	$7,632.00		$7,632.00
128 MB MEMORY OPTION	94G6682	$3,989.00	16	$63,824.00		$63,824.00
PC SERVER EXP. ENCL.	3518001	$2,016.00	12	$24,192.00	$11,520.00	$35,712.00
3-METER CABLE OPT.	94G5567	$64.00	12	$768.00		$768.00
IBM HOTSWAP BCK. PLANE	70G9855	$176.00	12	$2,112.00		$2,112.00
BCK. TO BC.PLANE CABLE	94G4070	$43.00	12	$516.00		$516.00
HOT SWAP DRIVE TRAY	70G9860	$85.00	143	$12,155.00		$12,155.00
4.5GB HOT SWAP DRIVE	94G6489	$1,322.00	143	$189,046.00		$189,046.00
500 PWR. SPPLY. UPG.	70G9739	$170.00	12	$2,040.00		$2,040.00
100/10 PCI ETHER ADPT.	25H4374	$159.00	1	$159.00		$159.00
IBM G50 15" COLOR MON.	6543301	$382.00	1	$382.00	$236.00	$618.00
4/10 4MM DAT DRIVE	74G8631	$1,299.00	1	$1,299.00	$594.00	$1,893.00
				$0.00		$0.00
SOLARIS WKGP. V2.5.1	251CDBWGS	$769.00	1	$769.00	$4,200.00	$4,969.00
				$0.00		$0.00
IBM PC SERVER 320	8640-ODV	$4,382.00	6	$26,292.00	$8,100.00	$34,392.00
1 P-133MHZ 16MB MEM.				$0.00		$0.00
1-2.25GB SCSI DRIVE				$0.00		$0.00
SCSI 2 F/W PCI ADPT.				$0.00		$0.00
4X CDROM DRIVE				$0.00		
32 MB SIMMs	92G7205	$372.00	48	$17,856.00		$17,856.00
TOTALS				$362,738.00	$29,450.00	$392,188.00

1000 SATELLITE BOULEVARD, SUITE 104, SUWANEE, GEORGIA 30174 770-232-7744 FAX 770-232-7733

GIBRALTAR
COMPUTER
LAN·WAN·IBM·MIDRANGE·MAINFRAME

CUSTOMER: IBM ATTENTION CHRIS KING

SALESMAN: CARL WILLIAMS DATE: 12/18/96 QUOTE #

ITEM DESCRIPTION	PART #	UNIT PRICE	QTY	TOTAL	5 YR. MAINT	TOTAL PRICE
100/10 PCI ETHER. ADPT.	25H4374	$159.00	12	$1,908.00		$1,908.00
IBM G50 15" COLOR MON.	6543301	$382.00	6	$2,292.00	$1,416.00	$3,708.00
				$0.00		$0.00
SOLARIS WKGP. V2.5.1	251CDBWGS	$769.00	6	$4,614.00	$25,200.00	$29,814.00
PROWORKS C++ 4.01	PWXC++4.0S1	$902.00	1	$902.00	$2,846.00	$3,748.00
				$0.00		$0.00
ACCULAN 24PT. 10MBP	AL,HUB24L	$282.00	264	$74,448.00	5YR WRNTY	#VALUE!
(10% SPARE)				$0.00		$0.00
ASANTE 100MBP 12 PT	99-00352-00	$1,372.00	3	$4,116.00	5YR WRNTY	#VALUE!
(2 SPARE)				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
				$0.00		$0.00
TOTALS				$88,280.00	$29,462.00	$117,742.00

1000 SATELLITE BOULEVARD, SUITE 104, SUWANEE, GEORGIA 30174 770-232-7744 FAX 770-232-7733

BEA SYSTEMS, INC.
140 Allen Road, Rm.133
Libery Corner, NJ 07938
Phone:(908)580-3026
FAX: (908)580-3049
From: Berny Wright

Dec. 18, 1996

For:
Chris King
IBM PC Server Group
3039 Cornwallis Road
Research Triangle Pk, NC 27709

ITEM	TYPE	DESCRIPTION	QTY	UNIT PRICE	TOTAL
1	License	BEA TUXEDO v. 4.2.2	6	$1,200	$7,200
2	Service	Annual Maintenance	6	$180	$1,080
		Total License fees and maintenance *First year cost of ownership			$8,280
3	Service	Annual Maintenance Years 2-5 Pricing based on Annual Maintenance for qty 6x4years	24	$180	$4,320

Please note that this proposal is valid for the next 60 days and is subject to the e BEA Systems, Inc. terms and conditions.
Please be advised that the BEA TUXEDO /T version 4.2.2 will not be actively marketed by BEA Systems, Incorporated after 12/31/96. BEA TUXEDO version 6.1 is the current version we market.

Senior Sales Executive-Berny Wright

DPT Distributed Processing Technology 140 Candace Drive • Maitland, Florida 32751 USA • (407) 830-5522 • Fax (407) 260-5366

December 20, 1996

Chris King
IBM
3039 Cornwallis Drive
RTP, NC 27709

Reference: DPT Price Quote

Dear Ms. King;

I am providing you with pricing for the DPT Wide PCI controller and the DPT Wide expansion module, which adds an additional two channels. The warranty period is five years. Pricing is effective for 60 days and is as follows:

MODEL	DESCRIPTION	PRICE	COMMENT
PM3334W	PCI-to-Wide SCSI SmartRAID controller	$ 946	Controller only price with one channel
SX4000/2W	Wide SCSI expansion module Two channel module	$ 452	Adds two additional channels (three total)

We appreciate the opportunity to provide you with this price quote. Should you have any questions, please do not hesitate to contact me.
Sincerely,

Pete H. Schwieder
OEM Sales Manager

(408) 435-8722 San Jose, CA office phone
(408) 435-8723 San Jose, CA office fax