IBM Power 570

Using

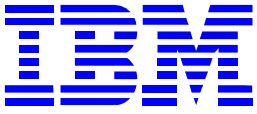
Oracle Database 10g Release 2 Enterprise Edition

and

Red Hat Enterprise Linux Advanced Platform 5 for POWER

TPC Benchmark™ C

Full Disclosure Report



Second Edition July 16, 2008

Special Notices

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

IBM System x

IBM

Oracle

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

TPC Benchmark, TPC-C, and tpmC are trademarks of the Transaction Processing Performance Council

Microsoft Windows 2003 server and COM+ are registered trademarks of Microsoft Corporation

Linux is a registered trademark of Linus Torvalds

SUSE is a registered trademark of Novell, Inc.

First Edition: October 4, 2007

Second Edition: July 16, 2008

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

TPC Benchmark Administrator
IBM Commercial Performance
Mail Stop 9571
11501 Burnet Road
Austin, TX 78758
FAX Number (512) 838-1852

IBM	IBM Power 570 Oracle 10g Release 2 Enterprise Edition	TPC-C Rev. 5.8 Report Date: October 4, 2007 Revised: July 16, 2008

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$574,615 USD	**236,271**	$2.43 USD	**April 4, 2008**

Database Processors/Cores/Threads	Database Manager	Operating System	Other Software	No. Users
4/4/8 2.2 GHz POWER5+	Oracle 10g R2	RHEL 5	Microsoft Visual C++ Microsoft COM+	189,440

Priced

8 Clients
IBM® System x3500
1x Intel 5160 DC
3.0GHz
1GB Memory
73GB Internal Drive
2 Gigabit Ethernet

IBM® System p5 570
4 x 2.2 GHz POWER5+
128GB Memory
1x36GB Internal SCSI Drives
1x73GB Internal SCSI Drives
5 x Dual Port 4GB FC
4 x Integrated GB Ethernet

Storage
8 x IBM® DS3400 Storage Servers
21x IBM® EXP3000 DiskEnclosures
304 x 73GB 15K RPM SAS Drives

System Components	Each of the 8 Clients		Server	
	Quantity	Description	Quantity	Description
Processors/Cores/ Threads	1/2/2	3.0GHz 2MB L2 Xeon 5160 Dual Core	4/4/8	2.2 GHz POWER5+
Memory	1	1GB	8	16 GB
Disk Controllers	1	SAS	2 5 7 1	Ultra 320 integrated SCSI 4Gb dual-port FC Adapters IBM DS3400 Controllers for Data IBM DS3400 Controller for Log
Disk Drives	1	73 GB	304 1 1	73GB 15K RPM SAS 36GB 15K RPM SCSI 73GB 15K RPM SCSI
Total Storage		288 GB		22,301 GB
Terminals	1	System Console	1	System Console

IBM Corporation	IBM Power 570 Oracle 10g Release 2 Enterprise Edition	TPC-C Revision 5.8 Report Date: October 4, 2007 Revised Date: July 16, 2008

Description	Part Number	Brand	Pricing Source	Unit Price	Quantity	Extended Price	3-Yr. Maint.
Server Hardware							
IBM System p5 570	9117-570	IBM	1	3,867	1	3,867	36,333
2-Way 2.2GHz POWER5+ Processor Card, 0-way active, 8 DDR2	8338	IBM	1	6,850	4	27,400	
One way Processor Activation for Processor FC 8338	7618	IBM	1	13,700	4	54,800	
Op Panel	1846	IBM	1	199	1	199	
Processor Cable	1847	IBM	1	2,647	1	2,647	
SP Flex Cable	1857	IBM	1	1,324	1	1,324	
16GB (4x4GB) DIMMS, 276 PIN, 533MHz, DDR2 SDRAM	4497	IBM	1	30,310	8	242,480	
73.4GB 15K RPM Ultra320 SCSI Disk Drive	3278	IBM	1	659	2	1,318	
4Gb Dual Port Fibre Channel	5759	IBM	1	3,308	5	16,540	
IDE Slimline DVD-ROM Drive	2640	IBM	1	274	1	274	
Processor Power Regulator	7768	IBM	1	675	6	4,050	
CEC Backplane	7865	IBM	1	1,588	2	3,176	
I/O Backplane	7866	IBM	1	5,426	2	10,852	
Midplane	7867	IBM	1	662	2	1,324	
DASD Backplane	7868	IBM	1	1,588	2	3,176	
Media Backplane	7869	IBM	1	185	1	185	
Power Midplane	7870	IBM	1	265	2	530	
System Port Riser Card	7878	IBM	1	132	2	264	
AC Power Supply, 240V, 1400W	7888	IBM	1	1,059	4	4,236	
FSP Service Processor Card	7997	IBM	1	860	1	860	
Power Cord, Drawer to IBM PDU 250V/10A	6671	IBM	1	19	4	76	
IBM Rack-mount Drawer Rail Kit	7164	IBM	1	222	2	444	
System Drawer Enclosure w/Bezel	7300	IBM	1	463	2	926	
Desktop Hardware Management Console	7310-C05	IBM	1	1,830	1	1,830	
IBM T541H/L150P 15"TFT Display	3637	IBM	1	508	1	508	
IBM Full Width USB Keyboard	5951	IBM	1	104	1	104	
IBM 3-Button Optical Mouse - Black - USB	8841	IBM	1	78	1	78	
					Subtotal	383,468	36,333
Server Storage							
IBM System Storage EXP3000	1727-HC1	IBM	1	3,199	21	67,179	
IBM 3M SAS cable	39R6531	IBM	1	135	42	5,670	
IBM EXP3000 Environmental Services Module (ESM)	39R6515	IBM	1	999	21	20,979	
IBM System Storage DS3400 Dual Controller Express	1726-42E	IBM	1	6,702	8	53,616	
IBM Hot-Swap 3.5 inch 73.4GB 15K SAS HDD	40K1043	IBM	1	309	304	93,936	
IBM S2 42U Standard Rack	93074RX	IBM	1	1,489	2	2,978	
D-LINK DGS-1024D 24-Port 10/100/1000 Switch (2 spares)	DGS1024D	compuplus	5	200	3	600	
ServicePac for 3-Year 24x7x4 Support (DS3400)	44J8073	IBM	1	1,300	8		10,400
ServicePac for 3-Year 24x7x4 Support (EXP3000)	41L2768	IBM	1	760	21		15,960
ServicePac for 3-Year 24x7x4 Support (Rack)	41L2760	IBM	1	300	2		600
					Subtotal	244,958	26,960
Server Software							
Oracle Database 10g Release 2 Enterprise Edition, Per Processor, Unlimited Users 3 years		Oracle	2	23,750	*3	71,250	
Oracle Database Server Support Package for 3 years		Oracle	2	2,300	3		6,900
Oracle Mandatory E-Business Discount		Oracle	2			(7,815)	
Red Hat Enterprise Linux Advanced Platform, Premium		Redhat	3	6,747	1	6,747	
					Subtotal	70,182	6,900
Client Hardware							
x3500 (Dual-Core Intel Xeon 5160 3.00GHz, 2x512MB RAM)	797792U	IBM	S 1	2,999	8	23,992	
73GB 15K SAS Hot-Swap Drive	40K1043	IBM	S 1	309	8	2,472	
ServicePac for 3-Year 24x7x4 Support (x3500)	96P2250	IBM	S 1	586	8		4,688
					Subtotal	26,464	4,688
Client Software							
Microsoft Windows Server 2003 Web Edition with COM+	P70-00275	CDW	6	400	8	3,200	
Visual Studio Standard 2005	127-00012	Microsoft	4	250	1	250	
Microsoft Problem Resolution Services		Microsoft	4	245	1		245
					Subtotal	3,450	245
					Total	728,522	75,126
						Total IBM Discounts*	$229,033
						Three-Year Cost of Ownership USD:	$574,615
						tpmC:	236,271
						$ USD/tpmC:	$2.43

For pricing details and contact information please see appendix D.

1) IBM 2) Oracle: MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com, 916-315-5081

* 3 = 0.75 * 4. Explanation: For the purposes of counting the number of processors which require licensing, a multicore chip with "n" cores shall be determined by multiplying "n" cores by a factor of .75.

3) Redhat, 4) Microsoft.com, 5) compuplus.com, 6) CDW.com

Substitution Notes: Eight x335s were substituted for the priced clients. 73GB drives were substituted for 36GB drives. See FDR.

*Discounts are based on US list prices for similar quantities & configurations including pre-payment for maintenance.

The discount of 35% applies to the totality of all items with price source of "1".

Audited by Francois Raab, InfoSizing, Inc. (www.sizing.com)

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.

Numerical Quantities Summary for the IBM Power 570

MQTH, computed Maximum Qualified Throughput: 236,271 tpmC

Response Times (in seconds)	90th %	Average	Maximum
New Order	1.27	0.44	7.36
Payment	1.29	0.43	7.01
Order-Status	1.28	0.44	7.04
Delivery (interactive)	0.47	0.18	5.52
Delivery (deferred)	0.08	0.04	1.70
Stock-Level	1.25	0.42	4.22
Menu	0.48	0.18	6.86

Response time delay added for emulated components was 0.1 seconds

Transaction Mix, in percent of total transactions	Percent
New Order	44.91%
Payment	43.03%
Order-Status	4.01%
Delivery	4.01%
Stock-Level	4.01%

Keying/Think Times (in seconds)	Min.	Average	Max.
New Order	18.00/0.01	18.01/12.02	18.02/120.20
Payment	3.00/0.01	3.01/12.01	3.02/120.20
Order-Status	2.00/0.01	2.01/10.01	2.02/100.10
Delivery	2.00/0.01	2.01/5.02	2.02/50.20
Stock-Level	2.00/0.01	2.01/5.02	2.02/50.20

Test Duration

Ramp-up Time	1 hour 24 minutes
Measurement interval	2 hours 30 minutes
Transactions during measurement interval (all types)	78,913,123

Checkpoints

Number of checkpoints	5
Checkpoint interval	29 minutes 36 seconds

Table of Content

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification Revision 5.8 dated December, 2006, for measurements on the IBM Power 570. The software used on the IBM Power 570 includes Red Hat Enterprise Linux Advanced Platform 5 for POWER operating system and Oracle 10g R2 data server. Microsoft COM+ is used as the transaction manager.

IBM Power 570

Company Name	System Name	Data Base Software	Operating System Software
IBM Corporation	IBM Power 570	Oracle 10g R2	Red Hat Enterprise Linux Advanced Platform 5 for POWER

Total System Cost	TPC-C Throughput	Price/Performance
• Hardware • Software • 3 Years Maintenance	Sustained maximum throughput of system running TPC-C expressed in transactions per minute	Total system cost/tpmC
$574,615 USD	236,271	$2.43 USD

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 5.8 in December 2006.

This is the full disclosure report for benchmark testing of the IBM Power 570 and Oracle 10g R2 according to the TPC Benchmark™ C Standard Specification.

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

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

This benchmark defines four on-line transactions and one deferred transaction, intended to emulate functions that are common to many OLTP applications. However, this benchmark does not reflect the entire range of OLTP requirements. The extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-C should not be used as a substitute for a specific customer application benchmarks 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 (tpmC). To be compliant with the TPC-C standard, all references to tpmC results must include the tpmC rate, the associated price-per-tpmC, and the availability date of the priced configuration.

0 General Items

0.1. Application Code Disclosure

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

Appendix A contains the application code for the five TPC Benchmark™ C transactions.

0.2. Benchmark Sponsor

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

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

0.3. Parameter Settings

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

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

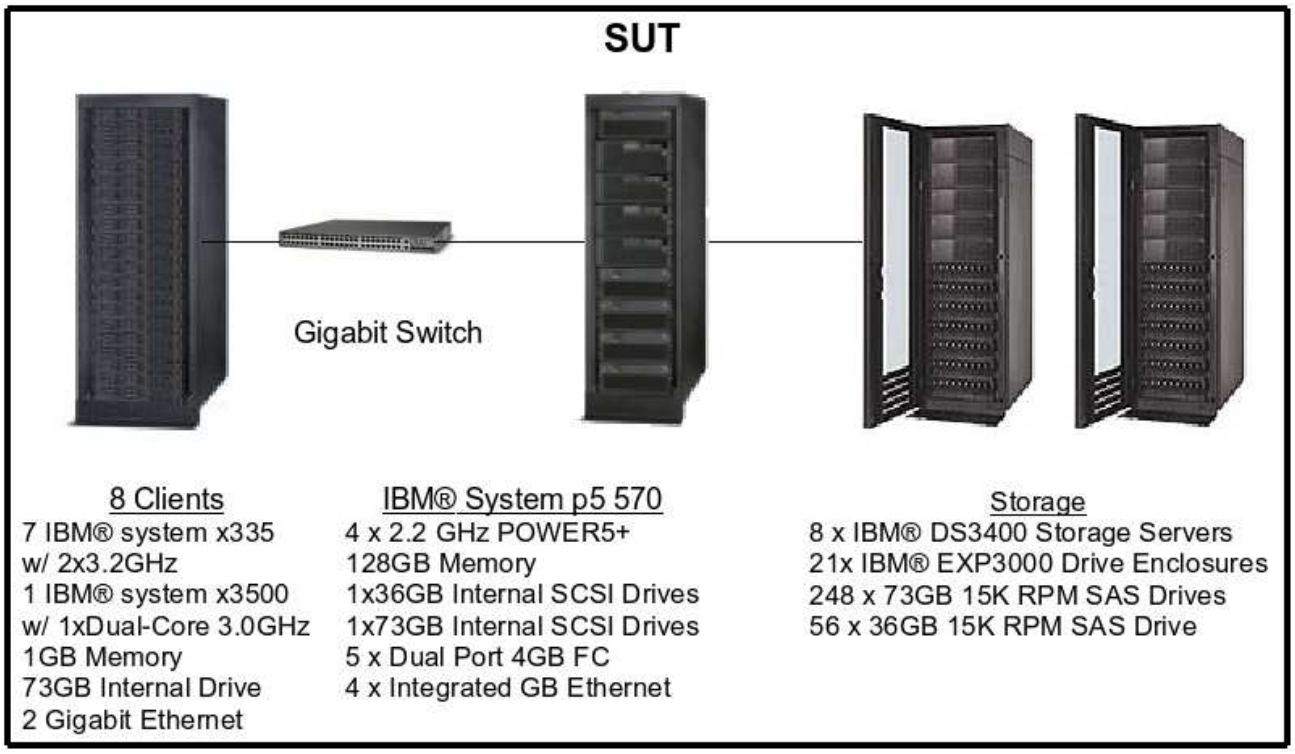
Appendix B contains the system, data base, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

0.4. Configuration Diagrams

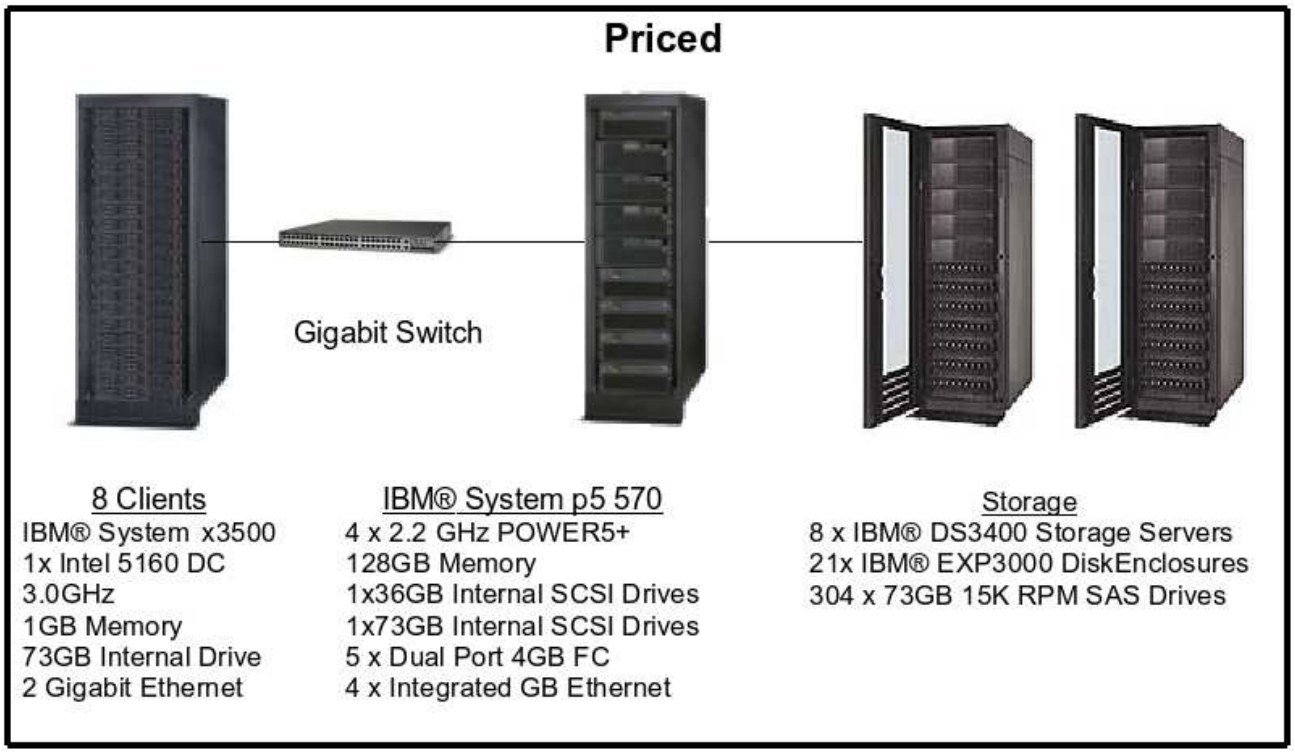
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

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

IBM Power 570 Benchmark Configuration



IBM Power 570 Priced Configuration



The measured configuration consisted of 1 x3500 client with 1 3.0GHz Intel 5160 Dual Core processors and 15 x335 clients each with two 3.2GHz Intel Xeon DP processors. However, 8 x3500 clients were priced. Based on the response times and throughput values collected from each of the clients during the run and submitted to the auditor, the auditor determined that this upgrade does not have a material effect on the reported performance.

The measured configuration consisted of 248 36GB 15K rpm SAS disk drives and 56 73GB 15K rpm SAS disk drives for data. However 304 73GB 15K rpm SAS drives were priced. The 56 73GB drives were contained in 4 logical drive arrays and the performance of these arrays were measured and compared against the same logical drives in a previous run in which all 36GB drives were used. The response times of the logical drives containing 73GB drives were equal to or better than the 36GB drives in all cases and data was submitted to the auditor.

All other components, including the disk controllers, adapters, HBAs, and the system under test, were priced as measured. For the priced configuration, see the Executive Summary.

1 Clause 1: Logical Data Base Design Related Items

1.1. Table Definitions

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

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

1.2. Database Organization

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

Physical space was allocated to Oracle on the server disks according to the details provided in Appendix C.

1.3. Insert and/or Delete Operations

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

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

The insert and delete functions were verified by the auditor. In addition, the auditor verified that the primary key for each database table could be updated outside the range of its initial partition.

1.4. Horizontal or Vertical Partitioning

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

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

2 Clause 2: Transaction & Terminal Profiles Related Items

2.1. Verification for the Random Number Generator

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

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

const double RANDOM_4_Z = 0.89837799236185

const double RANDOM_4_K = 0.97249842407114

double neg_exp_4(double average {

return ? average * (1/RANDOM_4_Z * log (1 ? RANDOM_4_K * drand48()))};

}

The seeds for each user were captured and verified by the auditor to be unique. In addition, the contents of the database were systematically searched and randomly sampled by the auditor for patterns that would indicate the random number generator had affected any kind of a discernible pattern; none were found.

2.2. Input/Output Screens

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

The screen layouts are now presented in HTML 1.0 web pages. Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3, and 2.8.3 of the TPC-C specifications were used as guidelines for html character placement.

2.3. Priced Terminal Features

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

The emulated workstations, IBM eServer x3500 systems, are commercially available and support all of the requirements in Clause 2.2.2.4.

2.4. Presentation Managers

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

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

2.5. Home and Remote Order-lines

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

Table 2-1 shows the percentage of home and remote transactions that occurred during the measurement period for the New-Order transactions.

2.6. New-Order Rollback Transactions

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

Table 2-1 shows the percentage of New-Order transactions that were rolled back due to an illegal item being entered.

2.7. Number of Items per Order

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

Table 2-1 show the average number of items ordered per New-Order transaction.

2.8. Home and Remote Payment Transactions

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

Table 2-1 shows the percentage of home and remote transactions that occurred during the measurement period for the Payment transactions.

2.9. Non-Primary Key Transactions

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

Table 2-1 shows the percentage of non-primary key accesses to the data base by the Payment and Order-Status transactions.

2.10. Skipped Delivery Transactions

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

Table 2-1 shows the percentage of Delivery transactions missed due to a shortage of supply of rows in the NEW-ORDER table.

2.11. Mix of Transaction Types

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

Table 2-1 shows the mix percentage for each of the transaction types executed by the SUT.

New Order	**IBM Power 570**
Percentage of Home order lines	99.01%
Percentage of Remote order lines	.99%
Percentage of Rolled Back Transactions	1.00%
Average Number of Items per order	9.99
Payment	
Percentage of Home transactions	85.01%
Percentage of Remote transactions	14.99%
Non-Primary Key Access	
Percentage of Payment using C_LAST	59.99%
Percentage of Order-Status using C_LAST	60.04%
Delivery	
Delivery transactions skipped	0
Transaction Mix	
New-Order	44.91%
Payment	43.03%
Order-Status	4.01%
Delivery	4.01%
Stock-Level	4.01%

Table 2-1: Numerical Quantities for Transaction and Terminal Profiles

2.12. Queuing Mechanism of Delivery

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

The Delivery transaction was submitted to an ISAPI queue that is separate from the COM+ queue that the other transactions used. This queue is serviced by a variable amount of threads that are separate from the worker threads inside the web server. Web server threads are able to complete the on-line part of the Delivery transaction and immediately return successful queuing responses to the drivers. The threads servicing the queue are responsible for completing the deferred part of the transaction asynchronously.

3 Clause 3: Transaction and System Properties

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

All ACID tests were conducted according to specification.

3.1. Atomicity Requirements

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

3.1.1. Atomicity of Completed Transaction

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

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

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

3.1.2. Atomicity of Aborted Transactions

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

The following steps were performed to verify the Atomicity of the aborted Payment transaction:

1. The Payment application code was implemented with a Perl script that allowed the transaction to be rolled back rather than committed.
2. The balance, BALANCE_3, was retrieved from the Customer table for the same Customer, District, and Warehouse combination used in the completed Payment transaction Atomicity test.
3. The Payment transaction was executed for the Customer, District and Warehouse used in step 2. Rather than commit the transaction, the transaction was rolled back.
4. The balance, BALANCE_4 was retrieved again for the Customer, District, and Warehouse combination used in step 2. It was verified that BALANCE_4 was equal to BALANCE_3, demonstrating that there were no remaining effects of the rolled back Payment transaction.

3.2. Consistency Requirements

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

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

The specification defines 12 consistency conditions of which the following four are required to be explicitly demonstrated:

1. The sum of balances (d_ytd) for all Districts within a specific Warehouse is equal to the balance (w_ytd) of that Warehouse.
2. For each District within a Warehouse, the next available Order ID (d_next_o_id) minus one is equal to the most recent Order ID [max(o_id)] for the Order table associated with the preceeding District and Warehouse. Additionally, that same relationship exists for the most recent Order ID [max(o_id)] for the New Order table associated with the same District and Warehouse. Those relationships can be illustrated as follows:

d_next_o_id – 1 = max(o_id) = max(no_o_id)

where (d_w_id = o_w_id = no_w_id) and (d_id = o_d_id = no_d_id)

3. For each District within a Warehouse, the value of the most recent Order ID [max(no_o_id)] minus the first Order ID [min(no_o_id)] plus one, for the New Order table associated with the District and Warehouse equals the number of rows in that New Order table. That relationship can be illustrated as follows:

   max(no_o_id) – min(no_o_id) + 1 = number of rows in New Order for the Warehouse/District

4. For each District within a Warehouse, the sum of Order Line counts [sum(o_ol_cnt)] for the Order table associated with the District equals the number of rows in the Order Line table associated with the same District. That relationship can be illustrated as follows:

   sum(o_ol_cnt) = number of rows in the Order Line table for the Warehouse/District

An RTE driven run was executed against a freshly loaded database. After the run the 4 consistency conditions defined above were tested using a script to issue queries to the database. All queries showed that the database was still in a consistent state.

3.3. Isolation Requirements

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

The benchmark specification defines nine tests to demonstrate the property of transaction isolation. The tests, described in Clauses 3.4.2.1 – 3.4.2.9 were all successfully executed using a series of scripts. Case D was observed during the execution of Isolation Test 7. Case B was observed during the execution of IsolationTest 8 and Test 9.

3.4. Durability Requirements

The tested system must guarantee durability: the ability to preserve the effects of committed transactions and insure data base 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)*

Failure of Log Disk, Log Fast Write Cache, Instantaneous Interruption and Memory Failure:

This test was conducted on a fully-scaled database. 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 full load test was started and allowed to run for over 6 minutes.
3. One of the disks containing the transaction log was removed. Since the log was implemented as a RAID-5 array, Oracle continued to process the transactions successfully.
4. The test continued for at least another 5 minutes.
5. A storage controller holding one copy of the mirrored write cache for the log was removed from the storage subsystem. The contents of the write cache mirrors became out-of-sync.
6. The system was subsequently powered off, which removed power from all system components, including memory.
7. The storage controller from step 5 was reinserted into the storage subsystem. The controller detected the cache out-of-sync condition and synchronized with the write cache mirror in the other controller.
8. The disk from step 3 was replaced.
9. The system was powered back on and Oracle was allowed to recover.

10. Step 1 was performed returning the value for SUM_2. It was verified that SUM_2 was greater than SUM_1 plus the completed New_Order transactions recorded by the RTE. The additional transactions found in the database were attributed to in-flight activity at the time of the failure.
11. Consistency condition 3 was verified.

Failure of Durable Medium Containing TPC-C Database Tables:

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

1. The contents of disk array "LD21", containing TPC-C database tables, was backed up in full.
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 scaled-down test was started with 12.5% of the full load.
4. A disk in array LD21 was removed, causing Oracle to report numerous errors.
5. The system was subsequently shutdown.
6. The disk was reinserted.
7. The system was powered back on.
8. Array LD21 was restored from the backup copy in step 1.
9. The system was rebooted when restore finished.
10. Oracle was restarted and the transactions in the log were applied to the database.
11. Step 2 was performed returning SUM_2. It was verified that SUM_2 was greater than SUM_1 plus the completed New_Order transactions recorded by the RTE.
12. Consistency condition 3 was verified.

4 Clause 4: Scaling and Data Base Population Related Items

4.1. Cardinality of Tables

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

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

All tables are based on 20,000 warehouses, the number of active warehouses during the benchmark was 18,944.

Table Name	Number of Rows
Warehouse	20,000
District	200,000
Customer	600000000
History	600000000
Order	600000000
New Order	180000000
Order Line	6000170528
Stock	2000000000
Item	100,000

Table 4-1: Initial Cardinality of Tables

4.2. Distribution of Tables and Logs

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

There is one Logical Disk (LD) for the logs.

There is one dual-port storage storage adapter for the log LD. Only 1 of the 2 ports are used.

The log LD is configured as a RAID5 (9+p) disk array, with 10 physical disks. Each physical disk has a capacity of 73.4 GB. The total RAID5 capacity available is 610.8 GB. The log LD was configured with 2 partitions of 305GB each. Each partition contained a logfile (log_1_3 and log_1_4).

There are 21 Logical Disks (LD) for the tables.

There are 4 dual-port storage adapters for the tables. Each adapter port is assigned 3 LDs. Only 7 of the 8 ports are used.

Each LD is configured as a RAID0 disk array, with 14 physical disks.

There are a total of 294 data disks and each physical disk has a capacity of 73.4 GB drives.

Each LD is partitioned with either 10 or 11 partitions containing a mix of datafiles, the mix of which is determined by a generated list. Each datafile is placed on a LD's partition. The next datafile in the list is placed on the next LDs partition. This process is completed on a round robin basis until the list is exhausted. Three additional partitions were added to support TPC-C audit space requirements.

Tablespaces are laid out to use LDs as follows:

Warehouse	1 tablespace using 1 partition
Warehouse Index 1	1 tablespace using 1 partition
District	1 tablespace using 2 partitions
District Index 1	1 tablespace using 1 partition
Item	1 tablespace using 1 partition
Item Index	1 tablespace using 1 partition
Stock	1 tablespace using 75 partitions
Stock Index 1	1 tablespace using 2 partitions
Customer	1 tablespace using 67 partitions
Customer Index 1	1 tablespace using 2 partitions
Customer Index 2	1 tablespace using 5 partitions
History	1 tablespace using 7 partitions
Orders	1 tablespace using 12 partitoins
Order Index	1 tablespace using 4 partitoins
New Orders	1 tablespace using 1 partition

4.3. Data Base Model Implemented

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

The database manager used for this testing was Oracle 10g R2. Oracle is a relational DBMS. Oracle remote stored procedures and embedded SQL statements were used. The Oracle stored procedures were invoked via SQL CALL statements. Both the client application and stored procedures were written in embedded C code.

4.4. Partitions/Replications Mapping

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

The specifics of the distribution of partitioned and non-partitioned tables across the physical media can be found in tables 4-2.

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

Data Distribution Logical Disks (LDs) attached to FC storage adapter			
Logical Drive-Partition #	Fiber Channel – Port	Tablespace	Data File
LD1-P1	FC1-A	ware_0	ware_0_0
LD2-P1	FC1-A	cust_0	cust_0_0
LD3-P1	FC1-A	cust_0	cust_0_1
LD4-P1	FC1-B	cust_0	cust_0_2
LD5-P1	FC1-B	cust_0	cust_0_3
LD6-P1	FC1-B	cust_0	cust_0_4
LD7-P1	FC2-A	cust_0	cust_0_5
LD8-P1	FC2-A	cust_0	cust_0_6
LD9-P1	FC2-A	cust_0	cust_0_7
LD10-P1	FC2-B	cust_0	cust_0_8
LD11-P1	FC2-B	cust_0	cust_0_9
LD12-P1	FC2-B	cust_0	cust_0_10
LD13-P1	FC3-A	cust_0	cust_0_11
LD14-P1	FC3-A	cust_0	cust_0_12
LD15-P1	FC3-A	cust_0	cust_0_13
LD16-P1	FC3-B	cust_0	cust_0_14
LD17-P1	FC3-B	cust_0	cust_0_15
LD18-P1	FC3-B	cust_0	cust_0_16
LD19-P1	FC4-A	cust_0	cust_0_17
LD20-P1	FC4-A	cust_0	cust_0_18
LD21-P1	FC4-A	cust_0	cust_0_19
LD1-P2	FC1-A	cust_0	cust_0_20
LD2-P2	FC1-A	cust_0	cust_0_21
LD3-P2	FC1-A	cust_0	cust_0_22
LD4-P2	FC1-B	cust_0	cust_0_23
LD5-P2	FC1-B	cust_0	cust_0_24
LD6-P2	FC1-B	cust_0	cust_0_25
LD7-P2	FC2-A	cust_0	cust_0_26
LD8-P2	FC2-A	cust_0	cust_0_27
LD9-P2	FC2-A	cust_0	cust_0_28

LD10-P2	FC2-B	cust_0	cust_0_29
LD11-P2	FC2-B	cust_0	cust_0_30
LD12-P2	FC2-B	cust_0	cust_0_31
LD13-P2	FC3-A	cust_0	cust_0_32
LD14-P2	FC3-A	cust_0	cust_0_33
LD15-P2	FC3-A	cust_0	cust_0_34
LD16-P2	FC3-B	cust_0	cust_0_35
LD17-P2	FC3-B	cust_0	cust_0_36
LD18-P2	FC3-B	cust_0	cust_0_37
LD19-P2	FC4-A	cust_0	cust_0_38
LD20-P2	FC4-A	cust_0	cust_0_39
LD21-P2	FC4-A	cust_0	cust_0_40
LD1-P3	FC1-A	cust_0	cust_0_41
LD2-P3	FC1-A	cust_0	cust_0_42
LD3-P3	FC1-A	cust_0	cust_0_43
LD4-P3	FC1-B	cust_0	cust_0_44
LD5-P3	FC1-B	cust_0	cust_0_45
LD6-P3	FC1-B	cust_0	cust_0_46
LD7-P3	FC2-A	cust_0	cust_0_47
LD8-P3	FC2-A	cust_0	cust_0_48
LD9-P3	FC2-A	cust_0	cust_0_49
LD10-P3	FC2-B	cust_0	cust_0_50
LD11-P3	FC2-B	cust_0	cust_0_51
LD12-P3	FC2-B	cust_0	cust_0_52
LD13-P3	FC3-A	cust_0	cust_0_53
LD14-P3	FC3-A	cust_0	cust_0_54
LD15-P3	FC3-A	cust_0	cust_0_55
LD16-P3	FC3-B	cust_0	cust_0_56
LD17-P3	FC3-B	cust_0	cust_0_57
LD18-P3	FC3-B	cust_0	cust_0_58
LD19-P3	FC4-A	cust_0	cust_0_59
LD20-P3	FC4-A	cust_0	cust_0_60
LD21-P3	FC4-A	cust_0	cust_0_61
LD1-P5	FC1-A	cust_0	cust_0_62
LD2-P5	FC1-A	cust_0	cust_0_63
LD3-P5	FC1-A	cust_0	cust_0_64
LD4-P5	FC1-B	cust_0	cust_0_65
LD5-P5	FC1-B	cust_0	cust_0_66
LD6-P5	FC1-B	dist_0	dist_0_0
LD7-P5	FC2-A	hist_0	hist_0_0
LD8-P5	FC2-A	hist_0	hist_0_1
LD9-P5	FC2-A	hist_0	hist_0_2
LD10-P5	FC2-B	hist_0	hist_0_3
LD11-P5	FC2-B	hist_0	hist_0_4
LD12-P5	FC2-B	hist_0	hist_0_5
LD13-P5	FC3-A	hist_0	hist_0_6
LD14-P5	FC3-A	stok_0	stok_0_0
LD15-P5	FC3-A	stok_0	stok_0_1
LD16-P5	FC3-B	stok_0	stok_0_2
LD17-P5	FC3-B	stok_0	stok_0_3
LD18-P5	FC3-B	stok_0	stok_0_4
LD19-P5	FC4-A	stok_0	stok_0_5
LD20-P5	FC4-A	stok_0	stok_0_6
LD21-P5	FC4-A	stok_0	stok_0_7
LD1-P6	FC1-A	stok_0	stok_0_8
LD2-P6	FC1-A	stok_0	stok_0_9
LD3-P6	FC1-A	stok_0	stok_0_10
LD4-P6	FC1-B	stok_0	stok_0_11
LD5-P6	FC1-B	stok_0	stok_0_12
LD6-P6	FC1-B	stok_0	stok_0_13
LD7-P6	FC2-A	stok_0	stok_0_14
LD8-P6	FC2-A	stok_0	stok_0_15
LD9-P6	FC2-A	stok_0	stok_0_16
LD10-P6	FC2-B	stok_0	stok_0_17
LD11-P6	FC2-B	stok_0	stok_0_18
LD12-P6	FC2-B	stok_0	stok_0_19
LD13-P6	FC3-A	stok_0	stok_0_20
LD14-P6	FC3-A	stok_0	stok_0_21
LD15-P6	FC3-A	stok_0	stok_0_22
LD16-P6	FC3-B	stok_0	stok_0_23
LD17-P6	FC3-B	stok_0	stok_0_24
LD18-P6	FC3-B	stok_0	stok_0_25
LD19-P6	FC4-A	stok_0	stok_0_26

LD20-P6	FC4-A	stok_0	stok_0_27
LD21-P6	FC4-A	stok_0	stok_0_28
LD1-P7	FC1-A	stok_0	stok_0_29
LD2-P7	FC1-A	stok_0	stok_0_30
LD3-P7	FC1-A	stok_0	stok_0_31
LD4-P7	FC1-B	stok_0	stok_0_32
LD5-P7	FC1-B	stok_0	stok_0_33
LD6-P7	FC1-B	stok_0	stok_0_34
LD7-P7	FC2-A	stok_0	stok_0_35
LD8-P7	FC2-A	stok_0	stok_0_36
LD9-P7	FC2-A	stok_0	stok_0_37
LD10-P7	FC2-B	stok_0	stok_0_38
LD11-P7	FC2-B	stok_0	stok_0_39
LD12-P7	FC2-B	stok_0	stok_0_40
LD13-P7	FC3-A	stok_0	stok_0_41
LD14-P7	FC3-A	stok_0	stok_0_42
LD15-P7	FC3-A	stok_0	stok_0_43
LD16-P7	FC3-B	stok_0	stok_0_44
LD17-P7	FC3-B	stok_0	stok_0_45
LD18-P7	FC3-B	stok_0	stok_0_46
LD19-P7	FC4-A	stok_0	stok_0_47
LD20-P7	FC4-A	stok_0	stok_0_48
LD21-P7	FC4-A	stok_0	stok_0_49
LD1-P8	FC1-A	stok_0	stok_0_50
LD2-P8	FC1-A	stok_0	stok_0_51
LD3-P8	FC1-A	stok_0	stok_0_52
LD4-P8	FC1-B	stok_0	stok_0_53
LD5-P8	FC1-B	stok_0	stok_0_54
LD6-P8	FC1-B	stok_0	stok_0_55
LD7-P8	FC2-A	stok_0	stok_0_56
LD8-P8	FC2-A	stok_0	stok_0_57
LD9-P8	FC2-A	stok_0	stok_0_58
LD10-P8	FC2-B	stok_0	stok_0_59
LD11-P8	FC2-B	stok_0	stok_0_60
LD12-P8	FC2-B	stok_0	stok_0_61
LD13-P8	FC3-A	stok_0	stok_0_62
LD14-P8	FC3-A	stok_0	stok_0_63
LD15-P8	FC3-A	stok_0	stok_0_64
LD16-P8	FC3-B	stok_0	stok_0_65
LD17-P8	FC3-B	stok_0	stok_0_66
LD18-P8	FC3-B	stok_0	stok_0_67
LD19-P8	FC4-A	stok_0	stok_0_68
LD20-P8	FC4-A	stok_0	stok_0_69
LD21-P8	FC4-A	stok_0	stok_0_70
LD1-P9	FC1-A	stok_0	stok_0_71
LD2-P9	FC1-A	stok_0	stok_0_72
LD3-P9	FC1-A	stok_0	stok_0_73
LD4-P9	FC1-B	stok_0	stok_0_74
LD5-P9	FC1-B	item_0	item_0_0
LD6-P9	FC1-B	ordr_0	ordr_0_0
LD7-P9	FC2-A	ordr_0	ordr_0_1
LD8-P9	FC2-A	ordr_0	ordr_0_2
LD9-P9	FC2-A	ordr_0	ordr_0_3
LD10-P9	FC2-B	ordr_0	ordr_0_4
LD11-P9	FC2-B	ordr_0	ordr_0_5
LD12-P9	FC2-B	ordr_0	ordr_0_6
LD13-P9	FC3-A	ordr_0	ordr_0_7
LD14-P9	FC3-A	ordr_0	ordr_0_8
LD15-P9	FC3-A	ordr_0	ordr_0_9
LD16-P9	FC3-B	ordr_0	ordr_0_10
LD17-P9	FC3-B	ordr_0	ordr_0_11
LD18-P9	FC3-B	nord_0	nord_0_0
LD19-P9	FC4-A	iware_0	iware_0_0
LD20-P9	FC4-A	icust1_0	icust1_0_0
LD21-P9	FC4-A	icust2_0	icust2_0_0
LD1-P10	FC1-A	icust2_0	icust2_0_1
LD2-P10	FC1-A	icust2_0	icust2_0_2
LD3-P10	FC1-A	icust2_0	icust2_0_3
LD4-P10	FC1-B	icust2_0	icust2_0_4
LD5-P10	FC1-B	idist_0	idist_0_0
LD6-P10	FC1-B	istok_0	istok_0_0
LD7-P10	FC2-A	iitem_0	iitem_0_0
LD8-P10	FC2-A	iordr2_0	iordr2_0_0

LD9-P10	FC2-A	iordr2_0	iordr2_0_1
LD10-P10	FC2-B	iordr2_0	iordr2_0_2
LD11-P10	FC2-B	iordr2_0	iordr2_0_3
LD12-P10	FC2-B	temp_0	temp_0_0
LD13-P10	FC3-A	temp_0	temp_0_1
LD14-P10	FC3-A	temp_0	temp_0_2
LD15-P10	FC3-A	temp_0	temp_0_3
LD16-P10	FC3-B	temp_0	temp_0_4
LD17-P10	FC3-B	temp_0	temp_0_5
LD18-P10	FC3-B	temp_0	temp_0_6
LD19-P10	FC4-A	temp_0	temp_0_7
LD20-P10	FC4-A	temp_0	temp_0_8
LD21-P10	FC4-A	temp_0	temp_0_9
LD1-P11	FC1-A	temp_0	temp_0_10
LD2-P11	FC1-A	temp_0	temp_0_11
LD3-P11	FC1-A	temp_0	temp_0_12
LD4-P11	FC1-B	temp_0	temp_0_13
LD5-P11	FC1-B	tpccaux	tpccaux
LD6-P11	FC1-B	control	control_001
LD7-P11	FC2-A	control	control_002
LD8-P11	FC2-A	sp	sp_0
LD9-P11	FC2-A		log_1_1
LD10-P11	FC2-B		log_1_2
LD11-P11	FC2-B	system	system_1
LD12-P11	FC2-B	roll1	roll1
LD13-P11	FC3-A	dist_0	dist_0_1
LD14-P11	FC3-A	icust1_0	icust1_0_1
LD15-P11	FC3-A	istok_0	istok_0_1

Table 4-2: IBM Power 570 Data Distribution Benchmark Configuration

4.5. 60-Day Space Calculations

Details of the 60 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

60 Day Space Calculations

TPM	236272	new_order	50975557
Warehouses	20000	Redo blocks written	533962296

SEGMENT	TYPE	BLOCKS	BLOCK SIZE	FIVE PCT	DAILY GROW	TOTAL (MB)
CUSTCLUSTER	CLUSTER	210251232	2048	10512562	0	431179.28
DB_STAT	SYS	1048576	2048	0	0	2048.00
DISTCLUSTER	CLUSTER	214512	2048	10726	0	439.92
HIST	TABLE	18456549	2048	0	3488613	42861.64
ICUST1	INDEX	1003017	16384	50151	0	16455.75
ICUST2	INDEX	16233932	2048	811697	0	33292.24
IDIST	INDEX	50512	2048	2526	0	103.59
IITEM	INDEX	5632	2048	282	0	11.55
IORDR2	INDEX	12910000	2048	645500	0	26475.59
ISTOCK	INDEX	2670784	16384	133539	0	43817.55
ITEMCLUSTER	CLUSTER	8446	2048	422	0	17.32
IWARE	INDEX	13012	2048	651	0	26.68
NORDCLUSTER_QUEUE	CLUSTER	2500253	2048	125013	0	5127.47
ORDRCLUSTER_QUEUE	CLUSTER	30484740	16384	0	5762152	566357.69
STOKCLUSTER	CLUSTER	286015296	2048	14300765	0	586554.81
SYSAUX	SYS	61440	2048	0	0	120.00
SYSTEM	SYS	204800	2048	0	0	400.00
SYS_IQ000009745$$	SYS	129200	16384	6460	0	2119.69
SYS_IQ000009749$$	SYS	357179	2048	17859	0	732.50
WARECLUSTER	CLUSTER	21912	2048	1096	0	44.94
Total		582,641,024		26,619,246	9,250,765	1,758,186.20

Dynamic space	512,372	Initial MB for (History+Order
Static space	1,148,967	Initial blocks +5% -Dynamic
Daily growth	96,847	
Daily spread	0	Oracle may be configured such that daily spread is 0
60-day space (MB)	6,959,806	
60-day (GB)	6,796.69	
Log block size	512	
Log blocks/tpmC	10.47	Number of log blocks used in one tpmC
8-hour log (GB)	566.46	

Disk Type	Formatted Capacity (GB)	SUT No of arrays	Capacity(GB)		
				Data	19952.52
				Log	610.79
Data RAID0(14-73GB)	950.12	21	19952.52	OS	102.27
Log RAID5(10-73GB)	610.79	1	610.79	**Total Space**	**20665.58**

5 Clause 5: Performance Metrics and Response Time Related Items

5.1. Response Times

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

Table 5-1 lists the response times and the ninetieth percentiles for each of the transaction types for the measured system.

5.2. Keying and Think Times

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

Table 5-1 lists the TPC-C keying and think times for the measured system.

Response Times	New Order	Payment	Order Status	Delivery (int./def.)	Stock Level	Menus
90 %	1.27	1.29	1.28	0.47/0.08	1.25	0.48
Average	0.44	0.43	0.44	0.18/0.04	0.42	0.18
Maximum	7.36	7.01	7.04	5.52/1.70	4.22	6.68
Think Times						
Minimum	0.01	0.01	0.01	0.01	0.01	N/A
Average	12.02	12.01	10.01	5.02	5.02	N/A
Maximum	120.20	120.20	100.10	50.20	50.20	N/A
Keying Times						
Minimum	18.00	3.00	2.00	2.00	2.00	N/A
Average	18.01	3.01	2.01	2.01	2.01	N/A
Maximum	18.02	3.02	2.02	2.02	2.02	N/A

Table 5-1: Think and Keying Times

5.3. Response Time Frequency Distribution

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

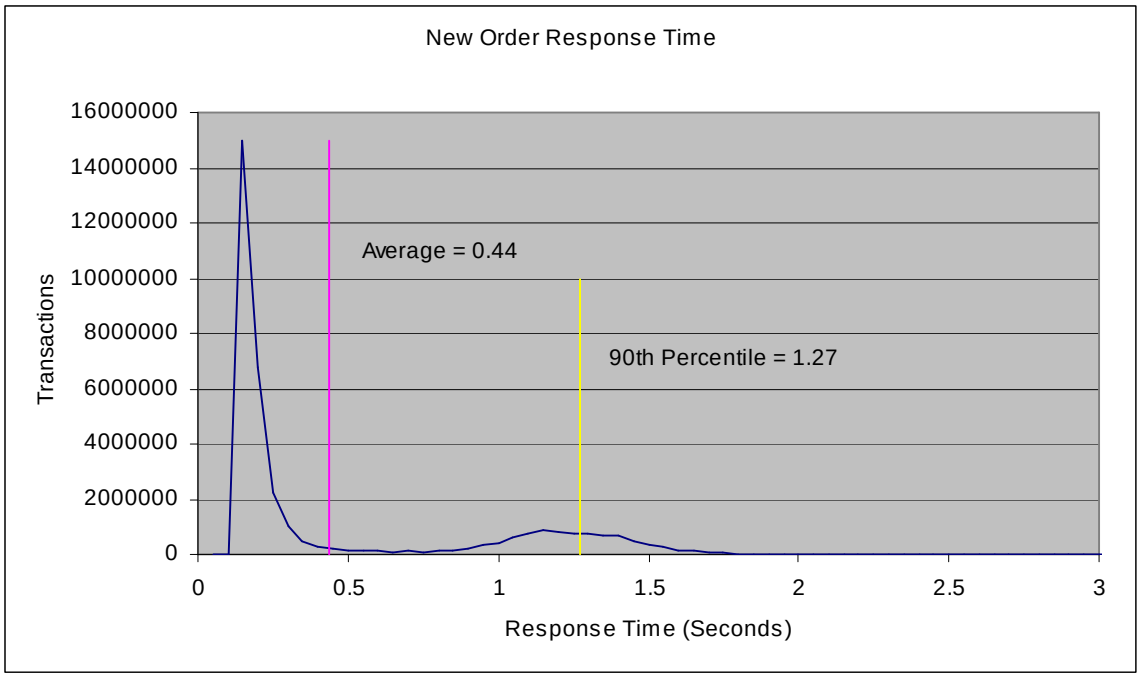


Figure 5-1: New-Order Response Time Distribution

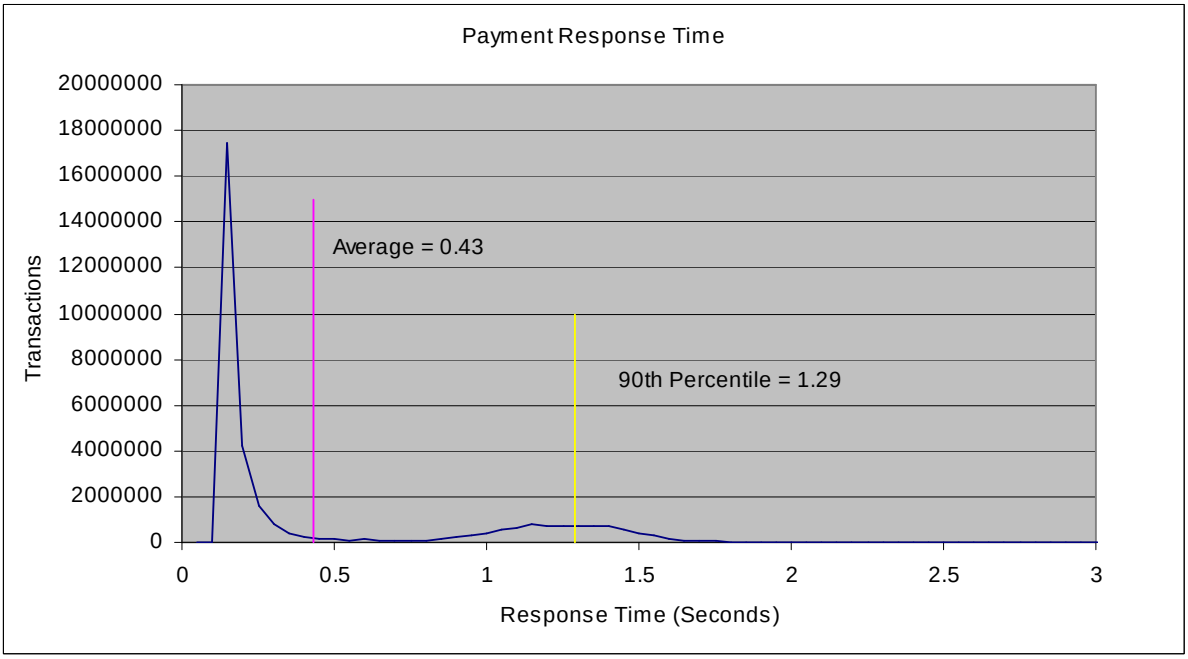


Figure 5-2: Payment Response Time Distribution

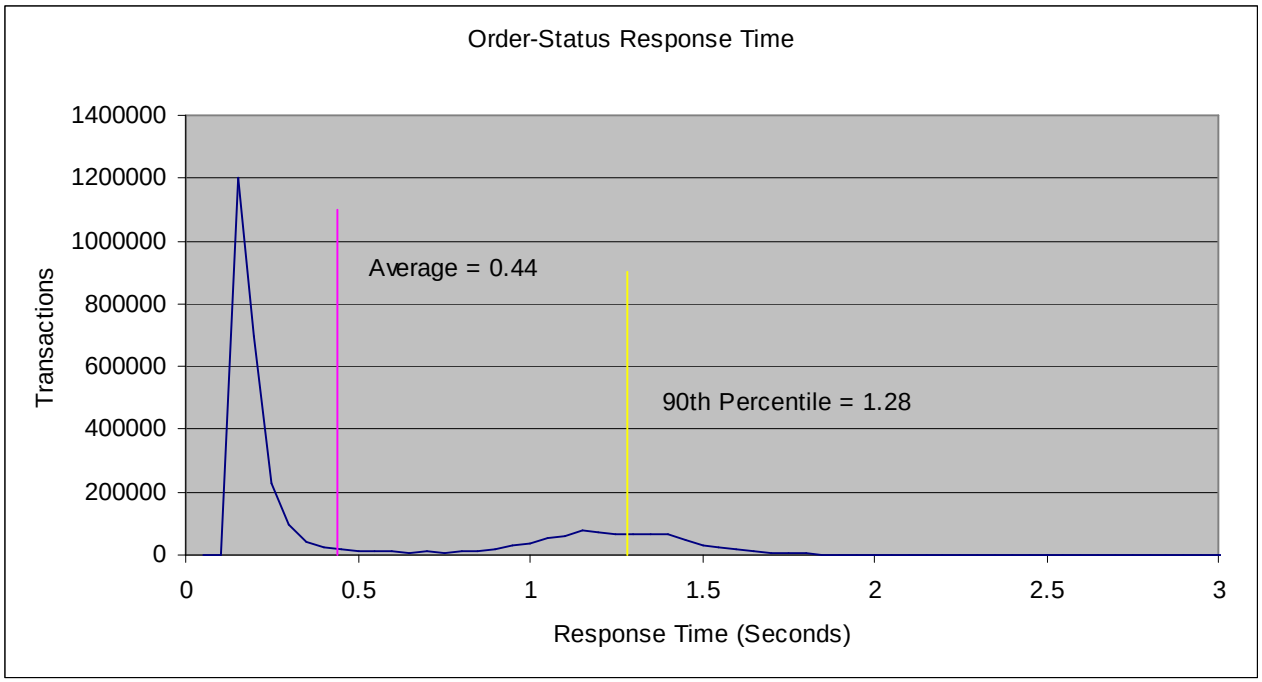


Figure 5-3: Order-Status Response Time Distribution

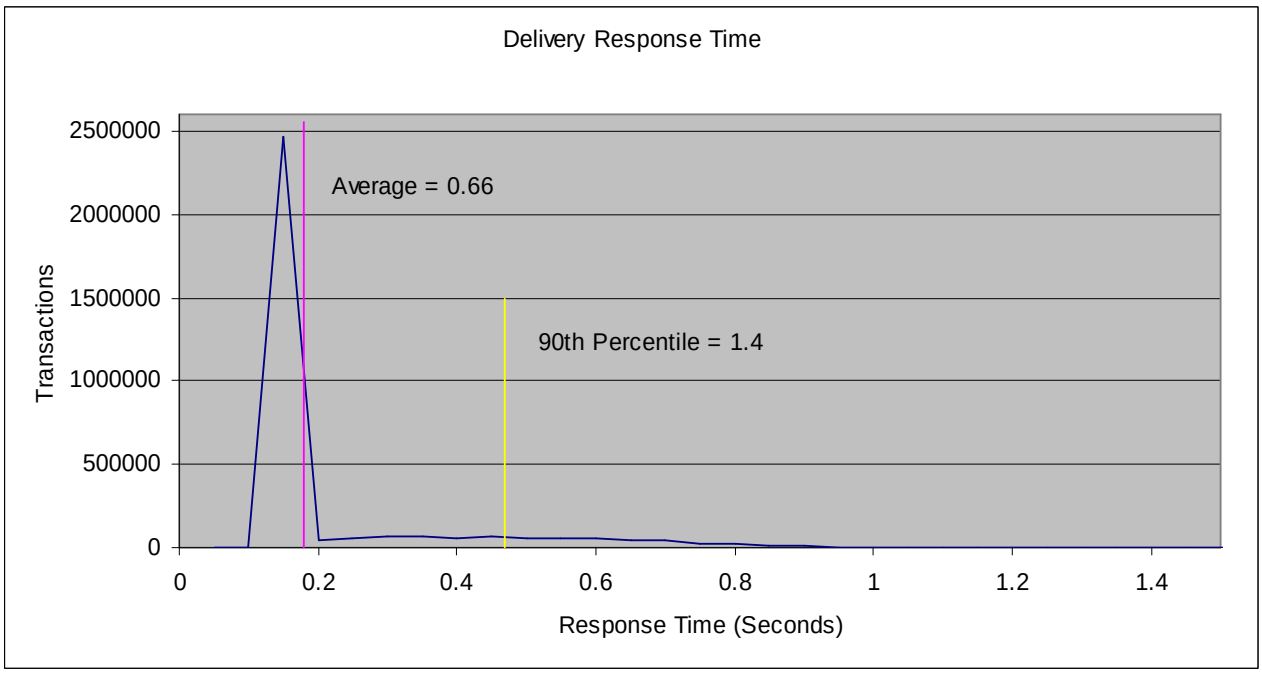


Figure 5-4: Delivery (Interactive) Response Time Distribution

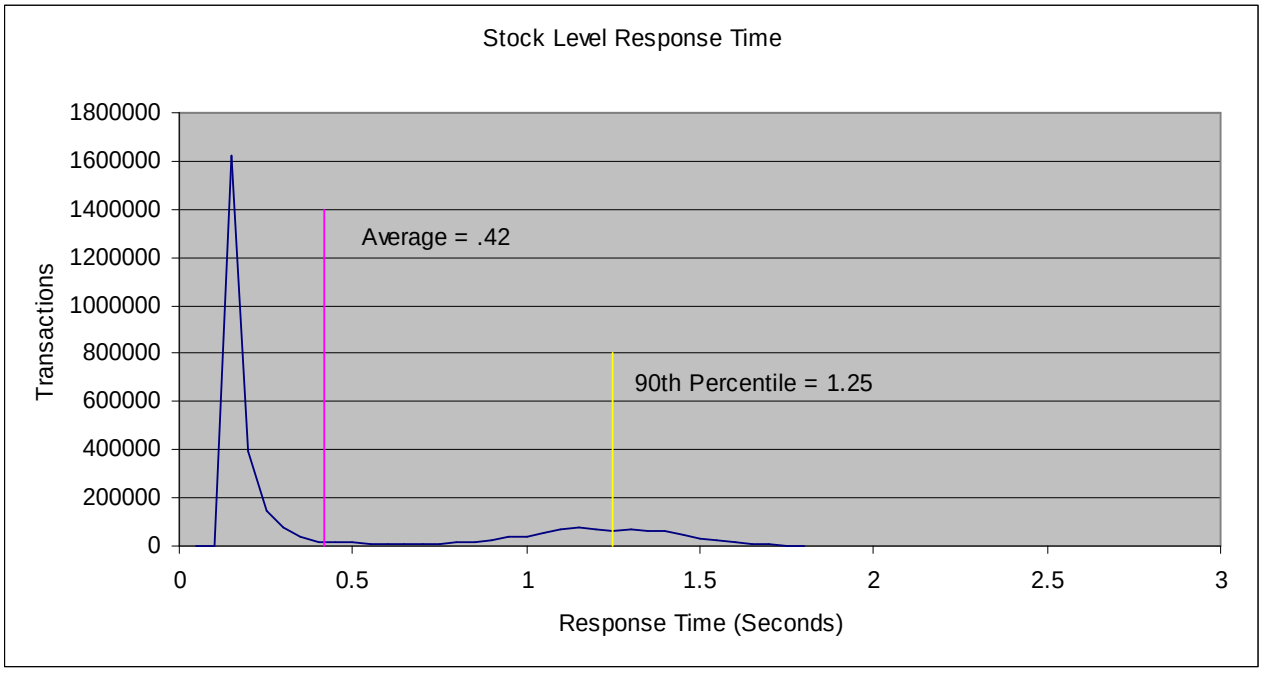


Figure 5-5: Stock Level Response Time Distribution

5.4. Performance Curve for Response Time versus Throughput

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

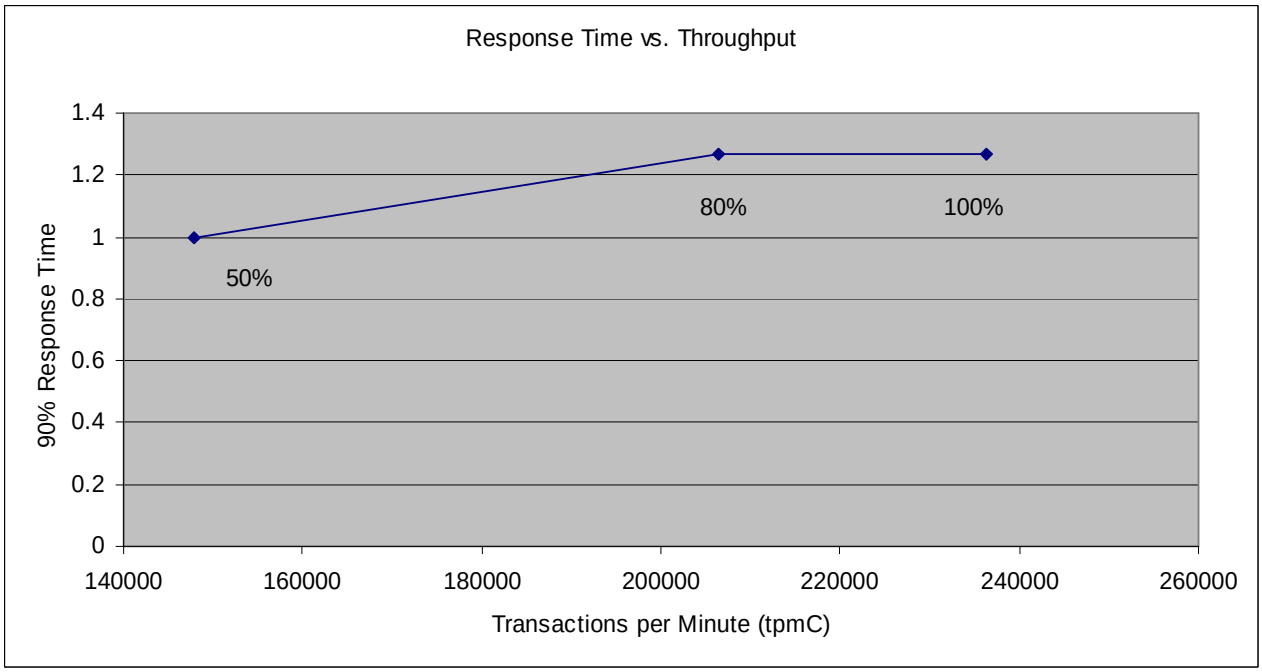


Figure 5-6: New-Order Response Time vs. Throughput

5.5. Think Time Frequency Distribution

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

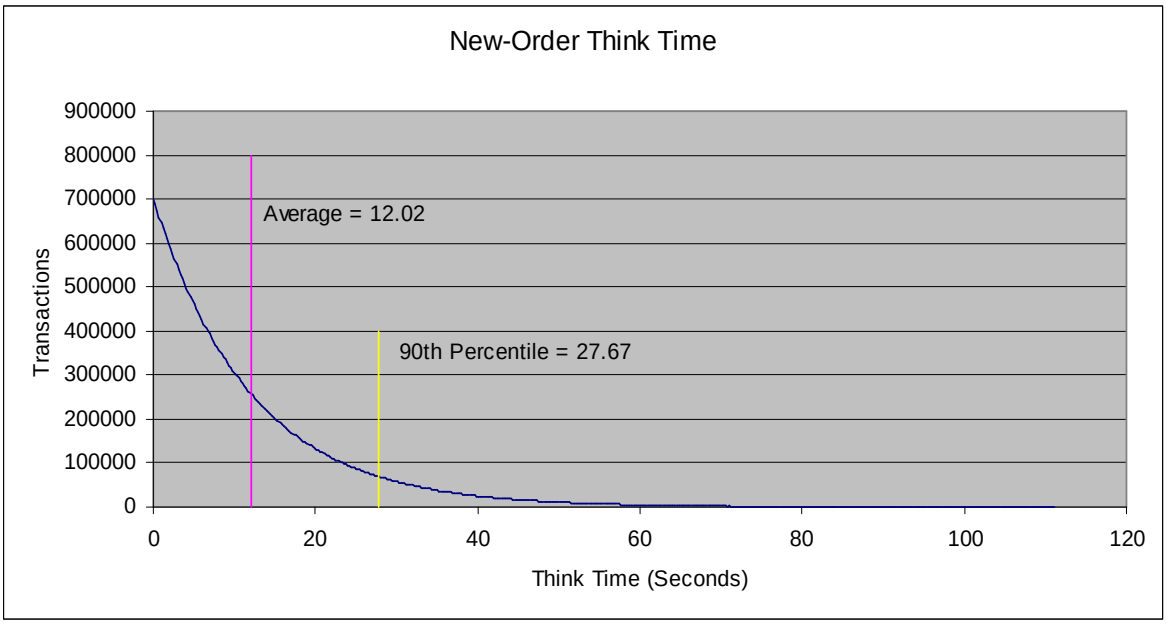


Figure 5-7: New-Order Think Time Distribution

5.6. Throughput versus Elapsed Time

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

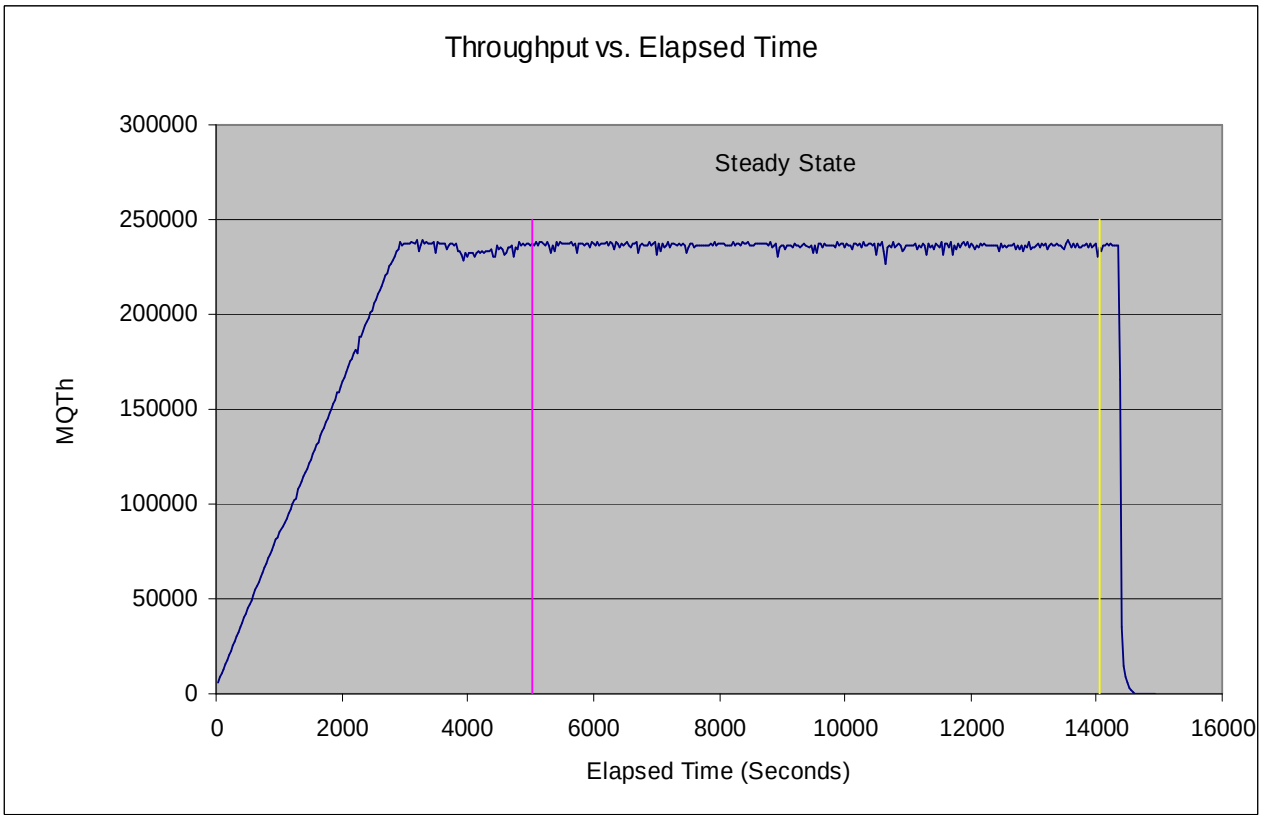


Figure 5-8: New-Order Throughput vs. Elapsed Time

5.7. Steady State Determination

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

All the emulated users were allowed to logon and do transactions. The user ramp-up phase is clearly visible on the graph above. Refer to the Numerical Quantities Summary pages for the rampup time. Figure 5-8 New-Order throughput versus Elapsed Time graph shows that the system maintained a steady state during the measurement interval

5.8. Work Performed During Steady State

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

A 2-hour 30 minute measurement interval was used to guarantee that all work normally performed during an 8-hour sustained test is included in the reported throughput.

5.8.1. Transaction Flow

Each of the 4 (non-delivery) transactions is serviced by 2 individual programs, Internet Information System 6.0 (IIS) and a Microsoft COM+ 1.0 Queued Component Server, used as the transaction manager (COM+). Both programs are running on the client system:

- The initial HTML 1.0 request is serviced by an ISAPI custom-written handler running on Internet Information System 6.0. IIS is responsible for handling all HTML requests. The web server communicates to the COM+ server through a Microsoft COM+ API interface.
- COM+ communicates with the server system over Ethernet and handles all database operations**,** using Oracle embedded SQL calls.
  When the COM+ server boots up, it creates a configurable amount of connections to the server (listed in application settings).
  COM+ routes the transaction and balances the load according to the options defined in the Component Services GUI for the COM+ server application and settings in the Windows 2003 registry. The configuration file and registry variables are listed in Appendix B.2.
  At the beginning, each TPC-C user sends a pair of HTML 1.0 requests submitting its unique warehouse and district to the IIS ISAPI handler. Upon successful validation of user's login, IIS the displays an HTML form which encapsulates the TPC-C transaction menu.
  The transaction flow is described below:
  - The TPC-C user requests the transaction type's HTML form and proceeds to generate (fill in) a GET request with the required files for the transaction.
  - IIS accepts the filled in GET request , parses, and validates all values entered by the user.
  - It then proceeds to transmit those values to the COM+ server through an transaction type specific COM+ api interface.
  - The COM+ Pool Manager receives the request and first decides if there is a connection object in the pool available to service it.
    If so, the connection is used to send the transaction request to the Server.
    If no connection is available, the request will enter a COM+ internal queue and will be serviced by the next available connection.
  - Once the connection is available to be used, a COM+ pool thread receives the transaction and calls a TPC-C back end Oracle client api to execute all database operations related to the transaction type. (All the transaction information entered on the HTML form is available in a data structure provided by the ISAPI caller).
  - The transaction is committed and the Oracle back end client returns control back to the COM pool thread.
  - COM pool thread returns control to the ISAPI caller.
    (All transaction results are inside the data structure that the ISAPI caller provided to the COM+ api in the parameter list).
  - ISAPI caller returns control to the "screen application" by doing a PUT request.

5.8.2. Database Transaction

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

Using SQL*Net calls, the TPC-C back-end program interacts with Oracle 10g Server to perform SQL data manipulations such as update, select, delete and insert, as required by the transaction. After all database operations are performed for a transaction, the transaction is committed.

Oracle 10g Server proceeds to update the database as follows:

When Oracle 10g Server changes a database table with an update, insert, or delete operation, the change is initially made in memory, not on disk. When there is not enough space in the memory buffer to read in or write additional data pages, Oracle 10g Server will make space by flushing some modified pages to disk. Modified pages are also written to disk when a checkpoint occurs. Before a change is made to the database, it is first recorded in the transaction log. This ensures that the database can be recovered completely in the event of a failure. Using the transaction log, transactions that started but did not complete prior to a failure can be undone, and transactions recorded as complete in the transaction log but not yet written to disk can be redone.

5.8.3. Checkpoints

A checkpoint is the process of writing all modified data pages to disk. The TPC-C benchmark was setup to automatically checkpoint every 30 minutes. Three checkpoints occurs during the rampup period, and five other occurring during the measurement interval. The incremental checkpoint duration during the measurement is 29 minutes and 30 second.

Checkpoint	Start Time	End Time	Duration
	13:50:01	Run Begins	
#0	13:47:29	14:17:05	29:36
#1	14:17:05	14:46:40	29:35
#2	14:46:40	15:16:17	29:37
	15:24:02	Measurement Interval Begins	
#3	15:45:53	16:15:30	29:37
#4	16:15:30	16:45:04	29:34
#5	16:45:04	17:14:40	29:36
#6	17:14:40	17:44:17	29:37
#7	17:44:17		
	17:54:02	Measurement Interval Ends	

5.9. Measurement Interval

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

A 2-hour 30 minute measurement interval was used. No connections were lost during the run.

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

6.1. RTE Availability

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

IBM used an internally developed RTE for these tests. 20,000 warehouses were configured. A rampup time of one hour twenty four minutes was specified, along with a run time of two hours thirty minutes.

6.2. Functionality and Performance of Emulated Components

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

No components were emulated.

6.3. Network Bandwidth

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

The network between the clients and the database server was configured as 1000 MegaBits per second Full Duplex.

6.4. Operator Intervention

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

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

7 Clause 7: Pricing Related Items

7.1. Hardware and Programs Used

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

The detailed list of all hardware and programs for the priced configuration is listed in the pricing sheets for each system reported. The prices for all products and features that are provided by IBM are available the same day as product or feature availability.

7.2. Three Year Cost of System Configuration

The total 3-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

The pricing details for this disclosure is contained in the executive summary pages. All 3rd party quotations are included at the end of this report in Appendix D. All prices are based on IBM US list prices.

Discount are based on US list prices and for similar quantities and configurations. A discount of 35% has been applied to specified IBM hardware, and services based on the total value and quantities of the components of the configuration.

7.3. Availability Dates

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

All components of the SUT will be available on April 4, 2008.

7.4. Statement of tpmC and Price/Performance

A statement of the measured tpmC, as well as the respective calculations for 3-year pricing, price/performance (price/tpmC), and the availability date must be disclosed.

System	tpmC	3-year System Cost	$/tpmC	Availability Date
IBM Power 570	236,271	$574,615 USD	$2.43 USD	April 4, 2008

Please refer to the price list on the Executive Summary page for details.

8 Clause 9: Audit Related Items

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

The auditor's attestation letter is included in this section of this report:

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Sponsors:

Raymond J. Venditti
IBM Linux Performance
11501 Burnet Road
Austin, TX 78758

May 7, 2007

I verified the TPC Benchmark™ C performance of the following Client Server configuration:

Platform: IBM System p5 570 c/s
Operating system: RHEL 5
Database Manager: Oracle 10G
Transaction Manager: Microsoft COM+

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: IBM System p5 570				
4 x POWER5+ (2.2 GHz)	128 GB main	1 x 36 GB SCSI int. 304 x 73 GB SAS 1 x 73 GB SCSI int.	1.27 Sec.	236,271.7
8 Clients: IBM System x3500 (Specification for each)				
1 x Intel Xeon 5160 Dual-Core (3.0 GHz)	1 GB main (2 MB L2 cache per Core)	1 x 73 GB	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for the current revision (5.7) of the benchmark.

125 West Monroe Street • Colorado Springs, CO 80907 • 719-473-7555 • www.sizing.com

The following verification items were given special attention:

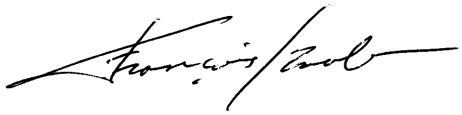
- The transactions were correctly implemented.
- The database records were the proper size.
- The database was properly scaled and populated.
- The ACID properties were met.
- Input data was generated according to the specified percentages.
- The transaction cycle times included the required browser delay, keying and think times.
- The reported response times were correctly measured.
- At least 90% of all delivery transactions met the 80 Second completion time limit.
- All 90% response times were under the specified maximums.
- The measurement interval was representative of steady state conditions.
- The reported measurement interval was 2.5 hours.
- Five checkpoints were taken during the measurement interval.
- The 60 day storage requirement was correctly computed.
- The system pricing was verified for major components and maintenance.

Additional Audit Notes:

The measured configuration included 7 IBM x335 client systems and one IBM x3500 client system. The IBM x335 systems were substituted, one for one, with IBM x3500 systems in the priced configuration. The TPC requirements for substitution of priced clients were verified and met.

The measured configuration included 248 36GB 15Krpm SAS disk drives and 56 73GB 15Krpm SAS disk drives as data drives. The 36GB disk drives were substituted, one for one, with 73GB disk drives in the priced configuration. The TPC requirements for substitution of priced disks were verified and met.

Respectfully Yours,



François Raab, President

125 West Monroe Street • Colorado Springs, CO 80907 • 719-473-7555 • www.sizing.com

9 Appendix A: Client Server Code

9.1. Client/Terminal Handler Code

tpccIsapi.def

```
; tpccIsapi.def : declares the module parameters for the DLL.

LIBRARY      "tpccIsapi"

EXPORTS
    HttpExtensionProc
    GetExtensionVersion
    TerminateExtension
```

tpcc.h

```
// Common defines and structures use internally by client code
// Not to be confused with structures actually passed in transaxtions
//

// standard includes

#ifndef _COMMON_TPCC
#define _COMMON_TPCC

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/timeb.h>
#include <time.h>

#ifdef DB2
    #include <db2tpcc.h>
#endif

#ifdef ORACLE
    #include "ora_tpcc.h"
#endif

#ifdef SYBASE
    #include <sybtpcc.h>
#endif


#include <iostream>
#include <fstream>
#include <process.h>
#include <ios>

////////////////////////////////////////////////////////
// Defines
////////////////////////////////////////////////////////

#define     OK                  0
#define     INVALID_STATUS      -1
#define     ERR                 -1
#define     INVALID_COM_STATUS      -2


#define TXN_MAX_COMMANDS        55
#define MAX_TRANSACTIONS        14
#define MAX_CMD_LENGTH          100
#define INPUT_ITEMS             3
#define MAX_INT_BUFFER          15
#define NORD_ITEMS              15
#define ITEM_START              11
#define ITEM_END                55
#define MAX_ITEMS               15

#define MAX_STRING_LEN          256
#define MAX_HTML_PAGE_LEN       4096
#define MAX_HTML_HEADER_LEN     512

#define DELIVERY_THREADS_NUM    100

#define DISTRICTS_PER_WAREHOUSE     10
////////////////////////////////////////////////////////
// Transaction Codes
////////////////////////////////////////////////////////

#define     TXN_LOGIN               0
#define     TXN_NEW_ORDER           1
#define     TXN_PAYMENT             2
#define     TXN_ORDER_STATUS        3
#define     TXN_DELIVERY            4
#define     TXN_STOCK               5
#define     TXN_EXIT            6
#define     TXN_LOGIN_RESULTS       7
#define     TXN_NEW_ORDER_RESULTS   8
#define     TXN_PAYMENT_RESULTS     9
#define     TXN_ORDER_STATUS_RESULTS    10
#define     TXN_DELIVERY_RESULTS    11
#define     TXN_STOCK_RESULTS       12

#define     CMD_NORD            "nord"
#define     CMD_PYMT            "pymt"
#define     CMD_ORDS            "ords"
#define     CMD_DLVY            "dlvy"
#define     CMD_STOK            "stok"
#define     CMD_EXIT            "exit"
#define     CMD_MENU            "menu"

#define     APP_NAME            "tpcc.html"
#define     HEADER              "Content-Type:text/html\r\nContent-Length:
%d\r\nConnection: Keep-Alive\r\n\r\n"

////////////////////////////////////////////////////////
// URL Commands
////////////////////////////////////////////////////////

#define     CMD_TXN_ID          "00"
#define     CMD_TERM_ID         "01"
#define     CMD_W_ID            "02"
#define     CMD_D_ID            "03"
#define     CMD_C_ID            "04"
#define     CMD_C_NAME          "05"
#define     CMD_C_W_ID          "06"
#define     CMD_C_D_ID          "07"
#define     CMD_AMT_PAID        "08"
#define     CMD_STK_THRESHOLD   "09"
#define     CMD_CARRIER_NUM     "10"

#define     ITEM01_SUPP_W       "11"
#define     ITEM01_ITEM_NUM     "12"
#define     ITEM01_OTY          "13"


#define CHAR_FILL           ' '
#define NUMERIC_FILL        ' '
#define NEGITIVE_SYMBOL         '-'
#define MONEY_SYMBOL        '$'
#define DECIMAL_SYMBOL          '.'
#define ZERO_SYMBOL             '0'
#define ZIP_DELIMITER       '-'
#define PHONE_DELIMITER         '-'
#define DATE_DELIMITER          '-'
#define TIME_DELIMITER      ':'

#define DEFAULT_MONEY64_LEN     15
#define DEFAULT_MONEY32_LEN     9
#define DEFAULT_MONEY16_LEN     9

#define DEFAULT_NUMERIC64_LEN   15
#define DEFAULT_NUMERIC32_LEN   9
#define DEFAULT_NUMERIC16_LEN   9

#define DEFAULT_DECIMAL64_LEN   5
#define DEFAULT_DECIMAL32_LEN   5
#define DEFAULT_DECIMAL16_LEN   5

#define DEFAULT_DATETIME_LEN    19
#define DEFAULT_DATE_LEN        11
#define DEFAULT_TIME_LEN        8

#define DEFAULT_STRING_LEN      25
#define DEFAULT_ZIP_LEN         17
#define DEFAULT_PHONE_LEN       18

////////////////////////////////////////////////////////
// String Field Lengths
////////////////////////////////////////////////////////

#define NAME_LEN        24
#define LAST_NAME_LEN       16
#define FIRST_NAME_LEN      16
#define INITIALS_LEN    2

#define CREDIT_LEN          2

#define STREET_LEN          20
#define CITY_LEN        20
#define STATE_LEN       2
#define ZIP_LEN             9

#define PHONE_LEN           16
#define DATA_LEN            200

#define ITEM_LIST           15
#define ORDER_LIST          10

////////////////////////////////////////////////////////
// Type definitions
////////////////////////////////////////////////////////

typedef __int8              INT8b;
typedef __int16             INT16b;
typedef __int32             INT32b;
typedef __int64             INT64b;

typedef unsigned __int8     UINT8b;
typedef unsigned __int16    UINT16b;
typedef unsigned __int32    UINT32b;
typedef unsigned __int64    UINT64b;

typedef INT16b              sqlint16;
typedef INT32b              sqlint32;
typedef INT64b              sqlint64;

typedef INT16b              int16_t;
typedef INT32b              int32_t;
```

```
typedef INT64b              int64_t;

typedef char              BYTE8b;
typedef double              DOUBLE;
typedef unsigned long       NATURAL;

////////////////////////////////////////////////////////////////
// Date and time values
////////////////////////////////////////////////////////////////

#define SECONDS_IN_DAY      86400
#define SECONDS_IN_HOUR     3600
#define SECONDS_IN_MINUTE   60
#define GMT_OFFSET          5

#define DAYS_IN_YEAR        365
#define YEARS_IN_LEAP       4
#define START_YEAR          1970
#define MONTHS_IN_YEAR      12

////////////////////////////////////////////////////////////////
// Error codes
////////////////////////////////////////////////////////////////
#define ERR_INVALID_TXN_TYPE          -1

#define ERR_MISSING_W_ID              -2
#define ERR_NON_NUMERIC_W_ID          -3
#define ERR_MISSING_D_ID              -4
#define ERR_NON_NUMERIC_D_ID          -5
#define ERR_MISSING_C_ID              -6
#define ERR_NON_NUMERIC_C_ID          -7

#define ERR_MISSING_SUPP_W            -8
#define ERR_NON_NUMERIC_SUPP_W        -9
#define ERR_MISSING_ITEM_NUM          -10
#define ERR_NON_NUMERIC_ITEM_NUM      -11
#define ERR_MISSING_ITEM_OTY          -12
#define ERR_NON_NUMERIC_ITEM_QTY      -13

#define ERR_MISSING_CLAST_NAME        -14
#define ERR_NON_NUMERIC_CUST_W_ID     -15
#define ERR_NON_NUMERIC_CUST_D_ID     -16
#define ERR_MISSING_AMOUNT_PAID       -17
#define ERR_NON_NUMERIC_AMOUNT_PAID   -18

#define ERR_INVALID_D_ID        "ERROR : Invalid District ID. Try Again."
#define ERR_INVALID_W_ID        "ERROR : Invalid Warehouse ID. Try Again."
#define ERR_INVALID_C_ID        "ERROR : Invalid Customer ID. Try Again."
#define ERR_INVALID_SUPPLY_W_ID     "ERROR : Invalid Item Supply Warehouse. Try
Again."
#define ERR_INVALID_ITEM_NUM        "ERROR : Invalid Item Number. Try Again."
#define ERR_INVALID_ITEM_OTY        "ERROR : Invalid Item Qty. Try Again."
#define ERR_MISSING_C_ID_OR_CLAST"ERROR : Must Enter Customer Id or Customer
Last Name. Try Again."
#define ERR_INVALID_PAYMENT_AMOUNT"ERROR : Invalid Payment Amount. Try
Again."
#define ERR_INVALID_CARRIER         "ERROR : Invalid Carrier Number. Try Again."
#define ERR_INVALID_THRESHOLD       "ERROR : Invalid Threshold. Try Again."
#define ERR_INVALID_C_D_ID        "ERROR : Invalid Customer District Id. Try Again."
#define ERR_INVALID_C_W_ID          "ERROR : Invalid Customer Warehouse Id. Try
Again."
#define ERR_TERMINAL_FULL       "ERROR : Terminal can not support user.  Terminal
full."
#define ERR_C_ID_OR_CLAST_ONLY     "ERROR : Either customer id or customer last
name can be specified."

#define ERR_UNABLE_TO_OPEN_REG          -50
#define ERR_DLVY_THREAD_FAILED          -51
#define ERR_DLVY_SEMAPHORE_INIT_FAILED  -52
#define ERR_DLVY_EVENT_INIT_FAILED      -53
#define ERR_DLVY_QUEUE_EATING_TAIL      -54
#define ERR_DLVY_QUEUE_CALLOC_FAIL      -55

#define ERR_INVALID_USERNAME          -70
#define ERR_INVALID_PASSWORD          -71
#define ERR_INVALID_DB_NAME           -72
#define ERR_INVALID_REGISTERY_KEY     -73
#define ERR_DB2_DLL_NOT_LOADED        -74
#define ERR_ORACLE_DLL_NOT_LOADED     -75
#define ERR_CONNECT_ADDRESS_NOT_FOUND  -76
#define ERR_NORD_ADDRESS_NOT_FOUND     -77
#define ERR_PYMT_ADDRESS_NOT_FOUND     -78
#define ERR_ORDS_ADDRESS_NOT_FOUND     -79
#define ERR_DLVY_ADDRESS_NOT_FOUND     -80
#define ERR_STOK_ADDRESS_NOT_FOUND     -81
#define ERR_NULL_DLL_NOT_LOADED        -82
#define ERR_UNKNOWN_DB                 -83
#define ERR_DISCONNECT_ADDRESS_NOT_FOUND -84
#define ERR_ORA_DLL_NOT_LOADED         -85
#define ERR_SYB_DLL_NOT_LOADED         -86
#define ERR_DBINIT_ADDRESS_NOT_FOUND   -87

#define ERR_SAVING_CONTEXT             -90
#define ERR_DETACHING_CONTEXT          -91
#define ERR_ATTACHING_CONTEXT          -92
#define ERR_HANDLE_IN_USE              -93

#define ERR_CONNECT_TO_TM_FAILED    -99
#define ERR_DLVY_LOG_OPEN_FAILED    -100
#define ERR_DLVY_QUEUE_FULL         -101

////////////////////////////////////////////////////////////////
// Registery Definitions
////////////////////////////////////////////////////////////////
#define     REGISTERY_SUB_KEY   "SOFTWARE\\TPCC"

#define     DELIVERY_THREADS      "dlvyThreads"
#define     DELIVERY_QUEUE_LEN    "dlvyQueueLen"
#define     DELIVERY_LOG_PATH     "dlvyLogPath"
#define     ERROR_LOG_FILE        "errorLogFile"
#define     HTML_TRACE_LOG_FILE     "htmlTraceLogFile"
#define     DB_NAME                 "dbName"
#define     NULL_DB             "nullDB"
#define     COM_NULL_DB             "comnullDB"
#define     CLIENT_NULL_DB          "clientNullDB"

#define     NUM_USERS           "numUsers"
#define     DB_TYPE             "dbType"

#define     TXN_MONITOR             "txn_server"
#define     COMM_POOL           "comm_pool"
#define     HTML_TRACE          "htmlTrace"
#define     ISAPI_TRACE         "isapi_trace"

#define     DEFAULT_DLVY_THREADS    1
#define     DEFAULT_DLVY_QUEUE_LEN    10
#define     DEFAULT_DLVY_LOG_PATH     "c:\\inetpub\\wwwroot\\tpcc\\dlvy"
#define     DEFAULT_ERROR_LOG_FILE    "c:\\inetpub\\wwwroot\\tpcc\\errorLog.txt"
#define     DEFAULT_HTML_TRACE_LOG_FILE
   "c:\\inetpub\\wwwroot\\tpcc\\htmlTrace.txt"
#define     DEFAULT_NUM_USERS         10000

#define     DEFAULT_DB_NAME         "tpcc"

////////////////////////////////////////////////////////////////
// Structure defines
////////////////////////////////////////////////////////////////

struct nord_wrapper {
    in_neword_struct in_nord;
    out_neword_struct   out_nord;
};

struct paym_wrapper {
   struct in_payment_struct in_paym;
   struct out_payment_struct out_paym;
};

struct ords_wrapper {
   struct in_ordstat_struct in_ords;
   struct out_ordstat_struct out_ords;
};

struct dlvy_wrapper {
   struct in_delivery_struct in_dlvy;
   struct out_delivery_struct out_dlvy;
};

struct stok_wrapper {
   struct in_stocklev_struct in_stok;
   struct out_stocklev_struct out_stok;
};

typedef struct
{
   int year;
   int month;
   int day;

   int hour;
   int minute;
   int second;
} datetime;

struct NEWORDERDATA
{
   struct   in_items_struct {
      int  s_OL_I_ID;
      int  s_OL_SUPPLY_W_ID;
      short  s_OL_QUANTITY;
   } in_item[15];

   long long  in_s_O_ENTRY_D_time;   /* init by SUT */
   int    in_s_C_ID;
   int    in_s_W_ID;
   short in_s_D_ID;
   short in_s_O_OL_CNT;      /* init by SUT */
   short in_s_all_local;
   short in_duplicate_items;

   struct   out_items_struct {
      float  s_I_PRICE;
      float  s_OL_AMOUNT;
      short s_S_QUANTITY;
      char    s_I_NAME[25];
      char    s_brand_generic;
   } out_item[15];

   long long  out_s_O_ENTRY_D_time;
   double  out_s_W_TAX;
   double  out_s_D_TAX;
   double  out_s_C_DISCOUNT;
   double  out_s_total_amount;
   int    out_s_O_ID;
   short out_s_O_OL_CNT;
   short out_s_transtatus;
   short out_deadlocks;
   char    out_s_C_LAST[17];
   char    out_s_C_CREDIT[3];
```

```
};

struct PAYMENTDATA
{
    long long   in_s_H_DATE_time;
    double  in_s_H_AMOUNT;
    int     in_s_W_ID;
    int     in_s_C_W_ID;
    int     in_s_C_ID;
    short in_s_C_D_ID;
    short in_s_D_ID;
    char  in_s_C_LAST[17];

    long long   out_s_H_DATE_time;
    long long   out_s_C_SINCE_time;
    double  out_s_C_CREDIT_LIM;
    double  out_s_C_BALANCE;
    double  out_s_C_DISCOUNT;
    int     out_s_C_ID;
    short out_s_transtatus;
    short out_deadlocks;
    char  out_s_W_STREET_1[21];
    char  out_s_W_STREET_2[21];
    char  out_s_W_CITY[21];
    char  out_s_W_STATE[3];
    char  out_s_W_ZIP[10];
    char  out_s_D_STREET_1[21];
    char  out_s_D_STREET_2[21];
    char  out_s_D_CITY[21];
    char  out_s_D_STATE[3];
    char  out_s_D_ZIP[10];
    char  out_s_C_FIRST[17];
    char  out_s_C_MIDDLE[3];
    char  out_s_C_LAST[17];
    char  out_s_C_STREET_1[21];
    char  out_s_C_STREET_2[21];
    char  out_s_C_CITY[21];
    char  out_s_C_STATE[3];
    char  out_s_C_ZIP[10];
    char  out_s_C_PHONE[17];
    char  out_s_C_CREDIT[3];
    char  out_s_C_DATA[201];
};

struct ORDERSTATUSDATA
{
    int   in_s_C_ID;
    int   in_s_W_ID;
    short in_s_D_ID;
    char  in_s_C_LAST[17];

    double out_s_C_BALANCE;
    long long  out_s_O_ENTRY_D_time;
    int    out_s_C_ID;
    int    out_s_O_ID;
    short  out_s_O_CARRIER_ID;
    short  out_s_ol_cnt;
    struct out_oitems_struct {
        long long  s_OL_DELIVERY_D_time;
        float   s_OL_AMOUNT;
        int   s_OL_I_ID;
        int   s_OL_SUPPLY_W_ID;
        short s_OL_QUANTITY;
    } out_item[15];
    short  out_s_transtatus;
    short  out_deadlocks;
    char   out_s_C_FIRST[17];
    char   out_s_C_MIDDLE[3];
    char   out_s_C_LAST[17];
};

struct DELIVERYDATA
{
    long long  in_s_O_DELIVERY_D_time;
    int   in_s_W_ID;
    short in_s_O_CARRIER_ID;
    int   out_s_O_ID[10];
    short out_s_transtatus;
    short outdeadlocks;
};

struct STOCKLEVELDATA
{
    int   in_s_threshold;
    int   in_s_W_ID;
    short in_s_D_ID;

    int   out_s_low_stock;
    short out_s_transtatus;
    short out_deadlocks;
};

struct DLVYQUEUEDATA
{
    int     warehouse;
    short in_s_0_CARRIER_ID;

    struct _timeb enqueueTime;
};


//              MISCELLANEOUS HELPER FUNCTIONS
inline void appendText(char **string,char *text);
inline void appendText(char **string,char *text,int length,int justify);
inline void appendChar(char **string,char byte);
inline void DEBUGMSG(FILE * debugFile, char * message);
inline void appendSpaces(char **string,int spaces);

inline void calcOutDateTime(const INT64b value,datetime *timestamp);
inline int copyOutPhone(char *buffer,char *value,int len);
inline bool copyInMoney64(const char * value,INT64b *number);
inline bool copyInMoney32(const char * value,float *number);
inline int copyInMoney(const char *value);
inline void copyOutMoney64(char *buffer,INT64b value,unsigned int len);
inline int copyOutDateTime(char*buffer,INT64b value);
inline int copyOutDate(char *buffer,INT64b value);
inline int copyOutTime(char *buffer,INT64b value);
inline int copyOutDecimal64(char *buffer,INT64b value,unsigned int len);

inline UINT16b changeOrder16(UINT16b value);
inline UINT32b changeOrder32(UINT32b value);
inline UINT64b changeOrder64(UINT64b value);

inline INT16b changeOrder16(INT16b value);
inline INT32b changeOrder32(INT32b value);
inline INT64b changeOrder64(INT64b value);

//
// Name         : appendText
// Description  :
//              Append text to string
// Parameters   :
//              char ** - string point to append to
//              char * - text to append
// Returns      :
//              None
// Comments     :
//

inline void appendText(char **string,char *text)
{
    while(*text)
    {
        *(*string)++ = *text++;
    }

    **string='\0';
    return;
}

//
// Name         : appendText
// Description  :
//              Append text to string
// Parameters   :
//              char ** - string point to append to
//              char * - text to append
//               int - total field length including blank spaces
//               int - justify flag
// Returns      :
//              None
// Comments     :
//               right justify
//               left justify


inline void appendText(char **string,char *text,int length,int justify)
{
    int byteCount = 0;

    if(justify)
    {
        while(*text)
        {
            *(*string)++ = *text++;
            if (++byteCount > length)
                break;
        }

        //append blank spaces if text is less than length at end
        for(byteCount;byteCount < length;byteCount++)
            *(*string)++ = ' ';
    }
    else
    {
        long long textLen = strlen(text);
        for(textLen;textLen < length;textLen++)
            *(*string)++ = ' ';

        while(*text)
            *(*string)++ = *text++;

    }
    **string='\0';
}


// Name                 : appendChar
// Description  :
//                              Append text to string
// Parameters   :
//                              char ** - string point to append to
//                              char * - text to append
// Returns              :
//                              None
// Comments             :
//

inline void appendChar(char **string,char byte)
```

```
{
    *(*string)++ = byte;
    **string='\0';

    return;
}


//
// Name         : appendSpaces
// Description :
//                appends buffer spaces to result page
// Parameters    :
//                  **htmlPage
//
// Returns      :
//                  amount of characters the function appened
//                  to the html page
// Comments     :
//

inline void appendSpaces(char **string,int spaces)
{
    for(int index=0;index<spaces;index++)
    {
        *(*string)++ = ' ';
    }

    **string='\0';
}

//
// Name         : appendCustData
// Description :
//                appends cust data buffer to result page
// Parameters    :
//                  **htmlPage
//
// Returns      :
//
//                  Adds a newline character every 50 characters displayed.
// Comments     :
//

inline void appendCustData(char **string,char *text)
{
    short byteCount = 0;
    while(*text)
    {
        *(*string)++ = *text++;
        byteCount++;
        if((byteCount % 50) == 0)
        {
            *(*string)++ = '\n';
            *(*string)++ = ' ';*(*string)++ = ' ';*(*string)++ = ' ';*(*string)++ = ' ';
            *(*string)++ = ' ';*(*string)++ = ' ';*(*string)++ = ' ';*(*string)++ = ' ';
            *(*string)++ = ' ';*(*string)++ = ' ';*(*string)++ = ' ';
        }
    }
    **string='\0';
}



//
// calcOutDateTime
//
// Title        : Calculate date & time data out of class array
// Parameters    : INT64b - date & time expressed in seconds
//                datetime * - timestamp
// Return Value  : None
// Comments     :
//

inline void calcOutDateTime(const INT64b value,datetime *timestamp)
{
    // fixed days in each month (FEB 29 is special case)
    static int daysInMonth[12]  = {31,28,31,30,31,30,31,31,30,31,30,31};

    // mask out EPOC seconds
    int dateValue = ((int) (value & 0xffffffff)) +
                    (SECONDS_IN_DAY - (GMT_OFFSET * SECONDS_IN_HOUR));

    int offset    = (int) (value >> 32);

    // break out the seconds
    int hms      = dateValue % SECONDS_IN_DAY;
    int days = dateValue / SECONDS_IN_DAY;

    int years    = (days - 1) / DAYS_IN_YEAR;
    int leaps    = years / YEARS_IN_LEAP;

    int daysUsed = (years * DAYS_IN_YEAR) + leaps;

    // adjust the number of days to account for calculated years
    days = days - daysUsed;

    // set the starting year, month, and day
    timestamp->day     = 1;
    timestamp->month = 1;
    timestamp->year        = START_YEAR + years;

    // is the current year a leap year
    int leap = !(timestamp->year % YEARS_IN_LEAP);

        // apply remaining days based on days in months
        int daysInCurrentMonth;

        while(days)
        {
            // get days in current month
            daysInCurrentMonth = daysInMonth[timestamp->month - 1];
            if(timestamp->month == 2 && leap)
                daysInCurrentMonth = daysInCurrentMonth + 1;

            // days > days in current month
            if(days > daysInCurrentMonth)
            {
                // increment month
                timestamp->month += 1;
                days = days - daysInCurrentMonth;

                // month exceeds months in year
                if(timestamp->month > MONTHS_IN_YEAR)
                {
                    // increment year and reset month
                    timestamp->year += 1; timestamp->month = 1;

                    // are we now on a leap year
                    leap = !(timestamp->year % YEARS_IN_LEAP);
                }
            }
            else
            {
                // set day of month to remaioning days
                timestamp->day = days; days = 0;
            }
        }

        // set time values to remaining seconds
        timestamp->hour = hms / SECONDS_IN_HOUR;
        hms = hms % SECONDS_IN_HOUR;

        timestamp->minute = hms / SECONDS_IN_MINUTE;
        timestamp->second = hms % SECONDS_IN_MINUTE;
        return;
    }

//
// copyOutZip
//
// Title        : Copy zip data out of class array
// Parameters    : char * - buffer to copy zip string into
//
// Return Value  : int - Length of copy
// Comments     :
//

inline int copyOutZip(char *buffer,char *value,int len = DEFAULT_ZIP_LEN)
{
    int index        = 0;
    int bufferPos = 0;

    // add each digit of zip number to buffer inserting delimiter at 5
    while(value[index] && bufferPos < len)
    {
        if(index == 5)
            buffer[bufferPos++] = ZIP_DELIMITER;

        buffer[bufferPos++] = value[index++];
    }

    // space fill to the required length
    while(bufferPos < len)
        buffer[bufferPos++] = CHAR_FILL;

    buffer[bufferPos] = NULL;
    return len;
}

//
// copyOutPhone
//
// Title        : Copy phone data out of class array
// Parameters    : char * - buffer to copy phone string into
//
// Return Value  : int - Length of copy
// Comments     :
//

inline int copyOutPhone(char *buffer,char *value,int len = DEFAULT_PHONE_LEN)
{
    int index        = 0;
    int bufferPos = 0;

    // add each digit of phone number to buffer inserting delimiter before 6, 9, and 12
    while(value[index] && index < len)
    {
        switch(index)
        {
        case  6:
        case  9:
        case 12:
            // insert delimiter
            buffer[bufferPos++] = PHONE_DELIMITER;
        default:
            // add phone digit to buffer
            buffer[bufferPos++] = value[index++];
        }
    }
```

```
    // space fill to the required length
    while(bufferPos < len)
        buffer[bufferPos++] = CHAR_FILL;

    buffer[bufferPos] = '\0';

    return len;
}

//
// copyInMoney64
//
// Title       : Copy money data into class array
// Parameters  : const char * - value string
// Return Value : INT64b integer value
// Comments    :
//

inline bool copyInMoney64(const char * value,INT64b *number)
{
    //INT64b   number        = 0;
    int    index       = 0;
    int    decimal       = 0;
    int    decimals      = 0;
    int    digitsAfterDec   = 0;

    bool  negitiveFlag  = false;

    // convert each digit to a numeric portion
    while(value[index])
    {
        // handle $ . - All the rest assumed numeric
        switch(value[index])
        {
        case MONEY_SYMBOL:
            // ignore $ sign
            break;
        case NEGITIVE_SYMBOL:
            // set negitive flag
            negitiveFlag = true;
            break;

        case DECIMAL_SYMBOL:
            // set decimal
            decimal=1;
            decimals++;
            if(decimals >1)
                //more than 1 decimal point found
                return false;
            break;

        default:
            // adjust decimal places
            decimal = decimal * 10;

            // add digit to running total
            if(value[index] >= '0' && value[index] <= '9')
            {
                if(decimal)
                    if(++digitsAfterDec > 2)
                        return false;

                *number = (*number * 10) + (value[index] - '0');
            }
            else
            {
                //non-numeric field inserted
                return false;
            }
        }
        index++;
    }


    // apply decimal where decimal not found
    if(decimal < 100)
    {
        if(decimal)
        {
            *number *= (100 / decimal);
        }
        else
        {
            *number *= 100;
        }
    }

    // make negitive
    if(negitiveFlag)
        *number = *number * (-1);

    return true;
}

inline bool copyInMoney32(const char * value,double *number)
{
    int    index       = 0;
    int    decimal       = 0;
    int    decimals      = 0;
    int    digitsAfterDec   = 0;

    bool  negitiveFlag  = false;

    // convert each digit to a numeric portion
    while(value[index])
    {
        // handle $ . - All the rest assumed numeric
        switch(value[index])
        {
        case MONEY_SYMBOL:
            // ignore $ sign
            break;
        case NEGITIVE_SYMBOL:
            // set negitive flag
            negitiveFlag = true;
            break;

        case DECIMAL_SYMBOL:
            // set decimal
            decimal=1;
            decimals++;
            if(decimals >1)
                //more than 1 decimal point found
                return false;
            break;

        default:
            // adjust decimal places
            decimal = decimal * 10;

            // add digit to running total
            if(value[index] >= '0' && value[index] <= '9')
            {
                if(decimal)
                    if(++digitsAfterDec > 2)
                        return false;

                *number = (*number * 10) + (value[index] - '0');
            }
            else
            {
                //non-numeric field inserted
                return false;
            }
        }
        index++;
    }


    // apply decimal where decimal not found
    if(decimal < 100)
    {
        if(digitsAfterDec==1)
        {
            *number /= 10;
        }
        else if (digitsAfterDec==2)
        {
            *number /= 100;
        }
    }

    // make negitive
    if(negitiveFlag)
        *number = *number * (-1);

    return true;
}


//
// copyInMoney
//
// Title       : Convert char string money field to double
// Parameters  : const char * - value string
// Return Value : double integer value
// Comments    :
//

inline int copyInMoney(const char *value)
{
    char buf[20];
    int i,j,decimalFound,digitsAfterDecimal=0;

    int decimal=0;

    //walk past $ if present in char string
    if(*value == '$')
        *value++;

    int len=(int)strlen(value);
    for (i=0;i<len;i++)
    {
        if(value[i] == '.')
        {
            decimalFound++;
            if(decimalFound > 1)
                return -1;
        }
        if(value[i] == '-')


        if (value[i] != '.')
        {
            if(decimal)
            {
                if(digitsAfterDecimal<2)
                    digitsAfterDecimal++;
                else
```

```
                return -1;
            }
            buf[j++] = value[i];
        }
    }
    int amount = atoi(buf);

    return amount;
}


//
// copyOutMoney64
//
// Title       : Copy money data out of class array
// Parameters  : char * - buffer to copy string 64 bit money into
//               INT64b - value
//               unsigned len - max number of bytes to copy
// Return Value : int - Length of copy
// Comments    :
//

inline void copyOutMoney64(char *buffer,INT64b value,unsigned int len =
DEFAULT_MONEY64_LEN)
{
    unsigned int  index       = len;

    int           places      = 0;

    bool          negitiveFlag  = false;
    bool          moneyFlag     = true;

    // NULL terminate string
    buffer[index] = NULL;

    // check length > 0
//  if(!index) return len;

    // handle negitive value
    if(value < 0)
    {
        negitiveFlag = true;
        value = value * (-1);
    }

    // break off each digit from value, fill if needed
    do
    {
        if(value)
        {
            // get next digit and add to buffer
            buffer[--index] = (char) (value % 10 + '0');
            value /= 10; places++;

            if(places == 2 && index)
            {
                places++;
                buffer[--index] = DECIMAL_SYMBOL;
            }
        }
        else
        {
            // add zeros to first place before decimal point on (i.e. 0.00)
            if(places < 2 || places == 3)
            {
                buffer[--index] = ZERO_SYMBOL;
            }
            else
            {
                // add the decimal point
                if(places == 2)
                {
                    buffer[--index] = DECIMAL_SYMBOL;
                }
                else
                {
                    // add the negitive indicator
                    if(negitiveFlag)
                    {
                        negitiveFlag = false;
                        buffer[--index] = NEGITIVE_SYMBOL;
                    }
                    else
                    {
                        // add the money indicator
                        if(moneyFlag)
                        {
                            moneyFlag = false;
                            buffer[--index] = MONEY_SYMBOL;
                        }
                        else buffer[--index] = NUMERIC_FILL;
                    }
                }
            }

            // need to trace place for decimal point and zero fill
            places++;
        }
    } while(index);

    //return len;
}

//
// copyOutDateTime
//
// Title       : Copy date & time data out of class array
// Parameters  : char * - buffer to copy date & time string into
//               INT64b - value
// Return Value : int - Length of copy
// Comments    : Fixed length
//

inline int copyOutDateTime(char*buffer,INT64b value)
{
    datetime timestamp;

    // break value into  time/date components
    calcOutDateTime(value,&timestamp);

    // put month into buffer
    *buffer++ =   (char) ((timestamp.month / 10) + '0');
    *buffer++ =   (char) ((timestamp.month % 10) + '0');
    *buffer++ =   DATE_DELIMITER;

    // put day into    buffer
    *buffer++ =   (char) ((timestamp.day / 10) + '0');
    *buffer++ =   (char) ((timestamp.day % 10) + '0');
    *buffer++ =   DATE_DELIMITER;

    // put year into buffer
    int year = timestamp.year;
    *buffer++ =   (char) ((year /   1000) + '0');   year = year   % 1000;
    *buffer++ =   (char) ((year /   100) + '0');year = year   % 100;
    *buffer++ =   (char) ((year /   10)   + '0');
    *buffer++ =   (char) ((year %  10)   + '0');
    *buffer++ =   CHAR_FILL;

    // put hour into buffer
    *buffer++ =   (char) ((timestamp.hour / 10) +   '0');
    *buffer++ =   (char) ((timestamp.hour % 10) + '0');
    *buffer++ =   TIME_DELIMITER;

    // put minute into buffer
    *buffer++ =   (char) ((timestamp.minute /   10)   + '0');
    *buffer++ =   (char) ((timestamp.minute %  10)   + '0');
    *buffer++ =   TIME_DELIMITER;

    // put second into buffer
    *buffer++ =   (char) ((timestamp.second /   10)   + '0');
    *buffer++ =   (char) ((timestamp.second % 10)   + '0');

    *buffer  = NULL;    return DEFAULT_DATETIME_LEN;
}
//
// copyOutTime
//
// Title       : Copy date data out of class array
// Parameters  : char * - buffer to copy date string into
//               INT64b - value
// Return Value : int - Length of copy
// Comments    : Fixed length
//

inline int copyOutDate(char *buffer,INT64b value)
{
    datetime timestamp;

    // break value into time/date components
    calcOutDateTime(value,&timestamp);

    // put month into buffer
    *buffer++ = (char) ((timestamp.month / 10) + '0');
    *buffer++ = (char) ((timestamp.month % 10) + '0');
    *buffer++ = DATE_DELIMITER;

    // put day into buffer
    *buffer++ = (char) ((timestamp.day / 10) + '0');
    *buffer++ = (char) ((timestamp.day % 10) + '0');
    *buffer++ = DATE_DELIMITER;

    // put year into buffer
    int year = timestamp.year;
    *buffer++ = (char) ((year / 1000) + '0');    year = year % 1000;
    *buffer++ = (char) ((year / 100) + '0');  year = year % 100;
    *buffer++ = (char) ((year / 10) + '0');
    *buffer++ = (char) ((year % 10) + '0');
    *buffer++ = CHAR_FILL;

    *buffer = NULL;

    return DEFAULT_DATE_LEN;
}

//
// copyOutTime
//
// Title       : Copy time data out of class array
// Parameters  : char * - buffer to copy time string into
//               INT64b - value
// Return Value : int - Length of copy
// Comments    : Fixed length TBD
//

inline int copyOutTime(char *buffer,INT64b value)
{
    datetime timestamp;

    // break value into time/date components
    calcOutDateTime(value,&timestamp);
```

```
    // put hour into buffer
    *buffer++ = (char) ((timestamp.hour / 10) + '0');
    *buffer++ = (char) ((timestamp.hour % 10) + '0');
    *buffer++ = TIME_DELIMITER;

    // put minute into buffer
    *buffer++ = (char) ((timestamp.minute / 10) + '0');
    *buffer++ = (char) ((timestamp.minute % 10) + '0');
    *buffer++ = TIME_DELIMITER;

    // put second into buffer
    *buffer++ = (char) ((timestamp.second / 10) + '0');
    *buffer++ = (char) ((timestamp.second % 10) + '0');

    *buffer = NULL; return DEFAULT_TIME_LEN;
}

//
// copyOutDecimal64
//
// Title       : Copy decimal data out of class array
// Parameters  : char * - buffer to copy string 64 bit money into
//               INT64b - value
//               unsigned len - max number of bytes to copy
// Return Value : int - Length of copy
// Comments    :
//

inline int copyOutDecimal64(char *buffer,INT64b value,unsigned int len =
DEFAULT_DECIMAL64_LEN)
{
    unsigned int  index        = len;

    int           places       = 0;

    bool          negitiveFlag  = false;

    // NULL terminate string
    buffer[index] = NULL;

    // check length > 0
    if(!index) return len;

    // handle negitive value
    if(value < 0)
    {
        negitiveFlag = true;
        value = value * (-1);
    }

    // break off each digit from value, fill if needed
    do
    {
        if(value)
        {
            // get next digit and add to buffer
            buffer[--index] = (char) (value % 10 + '0');
            value /= 10; places++;

            if(places == 2 && index)
            {
                places++;
                buffer[--index] = DECIMAL_SYMBOL;
            }
        }
        else
        {
            // add zeros to first place before decimal point on (i.e. 0.00)
            if(places < 2 || places == 3)
            {
                buffer[--index] = ZERO_SYMBOL;
            }
            else
            {
                // add the decimal point
                if(places == 2)
                {
                    buffer[--index] = DECIMAL_SYMBOL;
                }
                else
                {
                    // add the negitive indicator
                    if(negitiveFlag)
                    {
                        negitiveFlag = false;
                        buffer[--index] = NEGITIVE_SYMBOL;
                    }
                    else buffer[--index] = NUMERIC_FILL;
                }
            }

            // need to trace place for decimal point and zero fill
            places++;
        }
    } while(index);

    return len;
}

////////////////////////////////////////////////////////////////////////
//  Macros
////////////////////////////////////////////////////////////////////////
using namespace std;

#ifdef DEBUG
    extern int debugFlag;
#else
    extern int debugFlag;
#endif

inline BYTE8b *debugFileName(BYTE8b *filePath)
{
    BYTE8b *fileName = filePath + strlen(filePath);

    while(fileName != filePath)
    {
        if(*fileName == '/' || *fileName == '\\' && *(fileName + 1))
            return (fileName + 1);

        fileName--;
    }

    return filePath;
}

extern char *get_time_prefix(char *buffer);

#define DEBUGADDRESS(POINTER)    hex << (void *) POINTER << dec

#define ERRORMSG(TEXT)  {                                           \
                        EnterCriticalSection(&errorMutex);          \
                        char buf[50];                               \
                        errorStream << debugFileName(__FILE__)      \
                        << "|" << get_time_prefix(buf) << "|" << __LINE__ << "|"  \
                        << _getpid() << "|" << GetCurrentThreadId() << "|"  \
                        << TEXT;                                    \
                        errorStream.flush();                        \
                        LeaveCriticalSection(&errorMutex);          \
                        }

#ifdef DEBUG

#define DEBUGMSG(TEXT)  {                                           \
                        EnterCriticalSection(&debugMutex);          \
                        char buf[50];                               \
                        debugStream << debugFileName(__FILE__)      \
                        << "|" << get_time_prefix(buf) << "|" << __LINE__ << "|"  \
                        << _getpid() << "|" << GetCurrentThreadId() << "|"  \
                        << TEXT ;                                   \
                        debugStream.flush();                        \
                        LeaveCriticalSection(&debugMutex);          \
                        }


   #define DEBUGSTRING(TEXT,LENGTH)                                 \
           debugVarString(TEXT,LENGTH)


#else
       #define DEBUGMSG(TEXT) ;
       #define DEBUGSTRING(TEXT,LENGTH) ;

#endif

inline bool copyInMoney32(const char * value,float *number)
{
    int     index        = 0;
    int     decimal      = 0;
    int     decimals     = 0;
    int     digitsAfterDec   = 0;
    bool  negitiveFlag  = false;

    // convert each digit to a numeric portion
    while(value[index])
    {
        // handle $ . - All the rest assumed numeric
        switch(value[index])
        {
        case MONEY_SYMBOL:
            // ignore $ sign
            break;
        case NEGITIVE_SYMBOL:
            // set negitive flag
            negitiveFlag = true;
            break;

        case DECIMAL_SYMBOL:
            // set decimal
            decimal=1;
            decimals++;
            if(decimals >1)
                //more than 1 decimal point found
                return false;
            break;

        default:
            // adjust decimal places
            decimal = decimal * 10;

            // add digit to running total
            if(value[index] >= '0' && value[index] <= '9')
            {
                if(decimal)
                    if(++digitsAfterDec > 2)
                        return false;

                *number = (*number * 10) + (value[index] - '0');
            }
            else
```

```
        {
            //non-numeric field inserted
            return false;
        }
    }
    index++;
}


    // apply decimal where decimal not found

#ifndef DB2

    if(decimal)
    {
        *number /= decimal;
    }
#endif

    // make negitive
    if(negitiveFlag)
        *number = *number * (-1);

    return true;
}
#endif /* _COMMON_TPCC */
```

htmlPhraser.h

```
////////////////////////////////////////////////////////////////
// htmlPharaser.h
////////////////////////////////////////////////////////////////
// Class to decode a html query string
////////////////////////////////////////////////////////////////

#pragma once

#include <memory.h>

////////////////////////////////////////////////////////////////
// Definitions
////////////////////////////////////////////////////////////////

#define NULL                    0

#define COMMAND_ID              0
#define TERM_ID                 1
#define W_ID                    2
#define D_ID                    3
#define C_ID                    4
#define C_NAME                  5

#define C_W_ID                  6
#define C_D_ID                  7
#define AMT_PAID                8

#define STK_THRESHOLD           9
#define CARRIER_NUM             10

#define ITEM_LIST_START         11
#define ITEM_LIST_FINISH        55

#define MAX_QUERY_ID            55
#define MAX_FIELD_LEN           256
#define MAX_FIELD_NUM           56

////////////////////////////////////////////////////////////////
// Command Codes
////////////////////////////////////////////////////////////////

#define NEW_ORDER_CODE              'n'
#define PAYMENT_CODE                'p'
#define ORDER_STATUS_CODE           'o'
#define DELIVERY_CODE               'd'
#define STOCK_CODE                  's'
#define EXIT_CODE                   'e'
#define MENU_CODE                   'm'

#define COMMAND_LOGIN               0
#define COMMAND_NEW_ORDER           1
#define COMMAND_PAYMENT             2
#define COMMAND_ORDER_STATUS        3
#define COMMAND_DELIVERY            4
#define COMMAND_STOCK               5
#define COMMAND_EXIT                6

#define COMMAND_LOGIN_RESULTS           7
#define COMMAND_NEW_ORDER_RESULTS       8
#define COMMAND_PAYMENT_RESULTS         9
#define COMMAND_ORDER_STATUS_RESULTS    10
#define COMMAND_DELIVERY_RESULTS        11
#define COMMAND_STOCK_RESULTS           12

////////////////////////////////////////////////////////////////
// Class htmlPhraser
////////////////////////////////////////////////////////////////

class htmlPhraser
{
    // Constructors / Destructor
    public:
        htmlPhraser(char *queryString);
        ~htmlPhraser()          {return;}

    // geters
    public:
        int getCommandId();
        int validate(int txnType);

        char *  get_TERM_ID()       {return iQueryValues[TERM_ID];}
        char *  get_W_ID()          {return iQueryValues[W_ID];}
        char *  get_D_ID()          {return iQueryValues[D_ID];}
        char *  get_C_ID()          {return iQueryValues[C_ID];}
        char *  get_C_NAME()        {return iQueryValues[C_NAME];}
        char *  get_C_W_ID()        {return iQueryValues[C_W_ID];}
        char *  get_C_D_ID()        {return iQueryValues[C_D_ID];}
        char *  get_AMT_PAID()      {return iQueryValues[AMT_PAID];}
        char *  get_STK_THRESHOLD()     {return iQueryValues[STK_THRESHOLD];}
        char *  get_CARRIER_NUM()   {return iQueryValues[CARRIER_NUM];}

        char *  get_ITEM_SUPP_W(int item) {return iQueryValues[(ITEM_LIST_START + 0)
+ (item * 3)];}
        char *  get_ITEM_ITEM_NUM(int item) {return iQueryValues[(ITEM_LIST_START +
1) + (item * 3)];}
        char *  get_ITEM_QTY(int item)      {return iQueryValues[(ITEM_LIST_START + 2)
+ (item * 3)];}

    // Class Functions
    private:
        char convertQueryToken(char **queryString);

    // Class Attributes
    private:
        int iCustomerIdFlag;
        int iCarrierNumFlag;
        int iStockThresholdFlag;

        char  iQueryValues[MAX_FIELD_NUM][MAX_FIELD_LEN];
};
```

```
////////////////////////////////////////////////////////////////
```

resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by tpccIsapi.rc
//
#define IDS_PROJNAME 100



// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        201
#define _APS_NEXT_COMMAND_VALUE         32768
#define _APS_NEXT_CONTROL_VALUE         201
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

StdAfx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once

#define WIN32_LEAN_AND_MEAN         // Exclude rarely-used stuff from Windows headers

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS // some CString constructors will
be explicit

// turns off ATL's hiding of some common and often safely ignored warning messages
#define _ATL_ALL_WARNINGS

// critical error descriptions will only be shown to the user
// in debug builds. they will always be logged to the event log
#ifndef _DEBUG
#define ATL_CRITICAL_ISAPI_ERROR_LOGONLY
#endif


#ifndef _WIN32_WINNT
#define _WIN32_WINNT 0x0403
#endif

// TODO: this disables support for registering COM objects
// exported by this project since the project contains no
// COM objects or typelib. If you wish to export COM objects
// from this project, add a typelib and remove this line
#define _ATL_NO_COM_SUPPORT

#include "resource.h"
#include <atlsrvres.h>
#include <atlisapi.h>
#include <atlstencil.h>

// TODO: reference additional headers your program requires here
```

htmlPhraser.cpp

```
//////////////////////////////////////////////////////////////////
// htmlPhraser.cpp
//////////////////////////////////////////////////////////////////
// Class implmentation of htmlPhraser.
// This class will take a query string and break it into a series
// of consituant parts
//////////////////////////////////////////////////////////////////

#include "htmlPhraser.h"

//////////////////////////////////////////////////////////////////
// htmlPhraser::htmlPhraser
//////////////////////////////////////////////////////////////////
// Title   : Constructor
// Parameters    : char * query string
// Return Value  : None
// Comments :
//////////////////////////////////////////////////////////////////

htmlPhraser::htmlPhraser(char *queryString)
{
    // initilize query values
    iCustomerIdFlag = iCarrierNumFlag = iStockThresholdFlag = false;

    // this initilizes the query list to NULL's. This means that
    // characters being added are overwriting null characters and
    // therfore the string will be null terminated implicitly.

    memset(iQueryValues,NULL,(MAX_FIELD_NUM * MAX_FIELD_LEN));

    // controls
    char      queryChar     = NULL;

    int    queryIndex       = -1;
    int    valueIndex       = -1;

    // process each characeter of query string
    while(*queryString)
    {
        // check for special case characters
        if(queryChar)
        {
            // a percentage sign would indicate a token
            if(*queryString != '%')
            {
                // a plus sign repersents a space
                if(*queryString == '+')
                {
                    queryChar = ' ';
                    *queryString++;
                }
                else queryChar = *queryString++;
            }
            else queryChar = convertQueryToken(&queryString);
        }
        else queryChar = '&';

        // handle query reference (&)
        if(queryChar == '&')
        {
            // reset value index
            valueIndex = -1;

            // do we have a numeric query reference
            if(*queryString >= '0' && *queryString <= '9')
            {
                // numeric query id
                queryIndex =
                    ((*queryString - '0') * 10) + (*(queryString + 1) - '0');

                // walk past the two command characters
                queryString += 2;

                // validate query value
                if(queryIndex > MAX_QUERY_ID)
                  queryIndex = -1;
            }
            else queryIndex = -1;

            // finished processing for query reference
            continue;
        }

        // we have a query reference but need to wait until we see '='
        // before accepting value

        if(valueIndex == -1)
        {
            // we are waiting for '='
            if(queryChar == '=')
            {
                valueIndex = 0;

                // set query string flags
                switch(queryIndex)
                {
                case C_ID:
                    iCustomerIdFlag = true; break;
                case CARRIER_NUM:
                    iCarrierNumFlag = true; break;
                case STK_THRESHOLD:
                    iStockThresholdFlag = true; break;
                default: break;
                }
            }

            // finishes looging for '='
            continue;
        }

        // add each character to the query value
        if(queryIndex > -1 && valueIndex > -1)
        {
            // we are processing a query value
            if(valueIndex < MAX_FIELD_LEN)
            {
                // we have not exceeded max line len
                iQueryValues[queryIndex][valueIndex++] = queryChar;
            }
            continue;
        }
    }

    return;
}

//////////////////////////////////////////////////////////////////
// htmlPhraser::getCommandId
//////////////////////////////////////////////////////////////////
// Title   : Returns the page command
// Parameters    : None
// Return Value  : int - page command
// Comments :
//////////////////////////////////////////////////////////////////

int htmlPhraser::getCommandId()
{
    // return command numeric code
    switch(*iQueryValues[COMMAND_ID])
    {
    case NEW_ORDER_CODE:
        if(iCustomerIdFlag)
            return COMMAND_NEW_ORDER_RESULTS;
        else return COMMAND_NEW_ORDER;
    case PAYMENT_CODE:
        if(iCustomerIdFlag)
            return COMMAND_PAYMENT_RESULTS;
        else return COMMAND_PAYMENT;
    case ORDER_STATUS_CODE:
        if(iCustomerIdFlag)
            return COMMAND_ORDER_STATUS_RESULTS;
        else return COMMAND_ORDER_STATUS;
    case DELIVERY_CODE:
        if(iCarrierNumFlag)
            return COMMAND_DELIVERY_RESULTS;
        else return COMMAND_DELIVERY;
    case STOCK_CODE:
        if(iStockThresholdFlag)
            return COMMAND_STOCK_RESULTS;
        else return COMMAND_STOCK;
    case MENU_CODE:
        return COMMAND_LOGIN_RESULTS;
    case EXIT_CODE:
        return COMMAND_EXIT;
    default:
        return COMMAND_LOGIN;
    };


    // should not get here
    return COMMAND_LOGIN;
}

//////////////////////////////////////////////////////////////////
// htmlPhraser::validate
//////////////////////////////////////////////////////////////////
// Title   : validate url parameter list for all txn types
// Parameters    : int - txn type
// Return Value  : int - error code
// Comments :
//////////////////////////////////////////////////////////////////


int validate(int txnType)
{
    return 0;
}

//////////////////////////////////////////////////////////////////
// htmlPhraser::convertQueryToken
//////////////////////////////////////////////////////////////////
// Title   : Returns the page command
// Parameters    : None
// Return Value  : int - page command
// Comments :
//////////////////////////////////////////////////////////////////

char htmlPhraser::convertQueryToken(char **queryString)
{
    char queryChar = NULL;

    // skip over %
    (*queryString)++;

    // look at first character
    switch(**queryString)
    {
    case '2':
        {
            // what follows?
```

```
        (*queryString)++;

        switch(**queryString)
        {
        case '1':
            queryChar = '!';
            break;
        case '3':
            queryChar = '#';
            break;
        case '4':
            queryChar = '$';
            break;
        case '5':
            queryChar = '%';
            break;
        case '6':
            queryChar = '&';
            break;
        case '8':
            queryChar = '(';
            break;
        case '9':
            queryChar = ')';
            break;
        case 'B':
            queryChar = '+';
            break;
        case 'C':
            queryChar = ',';
            break;
        case 'F':
            queryChar = '/';
            break;
        case ' ':
            queryChar = ' ';
            break;
        }
    }

    break;
case '3':
    {
        // what follows?
        (*queryString)++;

        switch(**queryString)
        {
        case 'A':
            queryChar = ':';
            break;
        case 'B':
            queryChar = ';';
            break;
        case 'D':
            queryChar = '=';
            break;
        case 'F':
            queryChar = '?';
            break;
        case ' ':
            queryChar = ' ';
            break;
        }
    }

    break;
case '4':
    {
        // what follows?
        (*queryString)++;

        switch(**queryString)
        {
        case '0':
            queryChar = '@';
            break;
        case ' ':
            queryChar = ' ';
            break;
        }
    }

    break;
case '5':
    {
        // what follows?
        (*queryString)++;

        switch(**queryString)
        {
        case 'B':
            queryChar = '[';
            break;
        case 'D':
            queryChar = ']';
            break;
        case 'E':
            queryChar = '^';
            break;
        case ' ':
            queryChar = ' ';
            break;
        }
    }

    break;
case '7':
    {
        // what follows?
        (*queryString)++;

        switch(**queryString)
        {
        case 'B':
            queryChar = '{';
            break;
        case 'C':
            queryChar = '|';
            break;
        case 'D':
            queryChar = '}';
            break;
        case 'E':
            queryChar = '~';
            break;
        case ' ':
            queryChar = ' ';
            break;
        }
    }

    break;
case '+':
    queryChar = '+';
    break;
}

// advance pointer and return
(*queryString)++; return queryChar;
}

//////////////////////////////////////////////////////////////////

**StdAfx.cpp**

// stdafx.cpp : source file that includes just the standard includes
//  tpccIsapi.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STDAFX.H
// and not in this file

**tpccIsapi.cpp**

/*
********************************************************************
** Project      : AIX
** Component    : Performance/TPC-C Benchmark
** Name         : tpccIsapi.cpp
** Title        : TPCC html processing
********************************************************************
** Copyright (c) 2003 IBM Corporation
** All rights reserved
********************************************************************
** History      :
**              Developed at IBM Austin by the AIX RS/6000
**              performance group.
**
** Comments     :
**
********************************************************************
*/

#include "stdafx.h"

#include "..\tpccCom\tpccCom.h"
#include "..\tpccCom\tpccCom_i.c"
#include "tpccIsapi.hpp"

// For custom assert and trace handling with WebDbg.exe
[ module(name="tpccIsapi", type="dll") ];
[ emitidl(restricted) ];

#define _WIN32_DCOM

#ifdef _DEBUG
    int debugFlag = 1;
#else
    int debugFlag = 0;
#endif

//////////////////////////////////////////////////////////////
// Globals
//////////////////////////////////////////////////////////////

int     maxDataSize;            //max struct size of all txn(s)
int     numUsers;               //number of users that client will service.
int     FirstClient;
int     dlvyQueueLen;           //static length of dlvy queue
int     dlvyThreads;            //number of dlvy threads to create
int     dlvyBufferFreeSlots;    //length of dlvy txn queue
int     dlvyBufferSlotIndex;    //index into next avaliable slot in dlvy txn queue
int     dlvyBufferThreadIndex;  //thread index into dlvy txn queue
int     nullDB;                 //null db on client(bypass com call).
int     trace;
```

```
static DWORD          threadLSIndex;        //isapi thread local storage index
CRITICAL_SECTION    isapiLock;            //isapi lock
CRITICAL_SECTION    errorLock;            //error log file lock.
CRITICAL_SECTION    termLock;             //terminal array lock.
CRITICAL_SECTION    dlvyQueueLock;        //dlvy queue critical section lock
HANDLE                dlvyThreadDone = INVALID_HANDLE_VALUE;     //dlvy thread exit
event
HANDLE                dlvyThreadSemaphore = INVALID_HANDLE_VALUE; //dlvy thread
wrk to do semaphore
int                   dlvyThreadID = 0;

struct DLVYQUEUEDATA  *dlvyQueue;             //dlvy queue
HANDLE                *dlvyThreadHandles;     //ptr to array of thread handles

TERM_ENTRY  *termArray;                       //array of terminal entries to store each users
info.
int         termNextFree;                     //next available slot in terminal array

FILE *htmlDebug      =  NULL;             //html debug file
FILE *errorLog      =  NULL;              //error file
FILE *htmlTrace      =  NULL;

ofstream debugStream;
ofstream errorStream;
CRITICAL_SECTION debugMutex;
CRITICAL_SECTION errorMutex;

char  dlvyLogPath[128]      = {NULL};
char  errorLogFile[128]     = {NULL};
char  htmlTraceLogFile[128]   = {NULL};
char  dbName[64]            = {NULL};
char  dbType[16]            = {NULL};

typedef INT (*CONNECT_PTR)(char *dbName,void **connectHandle);
typedef INT   (*DISCONNECT_PTR)(void *connectHandle);
typedef INT (*DLVY_FUNC_PTR)(dlvy_wrapper *dlvy,void *connectHandle);
typedef INT (*NORD_FUNC_PTR)(nord_wrapper *nord,void *connectHandle);
typedef INT (*PYMT_FUNC_PTR)(paym_wrapper *pymt,void *connnectHandle);
typedef INT (*ORDS_FUNC_PTR)(ords_wrapper *ords,void *connectHandle);
typedef INT (*STOK_FUNC_PTR)(stok_wrapper *stok,void *connectHandle);

HINSTANCE     dbInstance;
CONNECT_PTR       db_connect;
DISCONNECT_PTR  db_disconnect;
DLVY_FUNC_PTR    dlvyCall;


////////////////////////////////////////////////////////
// Page functions arrays
////////////////////////////////////////////////////////

typedef int (*pageFuncPtr) (htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);

pageFuncPtr htmlPageFunctions[MAX_TRANSACTIONS] =
        {
            {doLoginForm},
            {doNewOrderForm},
            {doPaymentForm},
            {doOrderStatusForm},
            {doDeliveryForm},
            {doStockForm},
            {doExit},
            {doLoginResults},
            {doNewOrderResults},
            {doPaymentResults},
            {doOrderStatusResults},
            {doDeliveryResults},
            {doStockResults}
        };
```

```
extern "C" DWORD WINAPI HttpExtensionProc(LPEXTENSION_CONTROL_BLOCK
lpECB)
{
    struct TXN_HANDLE *txnHandle = NULL;

    txnHandle = (TXN_HANDLE *) TlsGetValue(threadLSIndex);

    if(txnHandle == NULL)
    {
        int rc = initTxnHandle(&txnHandle);
        if (rc != OK)
        {
            char response[256];   char htmlHeader[256];
            sprintf(response,"ERROR : Init txnHandle function failed.\n");

            size_t htmlPageLen = strlen(response);

            //add content length and keep alive header
            sprintf(htmlHeader,HEADER,htmlPageLen);
            lpECB->ServerSupportFunction(lpECB-
>ConnID,HSE_REQ_SEND_RESPONSE_HEADER,"200
OK",NULL,(DWORD*)htmlHeader);
            lpECB->WriteClient(lpECB->ConnID,response,(LPDWORD)&htmlPageLen,0);

            return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
        }

        txnHandle = (TXN_HANDLE *) TlsGetValue(threadLSIndex);
        if (txnHandle == NULL)
        {
            char response[256];   char htmlHeader[256];
            sprintf(response,"ERROR : Unable to retrieve txnHandle from TLS.\n");

            size_t htmlPageLen = strlen(response);

            //add content length and keep alive header
            sprintf(htmlHeader,HEADER,htmlPageLen);
            lpECB->ServerSupportFunction(lpECB-
>ConnID,HSE_REQ_SEND_RESPONSE_HEADER,"200
OK",NULL,(DWORD*)htmlHeader);
            lpECB->WriteClient(lpECB->ConnID,response,(LPDWORD)&htmlPageLen,0);

            return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
        }
    }

    try
    {
        txnHandle->urlString  = (char*)lpECB->lpszQueryString;

        DEBUGMSG("calling doHtml() w/ query string:" << txnHandle->urlString << endl);
        doHtml(txnHandle);

        size_t htmlPageLen;
        htmlPageLen = strlen(txnHandle->htmlPage);
        if(htmlPageLen >= 4096)
        {
            ERRORMSG("WARNING:  HTML PAGE IS >= 4096!, page
size:"<<htmlPageLen<<endl);
        }
        //add content length and keep alive header
        sprintf(txnHandle->htmlHeader,HEADER,htmlPageLen);
        size_t headerLen = strlen(txnHandle->htmlHeader);
        if(headerLen >= 256)
        {
            ERRORMSG("WARNING:  HTML HEADER IS >= 256!, header
size:"<<headerLen<<endl);
        }

        //write response to user
```

```
        lpECB->ServerSupportFunction(lpECB-
>ConnID,HSE_REQ_SEND_RESPONSE_HEADER,"200 OK",NULL,(DWORD*)txnHandle-
>htmlHeader);
        lpECB->WriteClient(lpECB->ConnID,txnHandle-
>htmlPage,(LPDWORD)&htmlPageLen,0);

        DEBUGMSG("HTML PAGE-->"<<endl<<txnHandle->htmlHeader<<txnHandle-
>htmlPage<<endl);

    }
    catch (...)
    {
        char response[256];
        ZeroMemory(response,256);
        char *ptr = response;

        appendText(&ptr,"<HTML><BODY> Error : Unhandled Exception
</BODY></HTML>");
        DWORD   cbResponse = sizeof(response)-1 ;

        //write response to user
        lpECB->ServerSupportFunction(lpECB-
>ConnID,HSE_REQ_SEND_RESPONSE_HEADER,"200 OK",NULL,(DWORD*)response);
        lpECB->WriteClient(lpECB->ConnID,response,&cbResponse,0);
    }

    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}

extern "C" BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO* pVer)
{
  // Create the extension version string, and copy string to HSE_VERSION_INFO structure.
    pVer->dwExtensionVersion =  MAKELONG(HSE_VERSION_MINOR,
HSE_VERSION_MAJOR);

    // Copy description string into HSE_VERSION_INFO structure.
    strcpy(pVer->lpszExtensionDesc, "TPCC ISAPI Extension");

    // Initialze isapi critcal section
    InitializeCriticalSection(&isapiLock);

    // Initailize error log critcal section
    InitializeCriticalSection(&errorLock);

    // Initialze terminal critcal section
    InitializeCriticalSection(&termLock);

    // Initialize debug/error critcal sections
    if(debugFlag)
        InitializeCriticalSection(&debugMutex);
    InitializeCriticalSection(&errorMutex);

    // Read registery values
    if(readRegisteryValues() != OK)
        return(FALSE);

    // Initialize terminal array
    termArray = (TERM_ENTRY*) calloc(numUsers,sizeof(TERM_ENTRY));
    termNextFree = 1;

    //open up error/debug streams
    errorStream.rdbuf( )->open(errorLogFile,ios::out);
    if(debugFlag)
        debugStream.rdbuf( )->open(htmlTraceLogFile,ios::out);

    ERRORMSG("Error log file open."<<endl);

    DEBUGMSG("Loading library for dlvy txn."<<endl);
    int rc = getDBInstance();
    if (rc != OK)
```

```
    {
        ERRORMSG("Error, unable to load database dll, rc:"<<rc);
        DEBUGMSG("Error, unable to load database dll, rc:"<<rc);

        return FALSE;
    }
    DEBUGMSG("Library loaded for dlvy txn."<<endl);

    DEBUGMSG("Calling initDlvy." <<endl);
    ERRORMSG("Calling initDlvy." <<endl);

    if(initDlvy() != OK)
        return (FALSE);

    DEBUGMSG("Initializing TLS." << endl);

    // Initialize thread local storage index
    threadLSIndex = TlsAlloc();
    if (threadLSIndex == TLS_NULL)
    {
        ERRORMSG("Isapi error: unable to initialize thread local storage(TLS), rc:" <<
GetLastError()<<endl);
        return(FALSE);
    }
    ERRORMSG("Initialized TLS." << endl);

    DEBUGMSG("sizeof out_neword_struct:   "<<sizeof(struct out_neword_struct)<<endl);
    DEBUGMSG("sizeof  in_neword_struct:   "<<sizeof(struct in_neword_struct)<<endl);
    DEBUGMSG("sizeof out_payment_struct:  "<<sizeof(struct out_payment_struct)<<endl);
    DEBUGMSG("sizeof  in_payment_struct:  "<<sizeof(struct in_payment_struct)<<endl);
    DEBUGMSG("sizeof out_ordstat_struct:  "<<sizeof(struct out_ordstat_struct)<<endl);
    DEBUGMSG("sizeof  in_ordstat_struct:  "<<sizeof(struct in_ordstat_struct)<<endl);
    DEBUGMSG("sizeof out_delivery_struct: "<<sizeof(struct out_delivery_struct)<<endl);
    DEBUGMSG("sizeof  in_delivery_struct: "<<sizeof(struct in_delivery_struct)<<endl);
    DEBUGMSG("sizeof out_stocklev_struct: "<<sizeof(struct out_stocklev_struct)<<endl);
    DEBUGMSG("sizeof  in_stocklev_struct: "<<sizeof(struct in_stocklev_struct)<<endl);

    //compute the max struct size for com data construct
    maxDataSize = max(maxDataSize,sizeof(nord_wrapper));
    maxDataSize = max(maxDataSize,sizeof(paym_wrapper));
    maxDataSize = max(maxDataSize,sizeof(ords_wrapper));
    maxDataSize = max(maxDataSize,sizeof(dlvy_wrapper));
    maxDataSize = max(maxDataSize,sizeof(stok_wrapper));
    maxDataSize += 10;

    DEBUGMSG("max data struct size:"<<maxDataSize <<endl);
    ERRORMSG("max data struct size:"<<maxDataSize <<endl);

    return true;
}

extern "C" BOOL WINAPI TerminateExtension(DWORD dwFlags)
{
//  ERRORMSG("TerminateExtension"<<endl);
    return true;
}


/*
**********************************************************************
** Name            :   initTxnHandle
** Description     :
**                 Isapi thread initializes its own com interface
**                 structure.
** Parameters      :
**                 TXN_HANDLE*     isapi txn handle
** Returns         :
**                 int - return code
** Comments        :
**
**********************************************************************
*/
int initTxnHandle(TXN_HANDLE **txnHandle)
{
    DEBUGMSG("Inside init txn handle, getting isapiLock." << endl);
//  ERRORMSG("Inside init txn handle, getting isapiLock." << endl);

    EnterCriticalSection(&isapiLock);

    HRESULT hres = NULL;

    try
    {
        DEBUGMSG("Got ispaiLock, initializing txnHandle:
"<<DEBUGADDRESS(*txnHandle)<< endl);
//      ERRORMSG("Got ispaiLock, initializing txnHandle:
"<<DEBUGADDRESS(*txnHandle)<< endl);
        *txnHandle = (TXN_HANDLE *) calloc(1,sizeof(TXN_HANDLE));
        if (*txnHandle == NULL)
        {
            ERRORMSG("Unable to allocated TXN_HANDLE, rc:"<<GetLastError()<<endl);
            return ERR;
        };

        (*txnHandle)->comInterface.comHandle = NULL;
        DEBUGMSG("Initializing txnHandle com data buffer to
"<<maxDataSize<<"bytes"<<endl);
//      ERRORMSG("Initializing txnHandle com data buffer to
"<<maxDataSize<<"bytes"<<endl);
        (*txnHandle)->comInterface.txnBuffer = (char *) CoTaskMemAlloc(maxDataSize);
        if (!((*txnHandle)->comInterface.txnBuffer))
        {
            ERRORMSG("CoTaskMemAlloc() failed of size "<<maxDataSize<<", rc:
"<<hres<<endl);
            return(ERR);
        };
        DEBUGMSG("txnHandle com data buffer initialized to " << maxDataSize << "bytes"
<<endl);
//      ERRORMSG("txnHandle com data buffer initialized to " << maxDataSize << "bytes"
<<endl);

        //(*txnHandle)->comInterface.comHandle = NULL;
        DEBUGMSG("Calling CoInitialize with txnHandle:
"<<DEBUGADDRESS(*txnHandle)<<endl);
//      ERRORMSG("Calling CoInitialize with txnHandle:
"<<DEBUGADDRESS(*txnHandle)<<endl);
        hres = CoInitializeEx(NULL,COINIT_MULTITHREADED);
        if (FAILED(hres))
        {
            ERRORMSG("CoInitializeEx() failed, rc : "<<hres<<endl);
            return(ERR);
        };

        struct  _timeb          startTime;
        struct  _timeb          endTime;

        DEBUGMSG("Calling CoCreateInstance with
txnHandle:"<<DEBUGADDRESS(*txnHandle)<< endl);
//      ERRORMSG("Calling CoCreateInstance with
txnHandle:"<<DEBUGADDRESS(*txnHandle)<< endl);
        _ftime(&startTime);
        hres =
CoCreateInstance(CLSID_tpcc_com,NULL,CLSCTX_SERVER,IID_ltpcc_com,(void
**)&(*txnHandle)->comInterface.comHandle);
        if (FAILED(hres))
        {
            _ftime(&endTime);
            //store error code in txnHandle
            ERRORMSG("CoCreateInstance() failed, code:"<<HRESULT_CODE(hres)<<"
facility:"<<HRESULT_FACILITY(hres)<<
                " hres:"<<hres<< " time waiting:"<<
                ((((endTime.time - startTime.time)*1000)+
                  (endTime.millitm - startTime.millitm))/1000.0)<<endl);

            DEBUGMSG("CoCreateInstance() failed, code:"<<HRESULT_CODE(hres)<<"
facility:"<<HRESULT_FACILITY(hres)<<
                " hres:"<<hres<< " time waiting:"<<
                ((((endTime.time - startTime.time)*1000)+
                  (endTime.millitm - startTime.millitm))/1000.0)<<endl);

//          return(ERR);
        };

        _ftime(&endTime);
        DEBUGMSG("CoCreateInstance successful.txnHande com initialized, time waiting for
object to be activated:" <<
            ((((endTime.time - startTime.time)*1000)+
              (endTime.millitm - startTime.millitm))/1000.0)<<endl);
//      ERRORMSG("CoCreateInstance successful.txnHande com initialized, time waiting for
object to be activated:" <<
//          ((((endTime.time - startTime.time)*1000)+
//            (endTime.millitm - startTime.millitm))/1000.0)<<endl);

        //call set complete to return object to pool.
        (*txnHandle)->comInterface.comHandle->doSetComplete();

        hres = (*txnHandle)->comInterface.comHandle->doDBInfo();
        (*txnHandle)->comInterface.comHandle->doSetComplete();

        //set the com buffers size
        DEBUGMSG("Setting txnHandle: " << DEBUGADDRESS(*txnHandle) << "com buffer
size to " << maxDataSize<< endl)
//      ERRORMSG("Setting txnHandle: " << DEBUGADDRESS(*txnHandle) << "com buffer
size to " << maxDataSize<< endl)
        (*txnHandle)->comInterface.size = maxDataSize;

        DEBUGMSG("txnHandle: "<<DEBUGADDRESS(*txnHandle) <<"set to " <<
maxDataSize << endl);
//      ERRORMSG("txnHandle: "<<DEBUGADDRESS(*txnHandle) <<"set to " <<
maxDataSize << endl);

        TlsSetValue(threadLSIndex,*txnHandle);

        DEBUGMSG("txnHandle: "<<DEBUGADDRESS(*txnHandle) << "stored in TLS" <<
endl);
//      ERRORMSG("txnHandle: "<<DEBUGADDRESS(*txnHandle) << "stored in TLS" <<
endl);

        ZeroMemory((*txnHandle)->htmlPage,MAX_HTML_PAGE_LEN);
        ZeroMemory((*txnHandle)->htmlHeader,MAX_HTML_HEADER_LEN);

        LeaveCriticalSection(&isapiLock);
        return(OK);
    }
    catch(...)
    {
        DEBUGMSG("Unhandled exeception in initTxnHandle, unlocking isapi lock" <<endl);
//      ERRORMSG("Unhandled exeception in initTxnHandle, unlocking isapi lock" <<endl);
        LeaveCriticalSection(&isapiLock);
    };

    return ERR;
}

/*
**********************************************************************
** Name            :   getDBInstance
** Description     :
**                 load db specific lib based on dbType registery
**                 value.
```

```
** Parameters   :
**
** Returns     :
**                int - return code
** Comments    :
**                This function only exists for the dlvy threads
**                Dlvy threads hold direct connections to the database
**                and therefore need to know what db interface to talk to.
*****************************************************************
*/
int getDBInstance()
{
    if(nullDB)
    {
        dbInstance = LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\nullDB.dll");
        if(dbInstance == NULL)
        {
            return ERR_NULL_DLL_NOT_LOADED;
        }
    }
    else if( (strcmp(dbType,"DB2") == 0) )
    {
        dbInstance = LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\tpccDB2glue.dll");
        if(dbInstance == NULL)
        {
            return ERR_DB2_DLL_NOT_LOADED;
        }
    }
    else if( (strcmp(dbType,"ORACLE") == 0) )
    {
        ERRORMSG("Loading Oracle dll"<<endl);
        dbInstance = LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\tpccOracleGlue.dll");
        if(dbInstance == NULL)
        {
            ERRORMSG("Could not Load Oracle dll"<<endl);
            return ERR_ORA_DLL_NOT_LOADED;
        }
        ERRORMSG("Loaded Oracle dll"<<endl);
    }
    else if( (strcmp(dbType,"SYBASE") == 0) )
    {
        ERRORMSG("Loading Sybase dll"<<endl);
        dbInstance = LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\tpccSybaseGlue.dll");
        if(dbInstance == NULL)
        {
            ERRORMSG("Could not Load Sybase dll"<<endl);
            return ERR_SYB_DLL_NOT_LOADED;
        }
        ERRORMSG("Loaded Sybase dll"<<endl);
    }
    else
    {
        return ERR_UNKNOWN_DB;
    }

    ERRORMSG("Get address"<<endl);
    db_connect = (CONNECT_PTR)GetProcAddress(dbInstance,"connect_db");
    if(db_connect == NULL)
    {
        ERRORMSG("Could not find connect_db function in dll"<<endl);
        return ERR_CONNECT_ADDRESS_NOT_FOUND;
    }
    ERRORMSG("Get address"<<endl);
    dlvyCall = (DLVY_FUNC_PTR)GetProcAddress(dbInstance,"do_dlvy");
    if(dlvyCall == NULL)
    {
        ERRORMSG("Could not find do_dlvy in dll"<<endl);
        return ERR_DLVY_ADDRESS_NOT_FOUND;
    }
    return OK;
}

/*
*****************************************************************
** Name          :   initDlvy
** Description   :
**                initialize dlvy threads/dlvy queueu
** Parameters    :
**
** Returns     :
**                int - return code
** Comments    :
**
*****************************************************************
*/

int initDlvy()
{
    ERRORMSG(">>initDlvy"<<endl);
    // Initialze critcal section
    InitializeCriticalSection(&dlvyQueueLock);

    //create dlvy queue
    dlvyQueue = (DLVYQUEUEDATA *) calloc(dlvyQueueLen,sizeof(DLVYQUEUEDATA));

    if (dlvyQueue == NULL)
    {
        ERRORMSG("calloc failed to allocate dlvyQueue"<<endl);
        return  ERR_DLVY_QUEUE_CALLOC_FAIL;
    }
    ERRORMSG(">>calloc"<<endl);
    //init dlvy buffer critcal section
    //InitializeCriticalSection(&dlvyQueueLock);

    dlvyThreadDone = CreateEvent(NULL,
                                 TRUE,    //manual reset
                                 FALSE,   //initially not signalled.
                                 NULL);
    if(dlvyThreadDone == NULL)
    {
        DEBUGMSG("Error: dlvyThreadDone handled init failed,
GetLastError:"<<GetLastError()<<endl);

        ERRORMSG("Error : dlvyThreadDone handled init failed,
GetLastError:"<<GetLastError()<<endl);

        return  ERR_DLVY_EVENT_INIT_FAILED;
    }
    //create dlvy semaphore
    dlvyThreadSemaphore = CreateSemaphore(NULL,0,dlvyQueueLen,NULL);
    if(dlvyThreadSemaphore == NULL)
    {
        DEBUGMSG("Error: dlvyThreadSemaphore semaphore init failed,
GetLastError:"<<GetLastError()<<endl);
        ERRORMSG("Error: dlvyThreadSemaphore semaphore init failed,
GetLastError:"<<GetLastError()<<endl);
        return  ERR_DLVY_SEMAPHORE_INIT_FAILED;

    }

    //set number of free slots available in queue
    dlvyBufferFreeSlots = dlvyQueueLen;

    //index into next avaliable slot in dlvy txn queue
    dlvyBufferSlotIndex = 0;

    //thread index into dlvy txn queue
    dlvyBufferThreadIndex = 0;

    dlvyThreadHandles = new HANDLE[dlvyThreads];

    //create threads
    for(int threadCount = 0;threadCount < dlvyThreads;threadCount++)
    {
        ERRORMSG(">>Calling dlvyThreadEntry"<<endl);
        dlvyThreadHandles[threadCount] = (HANDLE)_beginthread(dlvyThreadEntry,0,NULL);
        if(dlvyThreadHandles[threadCount] == INVALID_HANDLE_VALUE) {
            ERRORMSG(">>Calling dlvyThreadEntry failed"<<endl);
            return ERR_DLVY_THREAD_FAILED;
        }
    }


    return OK;
}


/*
*****************************************************************
** Name          :   readRegisteryValues
** Description   :
**                initialize isapi global variables from registery
** Parameters    :
**
** Returns     :
**                int - return code
** Comments    :
**
*****************************************************************
*/
int readRegisteryValues()
{
    HKEY   registeryKey;
    char  value[MAX_STRING_LEN];
    DWORD   regType;
    DWORD   regValue;
    DWORD   regValueSize = MAX_STRING_LEN;

//  ERRORMSG(">>readRegisteryValues"<<endl);
    //open up registery key
    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTERY_SUB_KEY,0,KEY_READ,&r
egisteryKey) != ERROR_SUCCESS)
        return ERR_UNABLE_TO_OPEN_REG;

    //get null db flag
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registeryKey,NULL_DB,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
            nullDB = regValue;
    else
            nullDB = 0;

    //get num dlvy threads
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registeryKey,DELIVERY_THREADS,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
            dlvyThreads = regValue;
    else
            dlvyThreads = DEFAULT_DLVY_THREADS;

    //get dlvy queue len
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registeryKey,DELIVERY_QUEUE_LEN,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
            dlvyQueueLen = regValue;
    else
            dlvyQueueLen = DEFAULT_DLVY_QUEUE_LEN;

    //get the htmlTrace flag
    regValueSize = sizeof(regValue);
```

```
    if(RegQueryValueEx(registeryKey,HTML_TRACE,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
        trace = regValue;
    else
        trace = 0;

    //get the client null db flag
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registeryKey,NULL_DB,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
        nullDB = regValue;
    else
        nullDB = 0;

    //get the num of users
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registeryKey,NUM_USERS,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
        numUsers = regValue;
    else
        numUsers = DEFAULT_NUM_USERS;

    //get dlvy log file path
    regValueSize = sizeof(value);
    if (RegQueryValueEx(registeryKey,DELIVERY_LOG_PATH,0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
        strcpy(dlvyLogPath,value);
    else
        strcpy(dlvyLogPath,DEFAULT_DLVY_LOG_PATH);

    //get global error log file path/name
    regValueSize = sizeof(value);
    if (RegQueryValueEx(registeryKey,ERROR_LOG_FILE,0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
        strcpy(errorLogFile,value);
    else
        strcpy(errorLogFile,DEFAULT_ERROR_LOG_FILE);

    //get global error log file path/name
    regValueSize = sizeof(value);
    if (RegQueryValueEx(registeryKey,HTML_TRACE_LOG_FILE,0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
        strcpy(htmlTraceLogFile,value);
    else
        strcpy(htmlTraceLogFile,DEFAULT_HTML_TRACE_LOG_FILE);

    //get db name
    regValueSize = sizeof(value);
    if (RegQueryValueEx(registeryKey,DB_NAME,0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
        strcpy(dbName,value);
    else
        strcpy(dbName,DEFAULT_DB_NAME);

    //get db type
    regValueSize = sizeof(value);
    if (RegQueryValueEx(registeryKey,DB_TYPE,0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
        strcpy(dbType,value);

    //get First Client
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registeryKey,"FirstClient",0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
        FirstClient = regValue;
    else
        FirstClient = 1;

    RegCloseKey(registeryKey);

    return OK;
}

/*
**********************************************************************
** Name            : doLoginForm
** Description     :
**                 HTML Login page entry point
** Parameters      :
**                 htmlPhrasercommand block
**                 char *      html result page
** Returns         :
**                 int - return code
** Comments        :
**
**********************************************************************
*/

int doLoginForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char buffer[20];
    DEBUGMSG("Entering doLoginForm()."<<endl);
//  ERRORMSG("Entering doLoginForm()."<<endl);
    char *html=txnHandle->htmlPage;

    DEBUGMSG("Creating html login page"<<endl);
    //begin html page
    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Client Home
Page</TITLE></HEAD>"
                    "<FORM ACTION=\""
                    APP_NAME
                    "\" METHOD=\"GET\">"
                    "<H2>Please Login.</H2>"
                    "<INPUT TYPE=\"hidden\" NAME=\""
                    CMD_TXN_ID
                    "\" VALUE=\""
                    CMD_MENU
                    "\">"
                    "<H3>Warehouse <INPUT NAME=\""
                    CMD_W_ID
                    "\" SIZE=6>"
                    " District <INPUT NAME=\""
                    CMD_D_ID
                    "\" SIZE=2></H3>"
                    "<INPUT TYPE=\"submit\" VALUE=\"Submit\">"
                    "</FORM></BODY></HTML>");

    appendText(&html," dlvyBufferFreeSlots ");
    appendText(&html,itoa(dlvyBufferFreeSlots,buffer,10));
    appendText(&html," dlvyBufferSlotIndex ");
    appendText(&html,itoa(dlvyBufferSlotIndex,buffer,10));
    appendText(&html," dlvyBufferThreadIndex ");
    appendText(&html,itoa(dlvyBufferThreadIndex,buffer,10));

    DEBUGMSG("Html login page done"<<endl);
    return OK;
}

/*
**********************************************************************
** Name            : doLoginResults
** Description     :
**                 HTML Login results page entry point
** Parameters      :
**                 htmlPhrasercommand block
**                 char *      html result page
** Returns         :
**                 int - return code
** Comments        :
**
**********************************************************************
*/

int doLoginResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=txnHandle->htmlPage;

    //validate parameters
    if( (txnHandle->w_id = atoi(commandBlock->get_W_ID())) == 0 )
    {
        doLoginErrorPage(html,ERR_INVALID_W_ID);
        return OK;
    }
    if( (txnHandle->d_id = atoi(commandBlock->get_D_ID())) == 0 )
    {
        doLoginErrorPage(html,ERR_INVALID_D_ID);
        return OK;
    }

    //store user into terminal array,
    //function will ERR if the terminal array is full
    if( assignTerminal(txnHandle) != OK)
    {
        doLoginErrorPage(html,ERR_TERMINAL_FULL);
        return OK;
    };

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Main Menu</TITLE></HEAD>\r\n"
                    "<BODY><FORM ACTION=\""
                    APP_NAME
                    "\" METHOD=\"GET\">\r\n"
                    "<H3>Please Select Transaction.</H3>\r\n");
    html+=appendButtons(html);

    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"</FORM></BODY></HTML>");

    return OK;
}

/*
**********************************************************************
** Name            : doLoginErrorPage
** Description     :
**                 HTML Login page entry point
** Parameters      :
**                 char *      html page buffer
**                 char *      error message
** Returns         :
**                 int - return code
** Comments        :
**
**********************************************************************
*/

int doLoginErrorPage(char *htmlPage,char *errorMessage)
{
    char *html=htmlPage;

    //begin html page
    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Client Home
Page</TITLE></HEAD>"
                    "<FORM ACTION=\""
                    APP_NAME
                    "\" METHOD=\"GET\">");

    appendText(&html,"<H2>Please Login.</H2>"
                    "<INPUT TYPE=\"hidden\" NAME=\""
                    CMD_TXN_ID
```

```
                "\" VALUE=\""
                CMD_MENU
                "\">"
                "<H3>Warehouse <INPUT NAME=\""
                CMD_W_ID
                "\" SIZE=6>"
                " District <INPUT NAME=\""
                CMD_D_ID
                "\" SIZE=2></H3>"
                "<INPUT TYPE=\"submit\" VALUE=\"Submit\">"
                "</FORM>");
    appendText(&html,errorMessage);
    appendText(&html,"<BODY></HTML>");

    return OK;
}


/*
**********************************************************************
** Name            : doNewOrderForm
** Description    :
**              HTML neworder page entry point
** Parameters    :
**              htmlPhrasercommand block
**              char *      html result page
** Returns     :
**              int - return code
** Comments     :
**
**********************************************************************
*/


int doNewOrderForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=txnHandle->htmlPage;

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C New Order</TITLE></HEAD>\r\n"
                "<BODY><FORM ACTION=\""
                APP_NAME
                "\" METHOD=\"GET\">\r\n"
                "<CENTER><H3>Please Fill In New Order Form.</H3></CENTER>\r\n"
//check if not needed
                "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                CMD_TXN_ID
                "\" VALUE=\""
                CMD_NORD
                "\">");

    //append the hidden
    html+=appendHiddenFields(html,txnHandle);

    //int buffer for warehouse
    char buffer[15];
    appendText(&html," <PRE>"
//                  "       1       2       3       4       5       6       7       8       9\r\n"
//
"12345678901234567890123456789012345678901234567890123456789012345678901234567890\r\n"
                "Warehouse: ");
    appendText(&html,itoa(txnHandle->w_id,buffer,10),7,1);
    appendText(&html,"District: <INPUT NAME=\""
                CMD_D_ID
                "\" SIZE=1>                 Date:<BR>"
               "Customer <INPUT NAME=\""
                CMD_C_ID
                "\" SIZE=6>  Name:                Credit:     %Disc.:<BR>"
                "Order Number:          Number of Lines:          W_tax:        D_tax:<BR>
<BR>"
//                  "       1       2       3       4       5       6       7       8       9\r\n"
//
"12345678901234567890123456789012345678901234567890123456789012345678901234567890\r\n"
                " Supp_W  Item_Num  Item_Name              Qty Stock  B/G  Price    Amount
<BR> ");

    //append the 15 items commands
    html+=appendItems(html,NORD_ITEMS,ITEM_START);

    //seal up html page
    appendText(&html,"</PRE></BODY></HTML>");

    return OK;
}

/*
**********************************************************************
** Name            : doNewOrderResults
** Description    :
**              HTML neworder page entry point
** Parameters    :
**              htmlPhrasercommand block
**              char *      html result page
** Returns     :
**              int - return code
** Comments     :
**
**********************************************************************
*/


int doNewOrderResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    DEBUGMSG("Entered doNewOrderResults" << endl);

    char *html=txnHandle->htmlPage;
    struct nord_wrapper *nord = NULL;

    DEBUGMSG("Casting COM txnBuffer to nord struct" <<endl);
    nord = (nord_wrapper*)txnHandle->comInterface.txnBuffer;
    ZeroMemory(nord,sizeof(nord_wrapper));
    DEBUGMSG("COM txnBuffer initialized, validating input parameters" << endl);

    //set warehouse,district and customer id from command block
    nord->in_nord.s_W_ID = txnHandle->w_id;
    DEBUGMSG("nord w_id:" <<  nord->in_nord.s_W_ID << endl);

    if( (nord->in_nord.s_D_ID = atoi(commandBlock->get_D_ID()) ) == 0)
    {
        doNewOrderErrorPage(html,ERR_INVALID_D_ID,commandBlock,txnHandle);
        return OK;
    }
    DEBUGMSG("nord d_id:" <<  nord->in_nord.s_D_ID << endl);

    if((nord->in_nord.s_C_ID = atoi(commandBlock->get_C_ID()) ) == 0)
    {
        doNewOrderErrorPage(html,ERR_INVALID_C_ID,commandBlock,txnHandle);
        return OK;
    }
    DEBUGMSG("nord c_id:" <<  nord->in_nord.s_C_ID << endl);

    int itemCmd          = ITEM_START;
    short itemComplete = 0;
    char field[256] ={NULL};

    for (int itemIndex=0;itemIndex<NORD_ITEMS;itemIndex++)
    {
        //supply warehouse
        if( *(commandBlock->get_ITEM_SUPP_W(itemIndex)) )
#ifdef DB2
            if ( (nord->in_nord.in_item[nord->in_nord.s_O_OL_CNT].s_OL_SUPPLY_W_ID =
atoi(commandBlock->get_ITEM_SUPP_W(itemIndex))) == 0)
#elif ORACLE
            if ( (nord->in_nord.s_OL_SUPPLY_W_ID[nord->in_nord.s_O_OL_CNT] =
atoi(commandBlock->get_ITEM_SUPP_W(itemIndex))) == 0)
#elif SYBASE
            if ( (nord->in_nord.s_OL_SUPPLY_W_ID[nord->in_nord.s_O_OL_CNT] =
atoi(commandBlock->get_ITEM_SUPP_W(itemIndex))) == 0)
#endif
            {

       doNewOrderErrorPage(html,ERR_INVALID_SUPPLY_W_ID,commandBlock,txnHandle);
                return OK;
            }
            else
                itemComplete++;

        //item number
        if( *(commandBlock->get_ITEM_ITEM_NUM(itemIndex)) )
        {
            if(itemComplete==1)
            {
#ifdef DB2
                if ( (nord->in_nord.in_item[nord->in_nord.s_O_OL_CNT].s_OL_I_ID =
atoi(commandBlock->get_ITEM_ITEM_NUM(itemIndex))) == 0)
#elif ORACLE
                if ( (nord->in_nord.s_OL_I_ID[nord->in_nord.s_O_OL_CNT] =
atoi(commandBlock->get_ITEM_ITEM_NUM(itemIndex))) == 0)
#elif SYBASE
                if ( (nord->in_nord.s_OL_I_ID[nord->in_nord.s_O_OL_CNT] =
atoi(commandBlock->get_ITEM_ITEM_NUM(itemIndex))) == 0)
#endif
                {

       doNewOrderErrorPage(html,ERR_INVALID_ITEM_NUM,commandBlock,txnHandle);
                    return OK;
                }
                else
                    itemComplete++;
            }
            //missing previous value of item supp warehouse, flag error
            else
            {

       doNewOrderErrorPage(html,ERR_INVALID_SUPPLY_W_ID,commandBlock,txnHandle);
                return OK;
            }
        }
        else if( (itemComplete==1) ) //nothing in the command block, check to see if the
previous item value is present
        {

       doNewOrderErrorPage(html,ERR_INVALID_ITEM_NUM,commandBlock,txnHandle);
            return OK;
        }

        //item qty
        if(*(commandBlock->get_ITEM_QTY(itemIndex)))
        {
            if(itemComplete==2)
            {
#ifdef DB2
                if( (nord->in_nord.in_item[nord->in_nord.s_O_OL_CNT].s_OL_QUANTITY =
atoi(commandBlock->get_ITEM_QTY(itemIndex))) == 0)
#elif ORACLE
                if( (nord->in_nord.s_OL_QUANTITY[nord->in_nord.s_O_OL_CNT] =
atoi(commandBlock->get_ITEM_QTY(itemIndex))) == 0)
#elif SYBASE
```

```
            if( (nord->in_nord.s_OL_QUANTITY[nord->in_nord.s_O_OL_CNT] =
atoi(commandBlock->get_ITEM_QTY(itemIndex))) == 0)
#endif
            {

    doNewOrderErrorPage(html,ERR_INVALID_ITEM_OTY,commandBlock,txnHandle);
                return OK;
            }
            else
                itemComplete++;
        }
        //missing previous value of item number
        else if (itemComplete ==1)
        {

    doNewOrderErrorPage(html,ERR_INVALID_ITEM_NUM,commandBlock,txnHandle);
            return OK;
        }
        //missing 1st value of supp warehouse
        else
        {

    doNewOrderErrorPage(html,ERR_INVALID_SUPPLY_W_ID,commandBlock,txnHandle);
            return OK;
        }
    }
    else if(itemComplete==2)       //nothing in the command block, check to see if the
previous item values are present
    {

    doNewOrderErrorPage(html,ERR_INVALID_ITEM_NUM,commandBlock,txnHandle);
        return OK;
    }

#ifndef DB2
    DEBUGMSG("nord item:" << nord->in_nord.s_O_OL_CNT << "SUPPLY_W_ID:" <<
nord->in_nord.s_OL_SUPPLY_W_ID[nord->in_nord.s_O_OL_CNT] <<
        " OL_I_ID:" << nord->in_nord.s_OL_I_ID[nord->in_nord.s_O_OL_CNT] << "
OL_QUANTITY:" << nord->in_nord.s_OL_QUANTITY[nord->in_nord.s_O_OL_CNT]
<<endl);
#endif
    if(itemComplete == 3)
        nord->in_nord.s_O_OL_CNT++;

    itemComplete=0;
  }

  DEBUGMSG("complete nord items:"<<nord->in_nord.s_O_OL_CNT<<" initializing
remaing unused items " << NORD_ITEMS - nord->in_nord.s_O_OL_CNT  << " to 0"
<<endl);
  for(int itemIndex=nord->in_nord.s_O_OL_CNT;itemIndex<NORD_ITEMS;itemIndex++)
  {
#ifdef DB2
    nord->in_nord.in_item[itemIndex].s_OL_SUPPLY_W_ID=0;
    nord->in_nord.in_item[itemIndex].s_OL_I_ID = 0;
    nord->in_nord.in_item[itemIndex].s_OL_QUANTITY =0;
#elif ORACLE
    nord->in_nord.s_OL_SUPPLY_W_ID[itemIndex]=0;
    nord->in_nord.s_OL_I_ID[itemIndex] = 0;
    nord->in_nord.s_OL_QUANTITY[itemIndex] =0;
#elif SYBASE
    nord->in_nord.s_OL_SUPPLY_W_ID[itemIndex]=0;
    nord->in_nord.s_OL_I_ID[itemIndex] = 0;
    nord->in_nord.s_OL_QUANTITY[itemIndex] =0;
#endif
  }

  DEBUGMSG("nord creating new order results html title page" <<endl);

  appendText(&html,"<HTML><HEAD><TITLE>TPC-C New Order
Results</TITLE></HEAD>\r\n"
              "<BODY><FORM ACTION=\""
              APP_NAME
              "\" METHOD=\"GET\">\r\n");
  //append menu buttons
  html+=appendButtons(html);
  html+=appendHiddenFields(html,txnHandle);

  appendText(&html,"</FORM><CENTER><H3>New Order</H3> <BR></CENTER>"
              "<PRE>"
//              "      1      2      3      4      5      6      7      8      9\r\n"
//
"12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890\r\n
              "");
  //assume failure
  nord->out_nord.s_transtatus = -1;
//  nord->in_nord.len = sizeof(in_neword_struct);
//  nord->out_nord.len = sizeof(out_neword_struct);

  DEBUGMSG("nord executing COM interface function,"<<endl<<
    "nord c_id: " <<  nord->in_nord.s_C_ID << endl <<
    "nord w_id: " <<  nord->in_nord.s_W_ID << endl <<
    "nord d_id: " <<  nord->in_nord.s_D_ID << endl);
  HRESULT hres=0;
  if (txnHandle->comInterface.size > maxDataSize)
  {
    ERRORMSG("[NO]txnHandle->comInterface.size "<<txnHandle->comInterface.size);
  }
  try
  {
    hres = txnHandle->comInterface.comHandle->doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);
  }
  catch(...)
  {
    html+=sprintf(html,"ERROR : nord com call caused exeception to
occur.</PRE></BODY></HTML>");
    ERRORMSG("COM+ exception [NO]txnHandle->comInterface.size "<<txnHandle-
>comInterface.size);
    return OK;
  }

  if(FAILED(hres))
  {
    ERRORMSG("ERROR : nord com call failed, rc:" << hex  << hres);
    DEBUGMSG("ERROR : nord com call failed, rc:" << hex  << hres);
  }
  else
  {
  DEBUGMSG("nord executed OK,"<<endl<<
    "nord c_id: " <<  nord->in_nord.s_C_ID << endl <<
    "nord w_id: " <<  nord->in_nord.s_W_ID << endl <<
    "nord d_id: " <<  nord->in_nord.s_D_ID << endl);
    //com call successful, return object back to pool.
    hres = txnHandle->comInterface.comHandle->doSetComplete();
    if(FAILED(hres))
    {
      ERRORMSG("ERROR : nord setcomplete call failed, rc:" << hex  << hres);
      DEBUGMSG("ERROR : nord setcomplete call failed, rc:" << hex  << hres);
      return OK;
    }
  }

  nord = (nord_wrapper *)txnHandle->comInterface.txnBuffer;
  DEBUGMSG("nord COM interface function successful, s_transtatus:" << nord-
>out_nord.s_transtatus << endl);

  int rc = nord->out_nord.s_transtatus;

  char buffer[10];
  appendText(&html,"Warehouse:  ");
  appendText(&html,itoa(nord->in_nord.s_W_ID,buffer,10),6,1);

  appendText(&html,"District: ");
  appendText(&html,itoa(nord->in_nord.s_D_ID,buffer,10),26,1);

  appendText(&html,"Date: ");
  appendText(&html,nord->out_nord.s_O_ENTRY_D_time,18,1);
  appendText(&html," <BR>"
              "Customer: ");
  appendText(&html,itoa(nord->in_nord.s_C_ID,buffer,10),8,1);

  appendText(&html,"Name: ");
  appendText(&html,nord->out_nord.s_C_LAST,LAST_NAME_LEN+3,1);

  appendText(&html,"Credit: ");
  appendText(&html,nord->out_nord.s_C_CREDIT,5,1);

  appendText(&html,"%Disc.: ");
  if(rc == OK)
  {
    html+=sprintf(html,"%2.2lf",nord->out_nord.s_C_DISCOUNT);
  }
  appendText(&html," <BR>"
            "Order Number: ");
  if(rc != INVALID_STATUS)
    appendText(&html,itoa(nord->out_nord.s_O_ID,buffer,10),10,1);

  appendText(&html,"Number of Lines: ");

  if(rc != INVALID_STATUS)
    appendText(&html,itoa(nord->out_nord.s_O_OL_CNT,buffer,10),10,1);

  appendText(&html,"W_Tax: ");
  if(rc == OK)
  {
    html+=sprintf(html,"%5.2lf",nord->out_nord.s_W_TAX);
  }

  appendText(&html,"   D_Tax: ");
  if(rc == OK)
  {
    html+=sprintf(html,"%5.2lf",nord->out_nord.s_D_TAX);
  }
  appendText(&html," <BR> <BR>"
//              "      1      2      3      4      5      6      7      8      9\r\n"
//
"12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890\r\n"
            " Supp_W  Item_Id   Item_Name               Qty Stock  B/G  Price    Amount <BR>
");

  //display items
  if (rc == OK)
  {
    //display valid items
    for(int itemCount=0;itemCount < nord->out_nord.s_O_OL_CNT;itemCount++)
    {
#ifdef DB2
      appendText(&html,itoa(nord-
>in_nord.in_item[itemCount].s_OL_SUPPLY_W_ID,buffer,10),8,1);
      appendText(&html,itoa(nord-
>in_nord.in_item[itemCount].s_OL_I_ID,buffer,10),10,1);
      appendText(&html,nord-
>out_nord.item[itemCount].s_I_NAME,DEFAULT_STRING_LEN+1,1);
      appendText(&html,itoa(nord-
>in_nord.in_item[itemCount].s_OL_QUANTITY,buffer,10),5,1);
```

```
        appendText(&html,itoa(nord-
>out_nord.item[itemCount].s_S_QUANTITY,buffer,10),7,1);
        html+=sprintf(html,"%c   $%-7.2lf $%-7.2lf <BR> ",nord-
>out_nord.item[itemCount].s_brand_generic,
                                        nord->out_nord.item[itemCount].s_I_PRICE,
                                        nord->out_nord.item[itemCount].s_OL_AMOUNT);
#elif ORACLE
        appendText(&html,itoa(nord-
>in_nord.s_OL_SUPPLY_W_ID[itemCount],buffer,10),8,1);
        appendText(&html,itoa(nord->in_nord.s_OL_I_ID[itemCount],buffer,10),10,1);
        appendText(&html,nord-
>out_nord.s_I_NAME[itemCount],DEFAULT_STRING_LEN+1,1);
        appendText(&html,itoa(nord->in_nord.s_OL_QUANTITY[itemCount],buffer,10),5,1);
        appendText(&html,itoa(nord->out_nord.s_S_QUANTITY[itemCount],buffer,10),7,1);
        html+=sprintf(html,"%c   $%-7.2lf $%-7.2lf <BR> ",nord-
>out_nord.s_brand_generic[itemCount],
                                        nord->out_nord.s_I_PRICE[itemCount]/100.0,
                                        nord->out_nord.s_OL_AMOUNT[itemCount]/100.0);
#elif SYBASE
        appendText(&html,itoa(nord-
>in_nord.s_OL_SUPPLY_W_ID[itemCount],buffer,10),8,1);
        appendText(&html,itoa(nord->in_nord.s_OL_I_ID[itemCount],buffer,10),10,1);
        appendText(&html,nord-
>out_nord.s_I_NAME[itemCount],DEFAULT_STRING_LEN+1,1);
        appendText(&html,itoa(nord->in_nord.s_OL_QUANTITY[itemCount],buffer,10),5,1);
        appendText(&html,itoa(nord->out_nord.s_S_QUANTITY[itemCount],buffer,10),7,1);
        html+=sprintf(html,"%c   $%-7.2lf $%-7.2lf <BR> ", nord-
>out_nord.s_brand_generic[itemCount][0],
                                        nord->out_nord.s_I_PRICE[itemCount],
                                        nord->out_nord.s_OL_AMOUNT[itemCount]);
#endif
      }
      //display blank line for remaining empty items in the order
      for(int lineBreaks=0;lineBreaks < (NORD_ITEMS-nord-
>out_nord.s_O_OL_CNT);lineBreaks++)
        appendText(&html," <BR>");
    }
    else
      appendText(&html," <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR>
<BR> <BR> <BR> <BR>");

    appendText(&html,"\r\n <BR> ");

    html+=displayStatus(html,rc);
    if(rc == OK)
      html+=sprintf(html,"      Total: $%.2lf",nord->out_nord.s_total_amount);
    else
      appendText(&html,"      Total: <BR>");

    appendText(&html,"</PRE></BODY> </HTML>");

    DEBUGMSG("nord html page complete.  returning to calling function" << endl);

    return OK;
}

/*
**********************************************************************
** Name            : doNewOrderErrorPage
** Description    :
**              HTML neworder page entry point
** Parameters   :
**              char *      html result page
**              char *      error message
** Returns     :
**              int - return code
** Comments    :
**
**********************************************************************
*/


int doNewOrderErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=htmlPage;

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C New Order</TITLE></HEAD>\r\n"
                "<BODY><FORM ACTION=\""
                APP_NAME
                "\" METHOD=\"GET\">\r\n"
                "<CENTER><H3>Please Fill In New Order Form.</H3></CENTER>\r\n"
                "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                CMD_TXN_ID
                "\" VALUE=\""
                CMD_NORD
                "\">");

    //append the hidden warehouse and district fields
    html+=appendHiddenFields(html,txnHandle);

    //int buffer for warehouse
    char buffer[15];
    /*appendText(&html,"<PRE>        1       2       3       4       5       6       7       8
9\r\n"

"12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
34567890123456789\r\n"
                "Warehouse: ");*/
    appendText(&html,"<PRE>Warehouse: ");
    appendText(&html,itoa(txnHandle->w_id,buffer,10),7,1);
    appendText(&html,"District: <INPUT NAME=\""
                CMD_D_ID
                "\" SIZE=1>                    Date:<BR>"
               "Customer <INPUT NAME=\""
                CMD_C_ID
                "\" SIZE=6>  Name:                  Credit:     %Disc.:<BR>"
                "Order Number:          Number of Lines:          W_tax:        D_tax:<BR>
<BR>"
                //"       1       2       3       4       5       6       7       8       9\r\n"

//"12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
345678901234567890\r\n"
                " Supp_W  Item_Num  Item_Name                Qty Stock  B/G  Price    Amount
<BR> ");

    //append the 15 items commands
    html+=appendItems(html,NORD_ITEMS,ITEM_START);
    appendText(&html,message);

    //seal up html page
    appendText(&html,"</PRE></BODY></HTML>");

    return OK;
}


/*
**********************************************************************
** Name            : doPaymentForm
** Description    :
**              HTML payment page entry point
** Parameters   :
**              htmlPhrasercommand block
**              char *      html result page
** Returns     :
**              int - return code
** Comments    :
**
**********************************************************************
*/

int doPaymentForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=txnHandle->htmlPage;
    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Payment</TITLE></HEAD>\r\n"
                "<BODY><FORM ACTION=\""
                APP_NAME
                "\" METHOD=\"GET\">\r\n"
                "<CENTER><H3>Please Fill In Payment Form.</H3></CENTER> <BR>\r\n"
                "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                CMD_TXN_ID
                "\" VALUE=\""
                CMD_PYMT
                "\">");
    html+=appendHiddenFields(html,txnHandle);
    appendText(&html,"<BR><PRE>\r\n"
                "Date:<BR>"
                "Warehouse: ");
    char buffer[15];
    appendText(&html,itoa(txnHandle->w_id,buffer,10));

    appendSpaces(&html,10);
    appendText(&html,"District: <INPUT NAME=\""
                CMD_D_ID
                "\" SIZE=1>\r\n<BR>"
                "<BR> <BR> <BR>"
                "Customer: "
                "<INPUT NAME=\""
                CMD_C_ID
                "\" SIZE=5>"
                "   "
                "Cust-Warehouse: "
                "<INPUT NAME=\""
                CMD_C_W_ID
                "\" SIZE=5>"
                "   "
                "Cust-District: "
                "<INPUT NAME=\""
                CMD_C_D_ID
                "\" SIZE=1><BR>"
                "Name:  <INPUT NAME=\""
                CMD_C_NAME
                "\" SIZE=20>");
    appendText(&html,"                              Since: <BR>"
                "                                 "
                "                           Credit: <BR>"
                "                                 "
                "                           %Disc: <BR>"
                "Amount Paid:             "
                "<INPUT NAME=\""
                CMD_AMT_PAID
                "\" SIZE=10>"
                "                                 "
                "New Cust-Balance:<BR>"
                "Credit Limit:<BR> <BR>Cust-Data:<BR> <BR> <BR> <BR> </PRE>");

    return OK;
}

/*
**********************************************************************
** Name            : doPaymentResults
** Description    :
**              HTML neworder page entry point
** Parameters   :
**              htmlPhrasercommand block
**              char *      html result page
** Returns     :
**              int - return code
```

```
** Comments    :
**
*********************************************************************
*/

int doPaymentResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=txnHandle->htmlPage;
    char buffer[50];

    struct paym_wrapper *pymt = NULL;
    pymt = (paym_wrapper*)txnHandle->comInterface.txnBuffer;
    ZeroMemory(pymt,sizeof(paym_wrapper));

    //set login warehouse id from command block
    pymt->in_paym.s_W_ID = txnHandle->w_id;

    //set district from command block
#ifdef SYBASE
    if( (pymt->in_paym.s_D_ID = (CS_TINYINT)atoi(commandBlock->get_D_ID())) == 0)
    {
        doPaymentErrorPage(html,ERR_INVALID_D_ID,commandBlock,txnHandle);
        return OK;
    }
#else
    if( (pymt->in_paym.s_D_ID = atoi(commandBlock->get_D_ID())) == 0)
    {
        doPaymentErrorPage(html,ERR_INVALID_D_ID,commandBlock,txnHandle);
        return OK;
    }
#endif
    //set customer id from command block
    if( (pymt->in_paym.s_C_ID = atoi(commandBlock->get_C_ID())) == 0)
    {
        if(*(commandBlock->get_C_NAME()) == NULL)
        {
            //no customer id nor customer last name specified.

    doPaymentErrorPage(html,ERR_MISSING_C_ID_OR_CLAST,commandBlock,txnHandl
e);
            return OK;
        }
        else
            strcpy(pymt->in_paym.s_C_LAST,commandBlock->get_C_NAME());
    }
    else
    {
        //make sure that the user only inserted just c_id
        if(*(commandBlock->get_C_NAME()) != NULL)
        {

    doPaymentErrorPage(html,ERR_C_ID_OR_CLAST_ONLY,commandBlock,txnHandle);
            return OK;
        }
    }

    //get customer warehose id field
    if( (pymt->in_paym.s_C_W_ID = atoi(commandBlock->get_C_W_ID())) == 0)
    {
        doPaymentErrorPage(html,ERR_INVALID_C_W_ID,commandBlock,txnHandle);
        return OK;
    }

    //get customer district id field
#ifdef SYBASE
    if ( (pymt->in_paym.s_C_D_ID = (CS_TINYINT)atoi(commandBlock->get_C_D_ID())) ==
0)
    {
        doPaymentErrorPage(html,ERR_INVALID_C_D_ID,commandBlock,txnHandle);
        return OK;
    }
#else
    if ( (pymt->in_paym.s_C_D_ID = atoi(commandBlock->get_C_D_ID())) == 0)
    {
        doPaymentErrorPage(html,ERR_INVALID_C_D_ID,commandBlock,txnHandle);
        return OK;
    }
#endif
#ifdef DB2
    pymt->in_paym.s_H_AMOUNT = (float)atof(commandBlock->get_AMT_PAID());
#else
    if(!copyInMoney32(commandBlock->get_AMT_PAID(),&pymt->in_paym.s_H_AMOUNT))
    {

    doPaymentErrorPage(html,ERR_INVALID_PAYMENT_AMOUNT,commandBlock,txnHa
ndle);
        return OK;
    }
#endif
    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Payment
Results</TITLE></HEAD>\r\n"
                "<BODY><FORM ACTION=\""
                APP_NAME
                "\" METHOD=\"GET\">\r\n");
    html+=appendButtons(html);

    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"</FORM><CENTER><H3>Payment</H3></CENTER>");

    DEBUGMSG("pymt executing COM interface function,"<<endl<<
        "pymt c_id: " <<  pymt->in_paym.s_C_ID << endl <<
        "pymt w_id: " <<  pymt->in_paym.s_W_ID << endl <<
        "pymt d_id: " <<  pymt->in_paym.s_D_ID << endl);
    //assume failure
    pymt->out_paym.s_transtatus = -1;

    DEBUGMSG("paym in,"<<endl<<
        "s_W_ID "<<pymt->in_paym.s_W_ID<<endl<<
        "s_C_W_ID "<<pymt->in_paym.s_C_W_ID<<endl<<
        "s_H_AMOUNT "<<pymt->in_paym.s_H_AMOUNT<<endl<<
        "s_D_ID "<<(int)pymt->in_paym.s_D_ID<<endl<<
        "s_C_D_ID "<<(int)pymt->in_paym.s_C_D_ID<<endl<<
        "s_C_ID "<<pymt->in_paym.s_C_ID<<endl<<
        "s_W_ID "<<pymt->in_paym.s_W_ID<<endl<<
        "s_C_LAST "<<pymt->in_paym.s_C_LAST<<endl);

    HRESULT hres=0;
    if (txnHandle->comInterface.size > maxDataSize)
    {
        ERRORMSG("[PY]txnHandle->comInterface.size "<<txnHandle->comInterface.size);
    }
    try
    {
        hres = txnHandle->comInterface.comHandle->doPayment(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);
    }
    catch(...)
    {
        html+=sprintf(html,"ERROR : Com Stock call caused exeception to
occur.</PRE></BODY></HTML>");
        ERRORMSG("COM+ exception [PY]txnHandle->comInterface.size "<<txnHandle-
>comInterface.size);
        return OK;
    }

    if(FAILED(hres))
    {
        html+=sprintf(html,"ERROR : pymt com call failed,
rc:%x</PRE></BODY></HTML>",hres);
        ERRORMSG("ERROR : pymt com call failed, rc:"<<hres<<endl);
        DEBUGMSG("ERROR : pymt com call failed, rc:"<<hres<<endl);
        return OK;
    }

    DEBUGMSG("pymt executed OK,"<<endl<<
        "pymt c_id: " <<  pymt->in_paym.s_C_ID << endl <<
        "pymt w_id: " <<  pymt->in_paym.s_W_ID << endl <<
        "pymt d_id: " <<  pymt->in_paym.s_D_ID << endl);
    txnHandle->comInterface.comHandle->doSetComplete();
    pymt = (paym_wrapper *)txnHandle->comInterface.txnBuffer;

    //get return code
    int rc = pymt->out_paym.s_transtatus;
    if( rc != OK)
    {
        html+=displayStatus(html,rc);
        appendText(&html,"</PRE></BODY></HTML>");
        return OK;
    }
//  appendText(&html, "<BR><PRE>\r\n");
//  appendText(&html,"        1       2       3       4       5       6       7       8<BR>");
//
    appendText(&html,"12345678901234567890123456789012345678901234567890123456789012345678901234
5678901234567890123456789012345678901234567890<BR>");

    //start creating result body
    appendText(&html, "<BR><PRE>\r\n"
                    "Date: ");

    appendText(&html,pymt->out_paym.s_H_DATE_time,18,1);
    appendText(&html, "<BR>"
                "Warehouse: ");
    appendText(&html,itoa(pymt->in_paym.s_W_ID,buffer,10),6+24,1);
    appendText(&html,"District: ");
    appendText(&html,itoa(pymt->in_paym.s_D_ID,buffer,10),2,1);
    appendText(&html,"<BR>");

    //print out warehouse and district information
    appendText(&html,pymt->out_paym.s_W_STREET_1,STREET_LEN+21,1);
    appendText(&html,pymt->out_paym.s_D_STREET_1,STREET_LEN,1);
    appendText(&html,"<BR>");

    appendText(&html,pymt->out_paym.s_W_STREET_2,STREET_LEN+21,1);
    appendText(&html,pymt->out_paym.s_D_STREET_2,STREET_LEN,1);
    appendText(&html,"<BR>");

    appendText(&html,pymt->out_paym.s_W_CITY,CITY_LEN+1,1);
    appendText(&html,pymt->out_paym.s_W_STATE,STATE_LEN+1,1);
    copyOutZip(buffer,pymt->out_paym.s_W_ZIP);
    appendText(&html,buffer);

    appendText(&html,pymt->out_paym.s_D_CITY,CITY_LEN+1,1);
    appendText(&html,pymt->out_paym.s_D_STATE,STATE_LEN+1,1);
    copyOutZip(buffer,pymt->out_paym.s_D_ZIP);
    appendText(&html,buffer);

    //print out customer information
    appendText(&html,"<BR> <BR>Customer: ");
    appendText(&html,itoa(pymt->out_paym.s_C_ID,buffer,10),5+1,1);

    appendText(&html,"Cust-Warehouse: ");
    appendText(&html,itoa(pymt->in_paym.s_C_W_ID,buffer,10),6+1,1);

    appendText(&html,"Cust-District: ");
    appendText(&html,itoa(pymt->in_paym.s_C_D_ID,buffer,10));

    //add customer information
    appendText(&html,"<BR>Name:   ");
    appendText(&html,pymt->out_paym.s_C_FIRST,FIRST_NAME_LEN+1,1);
```

```
    appendText(&html,pymt->out_paym.s_C_MIDDLE,INITIALS_LEN+1,1);
    DEBUGMSG("Last name:"<<pymt->out_paym.s_C_LAST<<endl);
    appendText(&html,pymt->out_paym.s_C_LAST,LAST_NAME_LEN+5,1);

    appendText(&html,"Since: ");
#ifdef ORACLE
    appendText(&html,pymt->out_paym.c_since_d);
#elif DB2
    appendText(&html,pymt->out_paym.s_C_SINCE_time,18,1);
#elif SYBASE
    appendText(&html,pymt->out_paym.c_since);
#endif
    appendText(&html,"<BR>");
    appendSpaces(&html,8);

    appendText(&html,pymt->out_paym.s_C_STREET_1,STREET_LEN+20,1);
    appendText(&html," Credit: ");
    appendText(&html,pymt->out_paym.s_C_CREDIT);

    appendText(&html,"<BR>");
    appendSpaces(&html,8);

    appendText(&html,pymt->out_paym.s_C_STREET_2,STREET_LEN+21,1);
    appendText(&html,"%Disc: ");
    html+=sprintf(html,"%2.2lf",pymt->out_paym.s_C_DISCOUNT);

    appendText(&html,"<BR>");
    appendSpaces(&html,8);

    appendText(&html,pymt->out_paym.s_C_CITY,CITY_LEN+1,1);

    appendText(&html,pymt->out_paym.s_C_STATE,STATE_LEN+1,1);

    copyOutZip(buffer,pymt->out_paym.s_C_ZIP);
    appendText(&html,buffer,15,1);

    appendText(&html,"Phone: ");
    copyOutPhone(buffer,pymt->out_paym.s_C_PHONE);
    appendText(&html,buffer);

    appendText(&html," <BR> <BR>Amount Paid: $");
    html+=sprintf(html,"%-9.2lf",pymt->in_paym.s_H_AMOUNT);

    appendText(&html," New Cust-Balance: $");
    html+=sprintf(html,"%-9.2lf",pymt->out_paym.s_C_BALANCE);

    appendText(&html,"<BR>Credit Limit: $");
    html+=sprintf(html,"%-9.2lf",pymt->out_paym.s_C_CREDIT_LIM);

    appendText(&html," <BR> <BR>Cust-Data: ");
    if(pymt->out_paym.s_C_CREDIT[0] == 'B' && pymt->out_paym.s_C_CREDIT[1] == 'C')
    {
        appendCustData(&html,pymt->out_paym.s_C_DATA);
        appendText(&html," <BR>");
    }
    else
        appendText(&html," <BR> <BR> <BR>");

    html+=displayStatus(html,rc);
    appendText(&html,"</PRE></BODY></HTML>");

    return OK;
}
/*
*****************************************************************
** Name            : doPaymentErrorPage
** Description     :
**                   append payment error body
** Parameters      :
**                   char *      html page result
**                   char *      error message
**                   htmlPhraser * command block
** Returns         :
**                   int - return code
** Comments        :
**
*****************************************************************
*/

int doPaymentErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=htmlPage;
    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Payment</TITLE></HEAD>\r\n"
                     "<BODY><FORM ACTION=\""
                     APP_NAME
                     "\" METHOD=\"GET\">\r\n"
                     "<CENTER><H3>Please Fill In Payment Form.</H3></CENTER> <BR>\r\n"
                     "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                     CMD_TXN_ID
                     "\" VALUE=\""
                     CMD_PYMT
                     "\">");
    html+=appendHiddenFields(html,txnHandle);
    appendText(&html,"<BR><PRE>\r\n"
                     "Date:<BR>"
                     "Warehouse: ");
    char buffer[15];
    appendText(&html,itoa(txnHandle->w_id,buffer,10));

    appendSpaces(&html,10);
    appendText(&html,"District: <INPUT NAME=\""
                     CMD_D_ID
                     "\" SIZE=1>\r\n<BR>"
                     "<BR> <BR> <BR> <BR>"
                     "Customer: "
                     "<INPUT NAME=\""
                     CMD_C_ID
                     "\" SIZE=5>"
                     "   "
                     "Cust-Warehouse: "
                     "<INPUT NAME=\""
                     CMD_C_W_ID
                     "\" SIZE=6>"
                     "   "
                     "Cust-District: "
                     "<INPUT NAME=\""
                     CMD_C_D_ID
                     "\" SIZE=1><BR>"
                     "Name:  <INPUT NAME=\""
                     CMD_C_NAME
                     "\" SIZE=20>");
    appendText(&html,"                          Since: <BR>"
                     "                                 "
                     "                          Credit: <BR>"
                     "                            "
                     "                          %Disc: <BR>"
                     "Amount Paid:               "
                     "<INPUT NAME=\""
                     CMD_AMT_PAID
                     "\" SIZE=10>"
                     "              "
                     "New Cust-Balance:<BR>"
                     "Credit Limit:<BR> <BR> <BR> Cust-Data:<BR> <BR> <BR> <BR> ");
    appendText(&html,message);
    appendText(&html,"</PRE>");

    return OK;
}
/*
*****************************************************************
** Name            : doOrderStatusForm
** Description     :
**                   HTML orderStatus page entry point
** Parameters      :
**                   htmlPhrasercommand block
**                   char *      html result page
** Returns         :
**                   int - return code
** Comments        :
**
*****************************************************************
*/

int doOrderStatusForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=txnHandle->htmlPage;

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Order
Status</TITLE></HEAD>\r\n"
                     "<BODY><FORM ACTION=\""
                     APP_NAME
                     "\" METHOD=\"GET\">\r\n"
                     "<CENTER><H3>Please Fill In Order Status Form.</H3></CENTER>
<BR>\r\n"
                     "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                     CMD_TXN_ID
                     "\" VALUE=\""
                     CMD_ORDS
                     "\">"
                     "<BR> ");
    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"<PRE>\r\n"
                     "Warehouse: ");
    char buffer[15];
    appendText(&html,itoa(txnHandle->w_id,buffer,10));

    appendText(&html,"              District: <INPUT NAME=\""
                     CMD_D_ID
                     "\" SIZE=1>\r\n<BR>"
                     "Customer: "
                     "<INPUT NAME=\""
                     CMD_C_ID
                     "\" SIZE=5>"
                     "   "
                     "Name: "
                     "<INPUT NAME=\""
                     CMD_C_NAME
                     "\" SIZE=20><BR>"
                     "Cust-Balance: <BR>"
                     "Order-Number:        Entry-Date:                  Carrier-Number<BR>"
                     "Supply-W     Item-Num     Oty          Amount          Delivery<BR></PRE>");

    appendText(&html,"</BODY></HTML>");

    return OK;
}

/*
*****************************************************************
** Name            : doOrderStatusResults
** Description     :
**                   HTML orderStatus page entry point
** Parameters      :
**                   htmlPhraser*   command block
**                   char *      html result page
** Returns         :
**                   int - return code
```

```
** Comments    :
**
*********************************************************************
*/

int doOrderStatusResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=txnHandle->htmlPage;
    struct ords_wrapper   *ords = NULL;
    ords = (ords_wrapper *) txnHandle->comInterface.txnBuffer;
    ZeroMemory(ords,sizeof(ords_wrapper));

    //set warehouse login id from command blk
    ords->in_ords.s_W_ID = txnHandle->w_id;

    //set district login id from command blk
    if( (ords->in_ords.s_D_ID = atoi(commandBlock->get_D_ID())) == 0)
    {
        doOrderStatusErrorPage(html,ERR_INVALID_D_ID,commandBlock,txnHandle);
        return OK;
    }

    if( (ords->in_ords.s_C_ID = atoi(commandBlock->get_C_ID())) == 0)
    {
        if(*(commandBlock->get_C_NAME()) == NULL)
        {
            //no customer id nor customer last name specified.

    doOrderStatusErrorPage(html,ERR_MISSING_C_ID_OR_CLAST,commandBlock,txnHa
ndle);
            return OK;
        }
        else
            strcpy(ords->in_ords.s_C_LAST,commandBlock->get_C_NAME());
    }
    else
    {
        //make sure that the user only inserted just c_id
        if(*(commandBlock->get_C_NAME()) != NULL)
        {

    doOrderStatusErrorPage(html,ERR_C_ID_OR_CLAST_ONLY,commandBlock,txnHandl
e);
            return OK;
        }
    }

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Order Status
Results</TITLE></HEAD>\r\n"
                    "<BODY><FORM ACTION=\""
                    APP_NAME
                    "\" METHOD=\"GET\">\r\n");
    html+=appendButtons(html);

    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"</FORM>");

    ords->out_ords.s_transtatus = -1;

    DEBUGMSG("ords executing COM interface function,"<<endl<<
        "ords c_id: " <<  ords->in_ords.s_C_ID << endl <<
        "ords w_id: " <<  ords->in_ords.s_W_ID << endl <<
        "ords d_id: " <<  ords->in_ords.s_D_ID << endl);
    HRESULT hres=0;
    if (txnHandle->comInterface.size > maxDataSize)
    {
        ERRORMSG("[OS]txnHandle->comInterface.size "<<txnHandle->comInterface.size);
    }
    try
    {
        hres = txnHandle->comInterface.comHandle->doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);
    }
    catch(...)
    {
        html+=sprintf(html,"ERROR : ords com call caused
exeception.</PRE></BODY></HTML>");
        ERRORMSG("COM+ exception [OS]txnHandle->comInterface.size "<<txnHandle-
>comInterface.size);
        return OK;
    }

    if(FAILED(hres))
    {
        html+=sprintf(html,"ERROR : ords com call failed,
rc:%x</PRE></BODY></HTML>",hres);
        ERRORMSG("ERROR : ords com call failed, rc:"<<DEBUGADDRESS(hres));
        DEBUGMSG("ERROR : ords com call failed, rc:"<<DEBUGADDRESS(hres));
        return OK;
    }
    DEBUGMSG("ords executed OK,"<<endl<<
        "ords c_id: " <<  ords->in_ords.s_C_ID << endl <<
        "ords w_id: " <<  ords->in_ords.s_W_ID << endl <<
        "ords d_id: " <<  ords->in_ords.s_D_ID << endl);

    txnHandle->comInterface.comHandle->doSetComplete();

    ords = (ords_wrapper *)txnHandle->comInterface.txnBuffer;
    int rc = ords->out_ords.s_transtatus;

    if( rc != OK)
    {
        html+=displayStatus(html,rc);
        appendText(&html,"</PRE></BODY></HTML>");
        return OK;
    }

    //start creating result body
    appendText(&html,"</FORM><CENTER><H3>Order-Status</H3></CENTER>");
    appendText(&html, "<BR><PRE>\r\n");
//  appendText(&html,"       1       2       3       4       5       6       7       8<BR>");
//
    appendText(&html,"12345678901234567890123456789012345678901234567890
5678901234567890123456789012345678901234567890<BR>");
    appendText(&html, "Warehouse: ");
    char buffer[50];

    appendText(&html,itoa(ords->in_ords.s_W_ID,buffer,10),6+1,1);
    appendText(&html,"District: ");
    appendText(&html,itoa(ords->in_ords.s_D_ID,buffer,10));
    appendText(&html,"<BR>"
                    "Customer: ");
    //get customer id
    appendText(&html,itoa(ords->in_ords.s_C_ID,buffer,10),6+1,1);
    appendText(&html,"Name: ");
    //get first, middle, and last from wrapper
    appendText(&html,ords->out_ords.s_C_FIRST,FIRST_NAME_LEN+1,1);
    appendText(&html,ords->out_ords.s_C_MIDDLE,INITIALS_LEN+1,1);
    appendText(&html,ords->out_ords.s_C_LAST,LAST_NAME_LEN+5,1);

    //get customer balance from wrapper
    appendText(&html,"\r\nCust-Balance: $");
    html+=sprintf(html,"%.2lf",ords->out_ords.s_C_BALANCE);
    //display order number, entry date, and carrier number
    appendText(&html,"<BR> <BR>"
                    "Order-Number ");
    appendText(&html,itoa(ords->out_ords.s_O_ID,buffer,10),11,1);
    appendText(&html,"Entry-Date:");
    appendText(&html,ords->out_ords.s_O_ENTRY_D_time,18,1);
    appendText(&html,"Carrier-Number: ");
    appendText(&html,itoa(ords->out_ords.s_O_CARRIER_ID,buffer,10));

    //add item title columns
    appendText(&html,"<BR>"
                    "Supply-W     "
                    "Item-Id    "
                    "Qty    "
                    "Amount      "
                    "Delivery-Date<BR>");

    //display items
    for (int itemCount=0;itemCount<ords->out_ords.s_ol_cnt;itemCount++)
    {
        appendSpaces(&html,2);

#ifdef DB2
        //get supp w
        appendText(&html,itoa(ords-
>out_ords.item[itemCount].s_OL_SUPPLY_W_ID,buffer,10),11,1);

        //get item num
        appendText(&html,itoa(ords->out_ords.item[itemCount].s_OL_I_ID,buffer,10),11,1);

        //get item oty
        appendText(&html,itoa(ords-
>out_ords.item[itemCount].s_OL_QUANTITY,buffer,10),6,1);

        //get item dollor amount
        html+=sprintf(html,"$%-14.2lf",ords->out_ords.item[itemCount].s_OL_AMOUNT);
#elif ORACLE
        //get supp w
        appendText(&html,itoa(ords-
>out_ords.s_OL_SUPPLY_W_ID[itemCount],buffer,10),11,1);

        //get item num
        appendText(&html,itoa(ords->out_ords.s_OL_I_ID[itemCount],buffer,10),11,1);

        //get item oty
        appendText(&html,itoa(ords->out_ords.s_OL_QUANTITY[itemCount],buffer,10),6,1);

        //get item dollor amount
        html+=sprintf(html,"$%-14.2lf",ords->out_ords.s_OL_AMOUNT[itemCount]/100.0);
#elif SYBASE
        //get supp w
        appendText(&html,itoa(ords-
>out_ords.s_OL_SUPPLY_W_ID[itemCount],buffer,10),11,1);

        //get item num
        appendText(&html,itoa(ords->out_ords.s_OL_I_ID[itemCount],buffer,10),11,1);

        //get item oty
        appendText(&html,itoa(ords->out_ords.s_OL_QUANTITY[itemCount],buffer,10),6,1);

        //get item dollor amount
        html+=sprintf(html,"$%-14.2lf",ords->out_ords.s_OL_AMOUNT[itemCount]);
#endif

        //get delivery date
#ifdef ORACLE
        appendText(&html,ords->out_ords.s_OL_DELIVERY_D_time[itemCount]);
#elif DB2
        appendText(&html,ords->out_ords.item[itemCount].s_OL_DELIVERY_D_time,18,1);
#elif SYBASE
        appendText(&html,ords->out_ords.s_OL_DELIVERY_D_time[itemCount]);
#endif
        appendText(&html," <BR>");

    }
```

```
    //append line breaks if item count is less than 15
    for (int itemCount=0;itemCount < (15-ords->out_ords.s_ol_cnt);itemCount++)
        appendText(&html,"<BR> ");

    html+=displayStatus(html,rc);

    appendText(&html,"</PRE></BODY></HTML>");

    return OK;
}

/*
**********************************************************************
** Name            : doOrderStatusErrorPage
** Description     :
**                 HTML orderStatus error page
** Parameters      :
**                 char *      html page result
**                 char *      error message
**                 htmlPhraser*   command block
**                 TXN_HANDLE*   txn handle
** Returns         :
**                 int - return code
** Comments        :
**
**********************************************************************
*/

int doOrderStatusErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=htmlPage;

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Order
Status</TITLE></HEAD>\r\n"
                    "<BODY><FORM ACTION=\""
                    APP_NAME
                    "\" METHOD=\"GET\">\r\n"
                    "<CENTER><H3>Please Fill In Order Status Form.</H3></CENTER>
<BR>\r\n"
                    "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                    CMD_TXN_ID
                    "\" VALUE=\""
                    CMD_ORDS
                    "\">"
                    "<BR> ");
    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"<PRE>\r\n"
                    "Warehouse: ");
    char buffer[15];
    appendText(&html,itoa(txnHandle->w_id,buffer,10));

    appendText(&html,"                District: <INPUT NAME=\""
                    CMD_D_ID
                    "\" SIZE=1>\r\n<BR>"
                    "Customer: "
                    "<INPUT NAME=\""
                    CMD_C_ID
                    "\" SIZE=5>"
                    "   "
                    "Name: "
                    "<INPUT NAME=\""
                    CMD_C_NAME
                    "\" SIZE=20><BR>"
                    "Cust-Balance: <BR>"
                    "Order-Number:          Entry-Date:                    Carrier-Number<BR>"
                    "Supply-W     Item-Num     Oty          Amount          Delivery <BR>");

    appendText(&html,message);

    appendText(&html,"</PRE></BODY></HTML>");

    return OK;
}

/*
**********************************************************************
** Name            : doDeliveryForm
** Description     :
**                 HTML payment page entry point
** Parameters      :
**                 htmlPhrasercommand block
**                 char *      html result page
** Returns         :
**                 int - return code
** Comments        :
**
**********************************************************************
*/
int doDeliveryForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=txnHandle->htmlPage;

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Delivery</TITLE></HEAD>\r\n"
                    "<BODY><FORM ACTION=\""
                    APP_NAME
                    "\" METHOD=\"GET\">\r\n"
                    "<CENTER><H3>Delivery.</H3></CENTER>\r\n"
                    "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                    CMD_TXN_ID
                    "\" VALUE=\""
                    CMD_DLVY
                    "\">");
    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"<BR> <PRE>"
                    "Warehouse: ");
    char buffer[10];
    appendText(&html,itoa(txnHandle->w_id,buffer,10));

    appendText(&html," <BR> <BR>"
                    "Carrier Number: "
                    "<INPUT NAME=\""
                    CMD_CARRIER_NUM
                    "\" SIZE=1>"
                    "</FORM></PRE>");

    appendText(&html,"</BODY></HTML>");

    return OK;
}

/*
**********************************************************************
** Name            : doDeliveryResults
** Description     :
**                 HTML payment page entry point
** Parameters      :
**                 htmlPhraser*   command block
**                 TXN_HANDLE*   txn handle
** Returns         :
**                 int - return code
** Comments        :
**
**********************************************************************
*/

int doDeliveryResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html = txnHandle->htmlPage;

    //declare delivery structure
    struct dlvy_wrapper        dlvy;

    //set warehouse login id from command blk
    dlvy.in_dlvy.s_W_ID = txnHandle->w_id;

    //set the carrier id from command blk
   if( (dlvy.in_dlvy.s_O_CARRIER_ID = atoi(commandBlock->get_CARRIER_NUM())) == 0)
    {
        doDeliveryErrorPage(html,ERR_INVALID_CARRIER,commandBlock,txnHandle);
        return OK;
    }

    //print title, add hidden fields , txn buttons
    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Delivery
Results</TITLE></HEAD>\r\n<BODY><FORM ACTION=\""
                    APP_NAME
                    "\" METHOD=\"GET\">\r\n");

    html+=appendButtons(html);

    html+=appendHiddenFields(html,txnHandle);

    appendText(&html, "<FORM><CENTER><H3>Delivery</H3></CENTER>");

    //call null db or comm w/ delivery wrapper
    int rc = queueDlvyTxn(dlvy.in_dlvy.s_W_ID,dlvy.in_dlvy.s_O_CARRIER_ID);
    //if we are using the null db, rc will always be ok.  The only time rc != OK is
    //1) unable to queue txn because dlvy queue is full.
    if( rc != OK)
    {
        html+=displayStatus(html,rc);
        appendText(&html,"</PRE></BODY></HTML>\r\n");

        ERRORMSG("ERROR : Unable to queue dlvy txn, rc:"<<rc<<endl);
        return OK;
    }

    //start creating result body
    appendText(&html,"Warehouse: ");

    //get w_id from wrapper
    char buffer[10];
    appendText(&html,itoa(dlvy.in_dlvy.s_W_ID,buffer,10));
    appendText(&html,"<BR> <BR>Carrier Number: ");

    //get carrier_id from wrapper
    appendText(&html,itoa(dlvy.in_dlvy.s_O_CARRIER_ID,buffer,10));
    appendText(&html,"<BR> <BR>Execution Status: Delivery has been queued
</PRE></BODY></HTML>");

    return OK;
}

/*
**********************************************************************
** Name            : doDeliveryErrorPage
** Description     :
**                 HTML payment error page entry point
** Parameters      :
**                 char *      html result page
**                 char *      error message
**                 htmlPhrasercommand block
**                 TXN_HANDLE*   txn handle
**
** Returns         :
**                 int - return code
** Comments        :
**
```

```
*****************************************************************
*/
int doDeliveryErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=htmlPage;

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Delivery</TITLE></HEAD>\r\n"
                "<BODY><FORM ACTION=\""
                APP_NAME
                "\" METHOD=\"GET\">\r\n"
                "<CENTER><H3>Delivery.</H3></CENTER>\r\n"
                "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                CMD_TXN_ID
                "\" VALUE=\""
                CMD_DLVY
                "\">");
    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"<BR> <PRE>"
                "Warehouse: ");
    char buffer[15];
    appendText(&html,itoa(txnHandle->w_id,buffer,10));

    appendText(&html," <BR> <BR>"
                "Carrier Number: "
                "<INPUT NAME=\""
                CMD_CARRIER_NUM
                "\" SIZE=1> <BR>");

    appendText(&html,message);
    appendText(&html,"</PRE></BODY></HTML>");

    return OK;
}



/*
*****************************************************************
** Name            : doStockForm
** Description     :
**              HTML stock page entry point
** Parameters    :
**              htmlPhrasercommand block
**              TXN_HANDLE*   txn handle
** Returns      :
**              int - return code
** Comments     :
**
*****************************************************************
*/

int doStockForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=txnHandle->htmlPage;
    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Stock Level</TITLE></HEAD>\r\n"
                "<BODY><FORM ACTION=\""
                APP_NAME
                "\" METHOD=\"GET\">\r\n"
                "<CENTER><H3>Please Fill In Stock Form.</H3></CENTER> <BR>\r\n"
                "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                CMD_TXN_ID
                "\" VALUE=\""
                CMD_STOK
                "\">");
    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"<PRE>"
                "Warehouse: ");
    char buffer[15];
    appendText(&html,itoa(txnHandle->w_id,buffer,10),6+1,1);
    appendText(&html,"District: ");

    appendText(&html,itoa(txnHandle->d_id,buffer,10));
    appendText(&html," <BR> <BR>"
                "Stock Level Threshold: "
                "<INPUT NAME=\""
                CMD_STK_THRESHOLD
                "\" SIZE=1> <BR> <BR>"
                "Low Stock: <BR>"
                "</PRE>");

    appendText(&html,"</FORM></BODY></HTML>");

    return OK;
}

/*
*****************************************************************
** Name            : doStockResults
** Description     :
**                  HTML stock page entry point
** Parameters    :
**              htmlPhrasercommand block
**              TXN_HANDLE *  handle for this transaction
** Returns      :
**              int - return code
** Comments     :
**
*****************************************************************
*/

int doStockResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    char *html = txnHandle->htmlPage;

    struct stok_wrapper   *stok;
    stok = (stok_wrapper*)txnHandle->comInterface.txnBuffer;
    ZeroMemory(stok,sizeof(stok_wrapper));

    //set warehouse login id from command blk
    stok->in_stok.s_W_ID = txnHandle->w_id;

    //set district login id from command blk
    stok->in_stok.s_D_ID = txnHandle->d_id;

    //set stock level threshold id from command blk
    if( (stok->in_stok.s_threshold = atoi(commandBlock->get_STK_THRESHOLD())) == 0)
    {
        doStockErrorPage(html,ERR_INVALID_THRESHOLD,commandBlock,txnHandle);
        return OK;
    }
    //assume failure, set s_transtatus to err
    stok->out_stok.s_transtatus = INVALID_STATUS;

    //print title, add hidden fields , txn buttons
    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Stock Level
Results</TITLE></HEAD>\r\n"
                "<BODY><FORM ACTION=\""
                APP_NAME
                "\" METHOD=\"GET\">\r\n");

    html+=appendButtons(html);

    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"</FORM>");

    stok->out_stok.s_transtatus = OK;

    DEBUGMSG("stok executing COM interface function,"<<endl<<
        "stok d_id: " <<  stok->in_stok.s_D_ID << endl <<
        "stok w_id: " <<  stok->in_stok.s_W_ID << endl <<
        "stok s_threshold: " <<  stok->in_stok.s_threshold << endl);

    HRESULT hres=0;
    if (txnHandle->comInterface.size > maxDataSize)
    {
        ERRORMSG("[SR]txnHandle->comInterface.size "<<txnHandle->comInterface.size);
    }
    try
    {
        hres = txnHandle->comInterface.comHandle->doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);
    }
    catch(...)
    {
        html+=sprintf(html,"ERROR : Com Stock call caused exeception to
occur.</PRE></BODY></HTML>");
        ERRORMSG("COM+ exception [SR]txnHandle->comInterface.size "<<txnHandle-
>comInterface.size);
        return OK;
    }

    //cast result back to stock structure
    if(FAILED(hres))
    {
        html+=sprintf(html,"ERROR : stok com call failed,
rc:%ld</PRE></BODY></HTML>",hres);
        ERRORMSG("ERROR : stok com call failed, rc:"<<DEBUGADDRESS(hres)<<endl);
        DEBUGMSG("ERROR : stok com call failed, rc:"<<DEBUGADDRESS(hres)<<endl);
        return OK;
    }

    DEBUGMSG("stok executed OK,"<<endl<<
        "stok d_id: " <<  stok->in_stok.s_D_ID << endl <<
        "stok w_id: " <<  stok->in_stok.s_W_ID << endl <<
        "stok s_threshold: " <<  stok->in_stok.s_threshold << endl);
    try
    {
        txnHandle->comInterface.comHandle->doSetComplete();
    }
    catch(...)
    {
        ERRORMSG("txnHandle address:"<<hex<<txnHandle<<
            "txnHandle->comInterface.comHandle:"<<hex<<txnHandle-
>comInterface.comHandle<<
            "txnHandle->comInterface.txnBuffer:"<<hex<<(void *)txnHandle-
>comInterface.txnBuffer<<endl);


        ERRORMSG("Com Stock setComplete call caused exeception to occur."<<endl);
        html+=sprintf(html,"ERROR : Com Stock setComplete call caused exeception to
occur.</PRE></BODY></HTML>");
        return OK;
    }

    stok = (stok_wrapper *)txnHandle->comInterface.txnBuffer;
    int rc = stok->out_stok.s_transtatus;
    if(rc != OK)
    {
        html+=displayStatus(html,rc);
        appendText(&html,"</PRE></BODY></HTML>");
        return OK;
    }

    //start creating result body
    appendText(&html,"<FORM><CENTER><H3>Stock-Level</H3></CENTER>");
    appendText(&html, "<BR><PRE>\r\n"
```

```
    "Warehouse: ");

    //get w_id from wrapper
    char buffer[10];
    appendText(&html,itoa(stok->in_stok.s_W_ID,buffer,10),6+1,1);

    appendText(&html,"District: ");
    appendText(&html,itoa(stok->in_stok.s_D_ID,buffer,10));

    appendText(&html," <BR> <BR>"
                "Stock Level Threshold: ");
    appendText(&html,itoa(stok->in_stok.s_threshold,buffer,10));

    appendText(&html," <BR> <BR>"
                "Low Stock: ");
    appendText(&html,itoa(stok->out_stok.s_low_stock,buffer,10));
    appendText(&html," <BR> <BR>");

    html+=displayStatus(html,rc);
    appendText(&html,"</PRE></BODY></HTML>");

    return OK;
}

/*
**********************************************************************
** Name             : doStockErrorPage
** Description    :
**                 HTML stock page entry point
** Parameters   :
**              htmlPhrasercommand block
**              char *      html result page
**                  char *      query string
**                  tpccHandle *    handle for this transaction
** Returns     :
**                  int - return code
** Comments     :
**
**********************************************************************
*/

int doStockErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
    char *html=htmlPage;

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C Stock Level</TITLE></HEAD>\r\n"
                "<BODY><FORM ACTION=\""
                APP_NAME
                "\" METHOD=\"GET\">\r\n"
                "<CENTER><H3>Please Fill In Stock Form.</H3></CENTER> <BR>\r\n"
                "Submit Transaction <INPUT TYPE=\"submit\" NAME=\""
                CMD_TXN_ID
                "\" VALUE=\""
                CMD_STOK
                "\">");
    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"<PRE>"
                "Warehouse:  ");
    char buffer[15];
    appendText(&html,itoa(txnHandle->w_id,buffer,10));
    appendSpaces(&html,2);
    appendText(&html,"District: ");
    appendText(&html,commandBlock->get_D_ID());
    appendText(&html," <BR> <BR>"
                "Stock Level Threshold: "
                "<INPUT NAME=\""
                CMD_STK_THRESHOLD
                "\" SIZE=1> <BR> <BR>"
                "Low Stock: <BR>");

    appendText(&html,message);


    appendText(&html,"</PRE></FORM></BODY></HTML>");

    return OK;
}



/*
**********************************************************************
** Name             : doExit
** Description    :
**                 HTML exit page entry point
** Parameters   :
**                 htmlPhrasercommand block
**                 char *      html result page
** Returns     :
**                 int - return code
** Comments     :
**
**********************************************************************
*/
int doExit(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle)
{
    return (doLoginForm(commandBlock,txnHandle));
}

/*
**********************************************************************
** Name             : displayStatus
** Description    :
**                 appends status string to the html page
** Parameters   :
**                 char*       html page
**                 int       rc
** Returns     :
**                 amount of characters the function appened
**                 to the html page
** Comments     :
**
**********************************************************************
*/
int displayStatus(char *htmlPage,int rc)
{
    char *html = htmlPage;

    appendText(&html,"");

    switch (rc)
    {
    case OK:
        appendText(&html,"Execution Status: Transaction Committed",50,1);
        break;
    case INVALID_ITEM:
        appendText(&html,"Execution Status: Item number is not valid",50,1);
        break;
    case INVALID_STATUS:
        appendText(&html,"Execution Status: ERROR: Rollback INVALID_STATUS",50,1);
        break;
    case INVALID_COM_STATUS:
        appendText(&html,"Execution Status: ERROR: Rollback COM FAILURE",50,1);
        break;
    case ERR_DLVY_QUEUE_FULL:
        appendText(&html,"Execution Status: ERROR: Rollback DLVY QUEUE FULL",50,1);
        break;
    default:
        appendText(&html,"Execution Status: ERROR: Rollback",50,1);

    };

    appendText(&html,"     ");

    return (int)(html - htmlPage);
}
/*
**********************************************************************
** Name             : appendButtons
** Description    :
**                 append hidden field to recognize user after login
**
** Parameters   :
**                 *htmlPage           html result page
**                 *TXN_HANDLE         txn handle
** Returns     :
**                 int             amount of characters the function appened
**                                 to the html page
** Comments     :
**
**********************************************************************
*/
int appendHiddenFields(char *htmlPage,TXN_HANDLE *txnHandle)
{
    char *html = htmlPage;
    char buffer[15];

    appendText(&html,"<INPUT TYPE=\"hidden\" NAME=\""
                    CMD_TERM_ID
                    "\" VALUE=\"");
    appendText(&html,itoa(txnHandle->term_id,buffer,10));
    appendText(&html,"\">\r\n");

    return (int)(html-htmlPage);

}
/*
**********************************************************************
** Name             : appendButtons
** Description    :
**                 appends buttons transaction buttons to result page
** Parameters   :
**                 *htmlPage
**
** Returns     :
**                 amount of characters the function appened
**                 to the html page
** Comments     :
**
**********************************************************************
*/
int appendButtons(char *htmlPage)
{
    char *html = htmlPage;

    appendText(&html,"<INPUT TYPE=\"submit\" NAME=\""
                    CMD_TXN_ID
                    "\" VALUE=\""
                    CMD_NORD
                    "\">\r\n"
                    "<INPUT TYPE=\"submit\" NAME=\""
                    CMD_TXN_ID
                    "\" VALUE=\""
                    CMD_PYMT
                    "\">\r\n"
                    "<INPUT TYPE=\"submit\" NAME=\""
                    CMD_TXN_ID
                    "\" VALUE=\""
```

```
                CMD_ORDS
                "\">\r\n"
                "<INPUT TYPE=\"submit\" NAME=\""
                CMD_TXN_ID
                "\" VALUE=\""
                CMD_DLVY
                "\">\r\n"
                "<INPUT TYPE=\"submit\" NAME=\""
                CMD_TXN_ID
                "\" VALUE=\""
                CMD_STOK
                "\">\r\n"
                "<INPUT TYPE=\"submit\" NAME=\""
                CMD_TXN_ID
                "\" VALUE=\""
                CMD_EXIT
               "\">\r\n <BR>");

    return (int)(html - htmlPage);
}

/*
**********************************************************************
** Name            : appendItems
** Description     :
**                 appends items to new order and order status page
** Parameters      :
**                 *htmlPage               html result page
**                 short                   items to append
**                 short                   item CMD id start
**
** Returns         :
**                 amount of characters the function appened
**                 to the html page
** Comments        :
**
**********************************************************************
*/
int appendItems(char *htmlPage,short itemCount,short cmdIDStart)
{
    char *html = htmlPage;
    char numBuffer[MAX_INT_BUFFER];

    for(int item=0;item < itemCount;item++)
    {
        appendText(&html,"<BR> <INPUT NAME=\"");
        appendText(&html,itoa(cmdIDStart++,numBuffer,10));
        appendText(&html,"\" SIZE=6> <INPUT NAME=\"");
        appendText(&html,itoa(cmdIDStart++,numBuffer,10));
        appendText(&html,"\" SIZE=6>                          <INPUT NAME=\"");
        appendText(&html,itoa(cmdIDStart++,numBuffer,10));
        appendText(&html,"\" SIZE=2>\r\n");
    }

    return (int)(html - htmlPage);
}

/*
**********************************************************************
** Name            : dlvyThreadEntry
** Description     :
**                 dlvy thread worker entry point
** Parameters      :
**
** Returns         :
**
** Comments        :
**                 All dlvy threads created by initDly enter at
**                 this point.  They must first make a connection
**                 to the database, then go to sleep.
**
**                 Main isapi threads control dlvy worker semaphore
**                 and signal when a dlvy txn is queued.
**
**********************************************************************
*/

void dlvyThreadEntry(void *)
{
    int     rc = 0;

    DEBUGMSG("dlvyThread " << GetCurrentThreadId() << " entered dlvyThreadEntry,
calling db_connect to db:" << dbName << endl);

    void *connectHandle;
    //connect to database.
    DEBUGMSG("ptr created. calling db_connect to db:" << dbName << endl);
    ERRORMSG("ptr created. calling db_connect to db:" << dbName << endl);
    rc = db_connect(dbName,&connectHandle);

    if(rc != OK)
    {
        ERRORMSG("dlvyThread " << GetCurrentThreadId() <<" unable to connect to
database, rc:" << rc << endl);
        DEBUGMSG("dlvyThread " << GetCurrentThreadId() <<" unable to connect to
database, rc:" << rc << endl);
        return;
    }

    DEBUGMSG("dlvyThread " << GetCurrentThreadId() << " connect to db:" << dbName
<< " successful" << endl);

    FILE *dlvyLog = NULL;
    char logFileName[MAX_STRING_LEN] = {NULL};

    EnterCriticalSection(&isapiLock);
    //open dlvy log file for this thread
    sprintf(logFileName,"%s\\del_%d.txt",dlvyLogPath,dlvyThreadID);
    dlvyLog = fopen(logFileName,"w");
    if(!dlvyLog)
    {
        ERRORMSG("dlvyThread " << GetCurrentThreadId() << " unable to open dlvy log "
                << dlvyLogPath << "\\del_" << dlvyThreadID << endl);
        DEBUGMSG("dlvyThread " << GetCurrentThreadId() << " unable toopen dlvy log "
                << dlvyLogPath << "\\del_" << dlvyThreadID << endl);
        return;
    }

    //increment the global dlvy thread id
    dlvyThreadID++;

    LeaveCriticalSection(&isapiLock);

    DEBUGMSG("dlvyThread " << GetCurrentThreadId() <<" dlvy log file name: " <<
logFileName << " open." << endl);

    HANDLE   workerHandles[2];                      //handle array to store event to wait on

    struct  DLVYQUEUEDATA     dlvyQueueData;        //dlvy queue struct to store queued
txn
    struct    dlvy_wrapper    dlvyTxn;              //dlvy wrapper of db2 structs

    struct  _timeb              endQueueTime;       //time stamp to queue removal time
    struct  _timeb              endProcessTime;         //time stamp for end process time

    char  orderIDs[MAX_STRING_LEN] = {NULL};          //string to store oids for each
district
    int     bytesWritten =  0;
    int     dlvyCount  = 0;

    DEBUGMSG("dlvyThread entering work loop" << endl);

    //successful, while true
    while(true)
    {
//      try
//      {
            DEBUGMSG("dlvyThread initializing wait handles" << endl);

            //wait for both program exit AND if there is work to do
            workerHandles[0] = dlvyThreadDone;
            workerHandles[1] = dlvyThreadSemaphore;

            DEBUGMSG("dlvyThread going to sleep waiting for wrk" << endl);

            rc = WaitForMultipleObjects(2,&workerHandles[0],FALSE,INFINITE);

            DEBUGMSG("dlvyThread awake, checking wake condition" << endl);

            if(rc == WAIT_OBJECT_0)
                break;
            else if(rc == (WAIT_OBJECT_0+1) )
            {
                DEBUGMSG("dlvyThread awake, wake condition of dlvyThreadSemaphore" <<
endl);
            }

            DEBUGMSG("dlvyThread trying to enter critical section" << endl);

            EnterCriticalSection(&dlvyQueueLock);

            DEBUGMSG("dlvyThread entered critical section" << endl);

            //remove queued dlvy txn
            dlvyQueueData.enqueueTime.time     =
    dlvyQueue[dlvyBufferThreadIndex].enqueueTime.time;
            dlvyQueueData.enqueueTime.millitm  =
    dlvyQueue[dlvyBufferThreadIndex].enqueueTime.millitm;
            dlvyQueueData.in_s_0_CARRIER_ID     =
    dlvyQueue[dlvyBufferThreadIndex].in_s_0_CARRIER_ID;
            dlvyQueueData.warehouse             =
    dlvyQueue[dlvyBufferThreadIndex].warehouse;

            DEBUGMSG("dlvyThread removed dlvy:" << dlvyCount << ",w_id:" <<
dlvyQueueData.warehouse
                            << " carrier_id:" << dlvyQueueData.in_s_0_CARRIER_ID << endl);

            DEBUGMSG("dlvyThread removed dlvy in queue index: "
<<dlvyBufferThreadIndex<< " w_id: " << dlvyQueueData.warehouse
                                << " carrier_id: " << dlvyQueueData.in_s_0_CARRIER_ID << endl);

            //increment the number of free slots
            dlvyBufferFreeSlots++;

            //increment the thread index to next slot in dlvy queue
            dlvyBufferThreadIndex++;

            DEBUGMSG("dlvyThread incremented amount of free slots:" <<
dlvyBufferFreeSlots << " and thread index:" <<
                                dlvyBufferThreadIndex << endl);

            //check if we reached the end of dlvy queue, if so, reset back index back to 0
            if(dlvyBufferThreadIndex == dlvyQueueLen)
            {
                DEBUGMSG("dlvyThread  reset dlvyBufferThreadIndex to 0, current
dlvyBufferThreadIndex:" << dlvyBufferThreadIndex
                        << " free slots:"<<dlvyBufferFreeSlots<<endl);
                ERRORMSG("dlvyThread  reset dlvyBufferThreadIndex to 0, current
dlvyBufferThreadIndex:" << dlvyBufferThreadIndex
                        << " free slots:"<<dlvyBufferFreeSlots<<endl);
```

```
            dlvyBufferThreadIndex=0;
        }

        DEBUGMSG("dlvyThread releasing critical section" << endl);

        LeaveCriticalSection(&dlvyQueueLock);

        //take enqueue time
        _ftime(&endQueueTime);

        DEBUGMSG("dlvyThread executing txn w_id:" << dlvyQueueData.warehouse
              << " carrier_id:" << dlvyQueueData.in_s_0_CARRIER_ID << endl);

        //prepare to call database
        dlvyTxn.in_dlvy.s_O_CARRIER_ID=  dlvyQueueData.in_s_0_CARRIER_ID;
        dlvyTxn.in_dlvy.s_W_ID         =  dlvyQueueData.warehouse;
#ifdef SYBASE
        dlvyTxn.in_dlvy.s_D_ID         =  1;
#endif
        dlvyTxn.out_dlvy.s_transtatus=  OK;

        //increment dlvy count
        dlvyCount++;

        DEBUGMSG("dlvyThread %d calling dlvy txn" << rc << endl);

        //call dvly txn
        rc = dlvyCall(&dlvyTxn,connectHandle);

        _ftime(&endProcessTime);

        rc = dlvyTxn.out_dlvy.s_transtatus;

#ifdef ORACLE
        DEBUGMSG("dlvy txn response time:"<<
              ((((endProcessTime.time - endQueueTime.time)*1000)+
                (endProcessTime.millitm - endQueueTime.millitm))/1000.0)<<
              " w_id:"<<dlvyTxn.in_dlvy.s_W_ID<<" carrier:"
<<dlvyTxn.in_dlvy.s_O_CARRIER_ID<<
              " rc: "<< rc <<endl);
#elif DB2
        DEBUGMSG("dlvy txn response time:"<<
              ((((endProcessTime.time - endQueueTime.time)*1000)+
                (endProcessTime.millitm - endQueueTime.millitm))/1000.0)<<
              " w_id:"<<dlvyTxn.in_dlvy.s_W_ID<<" carrier:"
<<dlvyTxn.in_dlvy.s_O_CARRIER_ID<<
              " deadLocks:"<<dlvyTxn.out_dlvy.deadlocks<<" rc: "<< rc <<endl);
#elif SYBASE
        DEBUGMSG("dlvy txn response time:"<<
              ((((endProcessTime.time - endQueueTime.time)*1000)+
                (endProcessTime.millitm - endQueueTime.millitm))/1000.0)<<
              " w_id:"<<dlvyTxn.in_dlvy.s_W_ID<<" carrier:"
<<dlvyTxn.in_dlvy.s_O_CARRIER_ID<<
              " rc: "<< rc <<endl);
#endif
        DEBUGMSG("dlvyThread dlvy s_transtatus:" << rc << endl);

        if(rc == OK)
        {
            bytesWritten=0;
            char *buffer = orderIDs;

#ifdef SYBASE
            for(int districtIndex=1;districtIndex <=
DISTRICTS_PER_WAREHOUSE;districtIndex++)
            {
                if(dlvyTxn.out_dlvy.s_O_ID[districtIndex] == 0)
                    bytesWritten = sprintf(buffer,"\nD_ID %d had no new orders",districtIndex);
                else
                    bytesWritten = sprintf(buffer,"%d ",dlvyTxn.out_dlvy.s_O_ID[districtIndex]);


                buffer+=bytesWritten;
            }
#else
            for(int districtIndex=0;districtIndex <
DISTRICTS_PER_WAREHOUSE;districtIndex++)
            {
                if(dlvyTxn.out_dlvy.s_O_ID[districtIndex] == 0)
                    bytesWritten = sprintf(buffer,"\nD_ID %d had no new orders",districtIndex);
                else
                    bytesWritten = sprintf(buffer,"%d ",dlvyTxn.out_dlvy.s_O_ID[districtIndex]);


                buffer+=bytesWritten;
            }
#endif
        }
        else
            sprintf(orderIDs,"/nDelivery transaction failed");


        fprintf(dlvyLog,DELIVERY_LOG_SUCCESS_STR,
                    dlvyCount,
                    dlvyQueueData.enqueueTime.time,
                    dlvyQueueData.enqueueTime.millitm,
                    endQueueTime.time,
                    endQueueTime.millitm,
                    dlvyQueueData.warehouse,
                    dlvyQueueData.in_s_0_CARRIER_ID,
                    orderIDs,
                    endProcessTime.time,
                    endProcessTime.millitm);


        fflush(dlvyLog);
//      }
//      catch(...)
//      {
//          ERRORMSG("ERROR : Unhandled exeception in dlvy thread. Thread
exiting"<<endl);
//          fprintf(dlvyLog,"ERROR : Unhandled exeception in dlvy thread %ld. Thread
exiting.\n",GetCurrentThreadId());
//          fflush(dlvyLog);

//          LeaveCriticalSection(&dlvyQueueLock);
//          throw;
//      }
    }//end while true
}

/*
**********************************************************************
** Name            : queueDlvyTxn
** Description     :
**                   function queues dlvy txn in dlvy queue
** Parameters      :
**                   int       warehouse
**                   short     carrier
** Returns         :
**                   int       error code
** Comments        :
**                     Function will queue dlvy txn if 2 points are true
**                     1) We have room in our dlvy buffer
**                     2) We writing over the end of the queue
**
**********************************************************************
*/

int queueDlvyTxn(int warehouse, short carrier_id)
{
    DEBUGMSG("Taking lock to queue dlvy txn.");

    EnterCriticalSection(&dlvyQueueLock);

    DEBUGMSG("Lock aquired to queue dlvy txn");

    if(dlvyBufferFreeSlots)
    {
        DEBUGMSG("Checking if we are inserting at tail of dlvy queue."<<endl);
        if( dlvyBufferSlotIndex == (dlvyBufferThreadIndex-1))
        {
            ERRORMSG("Error dlvy queue inserting over unserviced queued dlvy txn."<<endl);
            DEBUGMSG("Error dlvy queue inserting over unserviced queued dlvy txn."<<endl);
            LeaveCriticalSection(&dlvyQueueLock);
            return ERR_DLVY_QUEUE_EATING_TAIL;
        }

        DEBUGMSG("free slots dlvy queue:"<<dlvyBufferFreeSlots<<" inserting txn in slot: "
<<dlvyBufferSlotIndex<<
            "w_id: "<<warehouse<<" carrier: "<<carrier_id<<endl);

        dlvyQueue[dlvyBufferSlotIndex].warehouse = warehouse;
        dlvyQueue[dlvyBufferSlotIndex].in_s_0_CARRIER_ID = carrier_id;

        _ftime(&dlvyQueue[dlvyBufferSlotIndex].enqueueTime);

//take lock here

        //decrement the number of free slots in the buffer
        dlvyBufferFreeSlots--;

        //increment the index to the next dlvy queue slot.
        dlvyBufferSlotIndex++;

        DEBUGMSG("dlvy txn queued, slots available in queue:"<<dlvyBufferFreeSlots<<"
queue slot index:"<<dlvyBufferSlotIndex
                <<"w_id:"<<warehouse<<" carrier:"<<carrier_id<<endl);

        DEBUGMSG("dlvy txn queued, slots available in queue: "<<dlvyBufferFreeSlots<<"
queue slot index: "<<dlvyBufferSlotIndex
                <<" w_id: "<<warehouse<<" carrier: "<<carrier_id<<endl);

        if(dlvyBufferSlotIndex == dlvyQueueLen)
        {
            DEBUGMSG("queue slot index hit end of queue, reset to 0, current
index:"<<dlvyBufferSlotIndex<<" free slots:"<<dlvyBufferFreeSlots<<endl);
            ERRORMSG("queue slot index hit end of queue, reset to 0, current
index:"<<dlvyBufferSlotIndex<<" free slots:"<<dlvyBufferFreeSlots<<
                "Thread Worker Queue Index:"<<dlvyBufferThreadIndex<<endl);
            dlvyBufferSlotIndex=0;
        }
        //leave critical section
    }
    else
    {
        //no slots available in dlvy buffer, release critcal section and return an nord-
>in_nord.in_item
        LeaveCriticalSection(&dlvyQueueLock);
        ERRORMSG("dlvy queue buffer full, increase the dlvy queue length."<<endl);
        return ERR_DLVY_QUEUE_FULL;
    }

    LeaveCriticalSection(&dlvyQueueLock);

    //release semaphore to wake thread that there is work
    ReleaseSemaphore(dlvyThreadSemaphore,1,NULL);

    return OK;
}
```

```
/*
********************************************************************
** Name            : doHtml
** Description    :
**                HTML processing page entry point
** Parameters    :
**                txn handle
** Returns      :
**                int - return code
** Comments     :
**
********************************************************************
*/

void doHtml(TXN_HANDLE *txnHandle)
{
    DEBUGMSG("Entered doHtml(), parsing query string:"<< txnHandle->urlString << " into
command block"<< endl);
//  ERRORMSG("Entered doHtml(), parsing query string:"<< txnHandle->urlString << " into
command block"<< endl);
    htmlPhraser  commandBlock(txnHandle->urlString);
    DEBUGMSG("Query string parsed.  command:"<< commandBlock.getCommandId() <<
" user's terminal id:" << commandBlock.get_TERM_ID() << endl);

    int terminalID = atoi(commandBlock.get_TERM_ID());
    int commandID = commandBlock.getCommandId();

    DEBUGMSG("User sent in a terimal id:"<<terminalID<<", checking to see if user has
logged in before"<<endl);
    if(terminalID > 0)
    {
        DEBUGMSG("Terminal id > 0, user has logged in already,
terminalID:"<<terminalID<<" retrieving warehouse district pair"<<endl);
        if(getTerminal(terminalID,txnHandle) != OK)
            return;
        DEBUGMSG("User had valid terminal id, user's login warehouse:"<<txnHandle-
>w_id<<" district:"<<txnHandle->d_id<<endl);
    }
    else
    {
        DEBUGMSG("User did not submit a terminal id or valid terminal id, ensure that the
user is trying to log in."<<endl);
        if( (commandID != TXN_LOGIN) && (commandID != TXN_LOGIN_RESULTS) )
        {
            DEBUGMSG("ERROR : User has not logged in."<<endl);
            ERRORMSG("ERROR : User has not logged in."<<endl);
            sprintf(txnHandle->htmlPage,"ERROR : User has not logged in or did not submit a
valid terminal.");
            return;
        }
        DEBUGMSG("User is in process of logging in, commandID:"<<commandID<<endl);
    }

    DEBUGMSG("Calling html page function:"<<commandBlock.getCommandId()<<endl);
    int rc =
htmlPageFunctions[commandBlock.getCommandId()](&commandBlock,txnHandle);
    DEBUGMSG("Return from html page
function:"<<commandBlock.getCommandId()<<endl);

    return;
}

/*
********************************************************************
** Name            : getTerminal
** Description    :
**                retrieves terminal information based on terminal id
** Parameters    :
**                int          terminal id
**                TERM_HANDLE*txn handle
** Returns      :
**                int - return code
** Comments     :
**
********************************************************************
*/
int getTerminal(int terminal,TXN_HANDLE *txnHandle)
{
//  ERRORMSG(">>getTerminal"<<endl);
    //check to see if terminal id is out of range
    if(terminal >= numUsers)
    {
        //terminal id not valid.
        sprintf(txnHandle->htmlPage,"ERROR : Client does not support more than %d users,
terminal id:%d",numUsers,terminal);
        ERRORMSG("ERROR : Client does not support more than "<<numUsers<<" users,
terminal id:"<<terminal<<endl);
        return ERR;
    }

    //check if terminal id is points to a not in use terminal
    if(!(termArray+terminal)->terminalInUse)
    {
        sprintf(txnHandle->htmlPage,"ERROR : Terminal id given points to a not in use
terminal.");
        ERRORMSG("ERROR :  Terminal id given points to a not in use terminal."<<endl);
        return ERR;
    }

    DEBUGMSG("Storing terminal warehouse, district , and initial term id for
user:"<<terminal<<endl);

    //assign terminal values to txn_handle
    txnHandle->d_id = termArray[terminal].d_id;
    txnHandle->w_id = termArray[terminal].w_id;
    txnHandle->term_id = termArray[terminal].terminalID;

    DEBUGMSG("Users terminal:"<<terminal<<", stored warehouse:"<<txnHandle->w_id<<
            " district:"<<txnHandle->d_id<<" termialID stored:"<<txnHandle->term_id<<endl);

    return OK;
}

/*
********************************************************************
** Name            : assignTerminal
** Description    :
**                assigns terminal index to user
** Parameters    :
**                TERM_HANDLE*txn handle
** Returns      :
**                int - return code
** Comments     :
**
********************************************************************
*/
int assignTerminal(TXN_HANDLE *txnHandle)
{
//  ERRORMSG(">>assignTerminal"<<endl);
    EnterCriticalSection(&termLock);

    //check if terminal array is full.
    if(termNextFree == numUsers)
    {
        LeaveCriticalSection(&termLock);
        return ERR;
    }

    DEBUGMSG("Storing user warehouse:"<<txnHandle->w_id<<" district:"<< txnHandle-
>d_id<<
            " in terminal slot:"<<termNextFree<<endl);

    //store users w_id and d_id
    termArray[termNextFree].d_id = txnHandle->d_id;
    termArray[termNextFree].w_id = txnHandle->w_id;

    //set terminal slot to be in use
    termArray[termNextFree].terminalInUse = true;
    termArray[termNextFree].terminalID = termNextFree;
    //in txn handle, set the terminal id
    txnHandle->term_id = termNextFree;

    //increment to next free terminal.
    termNextFree++;

    DEBUGMSG("User warehouse:"<<txnHandle->w_id<<" district:"<< txnHandle->d_id <<
        " stored in terminal slot:"<<txnHandle->term_id<<" next terminal
free:"<<termNextFree<<endl);

    LeaveCriticalSection(&termLock);

    return OK;
}
```

tpccIsapi.hpp

```
/*
********************************************************************
** Project       : AIX
** Component    : Performance/TPC-W Benchmark
** Name            : tpccIsapi.hpp
** Title       : ISAPI interface for tpcc
********************************************************************
** Copyright (c) 2001,2002 IBM Corporation
** All rights reserved
********************************************************************
** History       :
**                Developed at IBM Austin by the AIX RS/6000
**                performance group.
**
** Comments      :
**
********************************************************************
*/

#ifndef __tpccISAPI_hpp__
#define __tpccISAPI_hpp__

#include <windows.h>
#include <httpext.h>

#include "tpcc.h"
#include "htmlPhraser.h"

#include <iomanip>

#ifdef DB2
    #include <db2tpcc.h>
#endif

#ifdef ORACLE
    #include <oratpcc.h>
#endif

#include <comsvcs.h>

////////////////////////////////////////////////////////////////////
// Terminal struct
```

```
//////////////////////////////////////////////////////////////////
struct TERM_ENTRY
{
    int    terminalID;
    bool  terminalInUse;
    int    w_id;
    short d_id;
};

//////////////////////////////////////////////////////////////////
// COM interface
//////////////////////////////////////////////////////////////////
struct COM_HANDLE
{
    Itpcc_com *comHandle;
    char      *txnBuffer;
    int        size;
};

//////////////////////////////////////////////////////////////////
// TXN handle
//////////////////////////////////////////////////////////////////
struct   TXN_HANDLE
{
    char  htmlPage[MAX_HTML_PAGE_LEN];
    char  htmlHeader[MAX_HTML_HEADER_LEN];
    char  *urlString;

    //user data
    int    w_id;
    int    d_id;
    int    sync_id;
    int    term_id;
    int    conn_id;

    COM_HANDLE comInterface;
};

//////////////////////////////////////////////////////////////////
// Definitions
//////////////////////////////////////////////////////////////////
#define     INVALID_ITEM     100
#define     HEADER              "Content-Type:text/html\r\nContent-Length:
%d\r\nConnection: Keep-Alive\r\n\r\n"
#define     TLS_NULL            0xFFFFFFFF
#define     ACCESS_TIMEOUT      3600000                          //One hour in milli
seconds

#define     DELIVERY_LOG_SUCCESS_STR"--Tran %d Queue %d.%03d Start
%d.%03d\nW_ID: %d CARRIER_ID: %d %s\nend-time: %d.%03d\n"

//////////////////////////////////////////////////////////////////
// Function Prototypes
//////////////////////////////////////////////////////////////////

int initDlvy();
int initTxnHandle(TXN_HANDLE **txnHandle);
int closeTxnHandle(TXN_HANDLE *txnHandle);
int readRegisteryValues();
int getTerminal(int terminal,TXN_HANDLE *txnHandle);
int assignTerminal(TXN_HANDLE *txnHandle);
int getDBInstance();

void doHtml(TXN_HANDLE *txnHandle);
int doLoginForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doLoginResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doNewOrderForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doNewOrderResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doPaymentForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doPaymentResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doOrderStatusForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doOrderStatusResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doDeliveryForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doDeliveryResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doStockForm(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doStockResults(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);
int doExit(htmlPhraser *commandBlock,TXN_HANDLE *txnHandle);

int doLoginErrorPage(char *htmlPage,char *message);
int doNewOrderErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doPaymentErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doOrderStatusErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doDeliveryErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doStockErrorPage(char *htmlPage,char *message,htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);

void dlvyThreadEntry(void *);
int  queueDlvyTxn(int warehouse, short carrier_id);

int appendButtons(char *htmlPage);
int appendItems(char *htmlPage,short itemCount,short cmdIDStart);
int appendHiddenFields(char *htmlPage,TXN_HANDLE *txnHandle);

int displayStatus(char *htmlPage,int rc);

#endif
```

time.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/timeb.h>
#include <time.h>

char *get_time_prefix(char *buffer)
{
    time_t cur_timet;
    char time_str[30];
    int len;

    cur_timet = time(&cur_timet);
    strftime(time_str, 29, "%X", localtime(&cur_timet));

    len = sprintf(buffer, "%s - ",
                  time_str);
    if (len >= 30) {
      sprintf(buffer, "too small: %d\n",
              30, len);
    }
    return(buffer);
}
```

9.2. Transaction Code

pldel.cpp

```
#include "../../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"

extern int sqlfile(FILE* server_logtrans,char *fnam, text *linebuf);
extern void oralogprintf(FILE* server_logtrans,char *format, ...);

extern char OracleHome[256];

#define DMLRETDEL

#define SQLTXT "BEGIN inittpcc.init_del ; END;"

#define SQLTXT1 "DELETE FROM nord WHERE no_d_id = :d_id \
                 AND no_w_id = :w_id and rownum <= 1 \
                RETURNING no_o_id into :o_id "

#define SQLTXT3 "UPDATE ordr SET o_carrier_id = :carrier_id \
                 WHERE o_id = :o_id and o_d_id = :d_id and o_w_id = :w_id \
                 returning o_c_id into :o_c_id"

#define SQLTXT4 "UPDATE ordl \
  SET ol_delivery_d = :cr_date \
  WHERE ol_w_id = :w_id AND ol_d_id = :d_id AND ol_o_id = :o_id \
  RETURNING sum(ol_amount) into :ol_amount "

#define SQLTXT6 "UPDATE cust 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



struct pldelctx {

  ub2 del_d_id_len[NDISTS];
  ub2 del_o_id_len[NDISTS];


  ub2 w_id_len;
  ub2 d_id_len[NDISTS];
  ub2 o_c_id_len[NDISTS];
  ub2 sums_len[NDISTS];
  ub2 carrier_id_len;
  ub2 ordcnt_len;
  ub2 del_date_len;

  int del_o_id[NDISTS];
  int del_d_id[NDISTS];
  int o_c_id[NDISTS];
#ifdef USE_IEEE_NUMBER
  float sums[NDISTS];
#else
  int sums[NDISTS];
#endif
  OCIDate del_date;
  int carrier_id;
  int ordcnt;

  ub4 del_o_id_rcnt;
  ub4 del_d_id_rcnt;
  ub4 o_c_id_rcnt;
  ub4 sums_rcnt;

  int retry;
  OCIStmt *curp1;
  OCIStmt *curp2;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *o_id_bp;
  OCIBind *o_c_id_bp;
  OCIBind *ordcnt_bp;
  OCIBind *sums_bp;
```

```
   OCIBind *del_date_bp;
   OCIBind *carrier_id_bp;
   OCIBind *retry_bp;

   int norow;

};
typedef struct pldelctx pldelctx;

#ifdef DMLRETDEL
struct amtctx {
  int ol_amt[NITEMS];
  sb2 ol_amt_ind[NITEMS];
  ub4 ol_amt_len[NITEMS];
  ub2 ol_amt_rcode[NITEMS];
  int ol_cnt;
};
typedef struct amtctx amtctx;

#endif


tkvcdinit (ora_cn_data_t *ora_SlotDataP)
{
   char sqlfilename[256];
   text stmbuf[SQL_BUF_SIZE];
   FILE *server_logtrans = ora_SlotDataP->server_logtrans;

   pldelctx *pldctx;
   dlvy_wrapper *delP;
   OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
   OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
   dvoid *errhp = ora_SlotDataP->errhp;
   OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
   OCISession *tpcusr = ora_SlotDataP->tpcusr;
   OCIStmt *curi = ora_SlotDataP->curi;


   delP = &ora_SlotDataP->delP;

    pldctx = (pldelctx *) malloc (sizeof(pldelctx));
    DISCARD memset(pldctx,(char)0,(ub4)sizeof(pldelctx));
    ora_SlotDataP->pldctx  = (void *)pldctx;
    /* Initialize */
    DISCARD OCIHandleAlloc(tpcenv, (dvoid**)&pldctx->curp1, OCI_HTYPE_STMT, 0,
                    (dvoid**)0);
    DISCARD sprintf ((char *) stmbuf, SQLTXT);
    DISCARD OCIStmtPrepare(pldctx->curp1, (OCIError *)errhp, stmbuf,
                    (ub4) strlen((char *)stmbuf),
                OCI_NTV_SYNTAX, OCI_DEFAULT);
    DISCARD OCIERROR(errhp,
      OCIStmtExecute(tpcsvc,pldctx->curp1,(OCIError *)errhp,1,0,NULLP(OCISnapshot),
                NULLP(OCISnapshot), OCI_DEFAULT));


    DISCARD OCIHandleAlloc(tpcenv,(dvoid**) &pldctx->curp2, OCI_HTYPE_STMT,
              0, (dvoid**)0);
#if defined(ISO5) || defined(ISO6) || defined(ISO8)
  #if defined(ISO5)
     sprintf(sqlfilename,"%s%s",OracleHome,"/tpcc-kit/benchrun/blocks/tkvcpdel_iso5.sql");
     sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuf);
  #endif
  #if defined(ISO6)
     sprintf(sqlfilename,"%s%s",OracleHome,"/tpcc-kit/benchrun/blocks/tkvcpdel_iso6.sql");
     sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuf);
  #endif
  #if defined(ISO8)
     sprintf(sqlfilename,"%s%s",OracleHome,"/tpcc-kit/benchrun/blocks/tkvcpdel_iso8.sql");
     sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuf);
  #endif
#else
     sprintf(sqlfilename,"%s%s",OracleHome,"/tpcc-kit/benchrun/blocks/tkvcpdel.sql");
     sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuf);
#endif
     DISCARD OCIStmtPrepare(pldctx->curp2, (OCIError *)errhp, stmbuf,
                    (ub4)strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);
     OCIBNDPL(pldctx->curp2, pldctx->w_id_bp , (OCIError *)errhp,":w_id",
         ADR(delP->in_dlvy.s_W_ID), SIZ(int), SQLT_INT,&pldctx->w_id_len);
     OCIBNDPL(pldctx->curp2, pldctx->ordcnt_bp , (OCIError *)errhp,":ordcnt",
            ADR(pldctx->ordcnt), SIZ(int), SQLT_INT,&pldctx->ordcnt_len);

     OCIBNDPL(pldctx->curp2, pldctx->del_date_bp,(OCIError *)errhp,":now",
            ADR(pldctx->del_date), SIZ(OCIDate),SQLT_ODT,&pldctx->del_date_len);

     OCIBNDPL(pldctx->curp2, pldctx->carrier_id_bp , (OCIError *)errhp,
         ":carrier_id", ADR(delP->in_dlvy.s_O_CARRIER_ID), SIZ(int),
            SQLT_INT, &pldctx->carrier_id_len);

     OCIBNDPLA(pldctx->curp2, pldctx->d_id_bp, (OCIError *)errhp,":d_id",
              pldctx->del_d_id, SIZ(int),SQLT_INT, pldctx->del_d_id_len,
              NDISTS, &pldctx->del_d_id_rcnt);
     OCIBNDPLA(pldctx->curp2, pldctx->o_id_bp, (OCIError *)errhp,":order_id",
              pldctx->del_o_id,SIZ(int),SQLT_INT, pldctx->del_o_id_len,NDISTS,
              &pldctx->del_o_id_rcnt);
#ifdef USE_IEEE_NUMBER
     OCIBNDPLA(pldctx->curp2, pldctx->sums_bp, (OCIError *)errhp,":sums",
              pldctx->sums,SIZ(float),SQLT_BFLOAT, pldctx->sums_len,NDISTS,
              &pldctx->sums_rcnt);
#else
     OCIBNDPLA(pldctx->curp2, pldctx->sums_bp, (OCIError *)errhp,":sums",
              pldctx->sums,SIZ(int),SQLT_INT, pldctx->sums_len,NDISTS,
              &pldctx->sums_rcnt);
#endif
     OCIBNDPLA(pldctx->curp2, pldctx->o_c_id_bp, (OCIError *)errhp,":o_c_id",
              pldctx->o_c_id,SIZ(int),SQLT_INT, pldctx->o_c_id_len,NDISTS,
              &pldctx->o_c_id_rcnt);
     OCIBND(pldctx->curp2, pldctx->retry_bp , (OCIError *)errhp,":retry",
           ADR(pldctx->retry), SIZ(int),SQLT_INT);

   return (0);

}

tkvcd (ora_cn_data_t *ora_SlotDataP)
{
   ub4 i;
   FILE *server_logtrans = ora_SlotDataP->server_logtrans;

   pldelctx *pldctx = (pldelctx *)ora_SlotDataP->pldctx;
#ifdef DMLRETDEL /* VMM 1/13/98 */
   amtctx *actx = (amtctx *)ora_SlotDataP->actx;
#endif /* DMLRETDEL */
   dlvy_wrapper *delP = &ora_SlotDataP->delP;
   OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
   OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
   dvoid *errhp = ora_SlotDataP->errhp;
   OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
   OCISession *tpcusr = ora_SlotDataP->tpcusr;
   OCIStmt *curi = ora_SlotDataP->curi;

retry:
     pldctx->w_id_len = sizeof (int);
     pldctx->carrier_id_len = sizeof (int);
   for (i = 0; i < NDISTS; i++)
   {
      pldctx->del_o_id_len[i] = sizeof(int);
      delP->out_dlvy.s_O_ID[i] = 0;
   }
     pldctx->del_date_len = DEL_DATE_LEN;
     DISCARD memcpy(&pldctx->del_date,&delP->in_dlvy.cr_date,sizeof(OCIDate));
     pldctx->retry=0;

     delP->out_dlvy.terror = OCIStmtExecute(tpcsvc,pldctx->curp2,(OCIError
*)errhp,1,0,NULLP(CONST OCISnapshot),
                NULLP(OCISnapshot),OCI_DEFAULT);
     if((delP->out_dlvy.terror != OCI_SUCCESS) && (delP->out_dlvy.terror !=
OCI_NO_DATA))
     {
       DISCARD OCITransRollback(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
       delP->out_dlvy.terror = OCIERROR((OCIError *)errhp,delP->out_dlvy.terror);
       if(delP->out_dlvy.terror == NOT_SERIALIZABLE)
       {
          delP->out_dlvy.retry++;
        goto retry;
       }
       else if (delP->out_dlvy.terror == RECOVERR)
       {
        delP->out_dlvy.retry++;
        goto retry;
       }
       else if (delP->out_dlvy.terror == SNAPSHOT_TOO_OLD)
       {
        delP->out_dlvy.retry++;
        goto retry;
       }
       else
       {
        return -1;
       }
     }
     for (i = 0; i < pldctx->del_o_id_rcnt; i++)
       delP->out_dlvy.s_O_ID[pldctx->del_d_id[i] - 1] = pldctx->del_o_id[i];
     return (0);

}


void tkvcddone (ora_cn_data_t *ora_SlotDataP)

{
   pldelctx *pldctx = (pldelctx *)ora_SlotDataP->pldctx;

     if (pldctx)
     {
       DISCARD OCIHandleFree((dvoid *)pldctx->curp1,OCI_HTYPE_STMT);
       DISCARD OCIHandleFree((dvoid *)pldctx->curp2,OCI_HTYPE_STMT);
       DISCARD free(pldctx);
     }
}
```

plnew.cpp

```
#include "../../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"

#define SQLTXT2 "BEGIN inittpcc.init_no(:idx1arr); END;"

#define NITEMS 15
#define ROWIDLEN 20
#define OCIROWLEN 20

extern void oralogprintf(FILE *server_logtrans,char *format, ...);

extern char OracleHome[256];
extern int sqlfile(FILE *server_logtrans,char *fnam, text *linebuf);

struct newctx {
```

```
  ub2 nol_i_id_len[NITEMS];
  ub2 nol_supply_w_id_len[NITEMS];
  ub2 nol_quantity_len[NITEMS];
  ub2 nol_amount_len[NITEMS];
  ub2 s_quantity_len[NITEMS];
  ub2 i_name_len[NITEMS];
  ub2 i_price_len[NITEMS];
  ub2 s_dist_info_len[NITEMS];
  ub2 ol_o_id_len[NITEMS];
  ub2 ol_number_len[NITEMS];
  ub2 s_remote_len[NITEMS];
  ub2 s_quant_len[NITEMS];
  ub2 ol_dist_info_len[NITEMS];
  ub2 s_bg_len[NITEMS];
  int ol_o_id[NITEMS];
  int ol_number[NITEMS];
#ifdef USE_IEEE_NUMBER
  float s_remote[NITEMS];
#else
  int s_remote[NITEMS];
#endif /* USE_IEEE_NUMBER */
  char s_dist_info[NITEMS][25];
  OCIStmt *curn1;
  OCIBind *ol_i_id_bp;
  OCIBind *ol_supply_w_id_bp;
  OCIBind *i_price_bp;
  OCIBind *i_name_bp;
  OCIBind *s_bg_bp;
  ub4 nol_i_count;
  ub4 nol_s_count;
  ub4 nol_q_count;
  ub4 nol_item_count;
  ub4 nol_name_count;
  ub4 nol_qty_count;
  ub4 nol_bg_count;
  ub4 nol_am_count;
  ub4 s_remote_count;
  OCIStmt *curn2;
  OCIBind *ol_quantity_bp;
  OCIBind *s_remote_bp;
  OCIBind *s_quantity_bp;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *c_id_bp;
  OCIBind *o_all_local_bp;
  OCIBind *o_all_cnt_bp;
  OCIBind *w_tax_bp;
  OCIBind *d_tax_bp;
  OCIBind *o_id_bp;
  OCIBind *c_discount_bp;
  OCIBind *c_credit_bp;
  OCIBind *c_last_bp;
  OCIBind *retries_bp;
  OCIBind *cr_date_bp;
  OCIBind *ol_o_id_bp;
  OCIBind *ol_amount_bp;

  ub2 w_id_len; /* RP - may need to be (ub2), was (sb2) */
  ub2 d_id_len;
  ub2 c_id_len;
  ub2 o_all_local_len;
  ub2 o_ol_cnt_len;
  ub2 w_tax_len;
  ub2 d_tax_len;
  ub2 o_id_len;
  ub2 c_discount_len;
  ub2 c_credit_len;
  ub2 c_last_len;
  ub2 retries_len;
  ub2 cr_date_len;
};

typedef struct newctx newctx;

tkvcninit (ora_cn_data_t *ora_SlotDataP)
{
  char sqlfilename[256];
  text stmbuf[32*1024];
  FILE *server_logtrans = ora_SlotDataP->server_logtrans;

  newctx *nctx;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  dvoid *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;
  nord_wrapper *newP;

  oralogprintf(ora_SlotDataP->server_logtrans,">>tkvcninit\n");
  nctx = (newctx *) malloc (sizeof(newctx));
  DISCARD memset(nctx,(char)0,sizeof(newctx));
  ora_SlotDataP->nctx = (void *)nctx;

  memset(&ora_SlotDataP->globals,(char)0,sizeof(nord_wrapper));
  newP = &ora_SlotDataP->globals;

  nctx->w_id_len = sizeof(newP->in_nord.s_W_ID);
  nctx->d_id_len = sizeof(newP->in_nord.s_D_ID);
  nctx->c_id_len = sizeof(newP->in_nord.s_C_ID);
  nctx->o_all_local_len = sizeof(newP->in_nord.s_all_local);
  nctx->o_ol_cnt_len = sizeof(newP->out_nord.s_O_OL_CNT);
  nctx->w_tax_len = 0;
  nctx->d_tax_len = 0;
  nctx->o_id_len = sizeof(newP->out_nord.s_O_ID);
  nctx->c_discount_len = 0;
  nctx->c_credit_len = 0;
  nctx->c_last_len = 0;
  nctx->retries_len = sizeof(newP->out_nord.retry);
  nctx->cr_date_len = sizeof(newP->in_nord.cr_date);

  /* open first cursor */
  DISCARD OCIERROR((OCIError *)errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&nctx->curn1),
          OCI_HTYPE_STMT, 0, (dvoid**)0));
#if defined(ISO)
  sprintf(sqlfilename,"%s/tpcc-kit/benchrun/blocks/tkvcpnew_iso.sql",OracleHome);
  sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuf);
#else
#if defined(ISO7)
  sprintf(sqlfilename,"%s/tpcc-kit/benchrun/blocks/tkvcpnew_iso7.sql",OracleHome);
  sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuf);
#else
  sprintf(sqlfilename,"%s/tpcc-kit/benchrun/blocks/tkvcpnew.sql",OracleHome);
  sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuf);
#endif
#endif

  DISCARD OCIERROR((OCIError *)errhp,OCIStmtPrepare(nctx->curn1, (OCIError *)errhp,
stmbuf,
          strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

  /* bind variables */
  oralogprintf(ora_SlotDataP->server_logtrans,"--tkvcninit\n");

  OCIBNDPL(nctx->curn1, nctx->w_id_bp, (OCIError *)errhp, ":w_id",ADR(newP-
>in_nord.s_W_ID),SIZ(newP->in_nord.s_W_ID),
          SQLT_INT, &nctx->w_id_len);
  OCIBNDPL(nctx->curn1, nctx->d_id_bp, (OCIError *)errhp, ":d_id",ADR(newP-
>in_nord.s_D_ID),SIZ(newP->in_nord.s_D_ID),
          SQLT_INT, &nctx->d_id_len);
  OCIBNDPL(nctx->curn1, nctx->c_id_bp, (OCIError *)errhp, ":c_id",ADR(newP-
>in_nord.s_C_ID),SIZ(newP->in_nord.s_C_ID),
          SQLT_INT, &nctx->c_id_len);
  OCIBNDPL(nctx->curn1, nctx->o_all_local_bp, (OCIError *)errhp, ":o_all_local",
       ADR(newP->in_nord.s_all_local), SIZ(newP->in_nord.s_all_local),SQLT_INT, &nctx-
>o_all_local_len);
  OCIBNDPL(nctx->curn1, nctx->o_all_cnt_bp, (OCIError *)errhp, ":o_ol_cnt",ADR(newP-
>out_nord.s_O_OL_CNT),
       SIZ(newP->out_nord.s_O_OL_CNT),SQLT_INT, &nctx->o_ol_cnt_len);
  OCIBNDPL(nctx->curn1, nctx->w_tax_bp, (OCIError *)errhp, ":w_tax",ADR(newP-
>out_nord.s_W_TAX),SIZ(newP->out_nord.s_W_TAX),
          SQLT_FLT, &nctx->w_tax_len);
  OCIBNDPL(nctx->curn1, nctx->d_tax_bp, (OCIError *)errhp, ":d_tax",ADR(newP-
>out_nord.s_D_TAX),SIZ(newP->out_nord.s_D_TAX),
          SQLT_FLT, &nctx->d_tax_len);
  OCIBNDPL(nctx->curn1, nctx->o_id_bp, (OCIError *)errhp, ":o_id",ADR(newP-
>out_nord.s_O_ID),SIZ(newP->out_nord.s_O_ID),
          SQLT_INT, &nctx->o_id_len);
  OCIBNDPL(nctx->curn1, nctx->c_discount_bp, (OCIError *)errhp, ":c_discount",
       ADR(newP->out_nord.s_C_DISCOUNT), SIZ(newP-
>out_nord.s_C_DISCOUNT),SQLT_FLT, &nctx->c_discount_len);
  OCIBNDPL(nctx->curn1, nctx->c_credit_bp, (OCIError *)errhp, ":c_credit",newP-
>out_nord.s_C_CREDIT,
       SIZ(newP->out_nord.s_C_CREDIT),SQLT_CHR, &nctx->c_credit_len);
  OCIBNDPL(nctx->curn1, nctx->c_last_bp, (OCIError *)errhp, ":c_last",newP-
>out_nord.s_C_LAST,SIZ(newP->out_nord.s_C_LAST),
          SQLT_STR, &nctx->c_last_len);
  OCIBNDPL(nctx->curn1, nctx->retries_bp, (OCIError *)errhp, ":retry",ADR(newP-
>out_nord.retry),
          SIZ(newP->out_nord.retry),SQLT_INT, &nctx->retries_len);
  OCIBNDPL(nctx->curn1, nctx->cr_date_bp, (OCIError *)errhp, ":cr_date",&newP-
>in_nord.cr_date,
          SIZ(OCIDate), SQLT_ODT, &nctx->cr_date_len);

  OCIBNDPLA(nctx->curn1, nctx->ol_i_id_bp,(OCIError *)errhp,":ol_i_id",newP-
>in_nord.s_OL_I_ID,
          SIZ(newP->in_nord.s_OL_I_ID[0]), SQLT_INT, nctx->nol_i_id_len,NITEMS,&nctx-
>nol_i_count);
  OCIBNDPLA(nctx->curn1, nctx->ol_supply_w_id_bp, (OCIError *)errhp, ":ol_supply_w_id"
       newP->in_nord.s_OL_SUPPLY_W_ID,SIZ(newP-
>in_nord.s_OL_SUPPLY_W_ID[0]),SQLT_INT, nctx->nol_supply_w_id_len,
          NITEMS, &nctx->nol_s_count);
#ifdef USE_IEEE_NUMBER
  OCIBNDPLA(nctx->curn1, nctx->ol_quantity_bp,(OCIError *)errhp,":ol_quantity",
          newP->in_nord.ol_quantity, SIZ(newP->in_nord.ol_quantity[0]),SQLT_FLT,nctx-
>nol_quantity_len,
          NITEMS,&nctx->nol_q_count);

  OCIBNDPLA(nctx->curn1, nctx->i_price_bp,(OCIError *)errhp,":i_price",newP-
>out_nord.s_I_PRICE,SIZ(newP->out_nord.s_I_PRICE[0]),
          SQLT_FLT, nctx->i_price_len, NITEMS, &nctx->nol_item_count);
#else
  OCIBNDPLA(nctx->curn1, nctx->ol_quantity_bp,(OCIError *)errhp,":ol_quantity",
       newP->in_nord.s_OL_QUANTITY, SIZ(newP-
>in_nord.s_OL_QUANTITY[0]),SQLT_INT,nctx->nol_quantity_len,
          NITEMS,&nctx->nol_q_count);
  OCIBNDPLA(nctx->curn1, nctx->i_price_bp,(OCIError *)errhp,":i_price",newP-
>out_nord.s_I_PRICE,SIZ(newP->out_nord.s_I_PRICE[0]),
          SQLT_INT, nctx->i_price_len, NITEMS, &nctx->nol_item_count);
#endif
  OCIBNDPLA(nctx->curn1, nctx->i_name_bp,(OCIError *)errhp,":i_name",newP-
>out_nord.s_I_NAME,
          SIZ(newP->out_nord.s_I_NAME[0]),SQLT_STR, nctx->i_name_len,NITEMS,
          &nctx->nol_name_count);
#ifdef USE_IEEE_NUMBER
  OCIBNDPLA(nctx->curn1, nctx->s_quantity_bp,(OCIError *)errhp,":s_quantity",newP-
>out_nord.s_S_QUANTITY,
          SIZ(newP->out_nord.s_S_QUANTITY[0]), SQLT_FLT,nctx-
>s_quant_len,NITEMS,&nctx->nol_qty_count);
```

```
#else
  OCIBNDPLA(nctx->curn1, nctx->s_quantity_bp,(OCIError *)errhp,":s_quantity",newP-
>out_nord.s_S_QUANTITY,
         SIZ(newP->out_nord.s_S_QUANTITY[0]), SQLT_INT,nctx-
>s_quant_len,NITEMS,&nctx->nol_qty_count);
#endif
  OCIBNDPLA(nctx->curn1, nctx->s_bg_bp,(OCIError *)errhp,":brand_generic",newP-
>out_nord.s_brand_generic,
         SIZ(newP->out_nord.s_brand_generic[0]), SQLT_CHR,nctx-
>s_bg_len,NITEMS,&nctx->nol_bg_count);
#ifdef USE_IEEE_NUMBER
  OCIBNDPLA(nctx->curn1, nctx->ol_amount_bp,(OCIError *)errhp,":ol_amount",newP-
>out_nord.s_OL_AMOUNT,
         SIZ(newP->out_nord.s_OL_AMOUNT[0]),SQLT_FLT, nctx-
>nol_amount_len,NITEMS,&nctx->nol_am_count);
  OCIBNDPLA(nctx->curn1, nctx->s_remote_bp,(OCIError *)errhp,":s_remote",nctx-
>s_remote,
         SIZ(nctx->s_remote[15]),SQLT_FLT, nctx->s_remote_len,NITEMS,&nctx-
>s_remote_count);
#else
  OCIBNDPLA(nctx->curn1, nctx->ol_amount_bp,(OCIError *)errhp,":ol_amount",newP-
>out_nord.s_OL_AMOUNT,
         SIZ(newP->out_nord.s_OL_AMOUNT[0]),SQLT_INT, nctx-
>nol_amount_len,NITEMS,&nctx->nol_am_count);
  OCIBNDPLA(nctx->curn1, nctx->s_remote_bp,(OCIError *)errhp,":s_remote",nctx-
>s_remote,
         SIZ(nctx->s_remote[0]),SQLT_INT, nctx->s_remote_len,NITEMS,&nctx-
>s_remote_count);
#endif

  /* open second cursor */
  DISCARD OCIERROR((OCIError *)errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&nctx-
>curn2),
              OCI_HTYPE_STMT, 0, (dvoid**)0));
  DISCARD sprintf ((char *) stmbuf, SQLTXT2);
  DISCARD OCIERROR((OCIError *)errhp,OCIStmtPrepare(nctx->curn2, (OCIError *)errhp,
stmbuf,
              strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));


  /* execute second cursor to init newinit package */
  {
    int idx1arr[NITEMS];
    OCIBind *idx1arr_bp;
    ub2 idx1arr_len[NITEMS];
    sb2 idx1arr_ind[NITEMS];
    ub4 idx1arr_count;
    ub2 idx;

    for (idx = 0; idx < NITEMS; idx++) {
      idx1arr[idx] = idx + 1;
      idx1arr_ind[idx] = TRUE;
      idx1arr_len[idx] = sizeof(int);
    }
    idx1arr_count = NITEMS;
    newP->out_nord.s_O_OL_CNT = NITEMS;


    /* Bind array */
    OCIBNDPLA(nctx->curn2, idx1arr_bp,(OCIError *)errhp,":idx1arr",idx1arr,
          SIZ(int), SQLT_INT, idx1arr_len, NITEMS,&idx1arr_count);

    newP->out_nord.terror = OCIStmtExecute(tpcsvc,nctx->curn2,(OCIError *)errhp,1,0,
              NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
     if(newP->out_nord.terror != OCI_SUCCESS) {
       OCITransRollback(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
       newP->out_nord.terror = OCIERROR((OCIError *)errhp,newP->out_nord.terror);
       return -1;
     }
   }
```

```
  oralogprintf(ora_SlotDataP->server_logtrans,"<<tkvcninit\n");
  return (0);

}

tkvcn (ora_cn_data_t *ora_SlotDataP)
{

   int i;
   int rcount;
   int invalid=0;
   FILE *server_logtrans = ora_SlotDataP->server_logtrans;

   OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
   OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
   dvoid *errhp = ora_SlotDataP->errhp;
   OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
   OCISession *tpcusr = ora_SlotDataP->tpcusr;
   OCIStmt *curi = ora_SlotDataP->curi;
   nord_wrapper *newP = &ora_SlotDataP->globals;
   newctx *nctx = (newctx *)ora_SlotDataP->nctx;

retry:

   newP->out_nord.status = 0;                    /* number of invalid items */

   /* get number of order lines, and check if all are local */

   newP->out_nord.s_O_OL_CNT = NITEMS;
   newP->in_nord.s_all_local = 1;
   for (i = 0; i < NITEMS; i++) {
       if (newP->in_nord.s_OL_I_ID[i] == 0) {
           newP->out_nord.s_O_OL_CNT = i;
         break;
       }
       if (newP->in_nord.s_OL_SUPPLY_W_ID[i] != newP->in_nord.s_W_ID) {
#ifdef USE_IEEE_NUMBER
           nctx->s_remote[i] = 1.0;
#else
           nctx->s_remote[i] = 1;
#endif
           newP->in_nord.s_all_local = 0;
       }
       else
          nctx->s_remote[i] = 0;
   }

   nctx->w_id_len = sizeof(newP->in_nord.s_W_ID);
   nctx->d_id_len = sizeof(newP->in_nord.s_D_ID);
   nctx->c_id_len = sizeof(newP->in_nord.s_C_ID);
   nctx->o_all_local_len = sizeof(newP->in_nord.s_all_local);
   nctx->o_ol_cnt_len = sizeof(newP->out_nord.s_O_OL_CNT);
   nctx->w_tax_len = 0;
   nctx->d_tax_len = 0;
   nctx->o_id_len = sizeof(newP->out_nord.s_O_ID);
   nctx->c_discount_len = 0;
   nctx->c_credit_len = 0;
   nctx->c_last_len = 0;
   nctx->retries_len = sizeof(newP->out_nord.retry);
   nctx->cr_date_len = sizeof(newP->in_nord.cr_date);
   /* this is the row count */
   rcount = newP->out_nord.s_O_OL_CNT;
   nctx->nol_i_count = newP->out_nord.s_O_OL_CNT;
   nctx->nol_q_count = newP->out_nord.s_O_OL_CNT;
   nctx->nol_s_count = newP->out_nord.s_O_OL_CNT;
   nctx->s_remote_count = newP->out_nord.s_O_OL_CNT;

   nctx->nol_qty_count  = 0;
   nctx->nol_bg_count = 0;
```

```
   nctx->nol_item_count = 0;
   nctx->nol_name_count = 0;
   nctx->nol_am_count   = 0;

   /* initialization for array operations */
   for (i = 0; i < newP->out_nord.s_O_OL_CNT; i++) {
      nctx->ol_number[i] = i + 1;
      nctx->nol_i_id_len[i] = sizeof(newP->in_nord.s_OL_I_ID[i]);
      nctx->nol_supply_w_id_len[i] = sizeof(newP->in_nord.s_OL_SUPPLY_W_ID[i]);
      nctx->nol_quantity_len[i] = sizeof(newP->in_nord.s_OL_QUANTITY[i]);
      nctx->nol_amount_len[i] = sizeof(newP->out_nord.s_OL_AMOUNT[i]);
      nctx->ol_o_id_len[i] = sizeof(newP->out_nord.s_O_ID);
      nctx->ol_number_len[i] = sizeof(int);
      nctx->ol_dist_info_len[i] = nctx->s_dist_info_len[i];
      nctx->s_remote_len[i] = sizeof(int);
      nctx->s_quant_len[i] = sizeof(int);
      nctx->i_name_len[i]=0;
      nctx->s_bg_len[i] = 0;
      invalid += newP->in_nord.s_OL_I_ID[i] > 100000 ? 1 : 0;
   }
   for (i = newP->out_nord.s_O_OL_CNT; i < NITEMS; i++) {
      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_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->i_name_len[i]=0;
      nctx->s_bg_len[i] = 0;
   }

   newP->out_nord.terror = OCIStmtExecute(tpcsvc,nctx->curn1,(OCIError *)errhp,1,0,0,
                        OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);

   if(newP->out_nord.terror != OCI_SUCCESS) {
     OCITransRollback(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
     newP->out_nord.terror = ocierror(server_logtrans,__FILE__,__LINE__,errhp,newP-
>out_nord.terror);
     oralogprintf(ora_SlotDataP->server_logtrans,"newP->out_nord.terror %d\n",newP-
>out_nord.terror);
     if(newP->out_nord.terror == NOT_SERIALIZABLE) {
        newP->out_nord.retry++;
        goto retry;
     } else if (newP->out_nord.terror == RECOVERR) {
        newP->out_nord.retry++;
        goto retry;
     } else if (newP->out_nord.terror == SNAPSHOT_TOO_OLD) {
        newP->out_nord.retry++;
        goto retry;
     } else {
        return -1;
     }
   }

   /* did the txn succeed ? */
   if (rcount != newP->out_nord.s_O_OL_CNT)
   {
       newP->out_nord.status = rcount - newP->out_nord.s_O_OL_CNT;
       newP->out_nord.s_O_OL_CNT = rcount;
       newP->out_nord.terror = 100;
   }

   newP->out_nord.s_total_amount = 0;
   for (i = 0; i < newP->out_nord.s_O_OL_CNT; i++)
   {
      newP->out_nord.s_total_amount += newP->out_nord.s_OL_AMOUNT[i];
   }
```

```
   newP->out_nord.s_total_amount *= ((float)(1.0 - newP->out_nord.s_C_DISCOUNT)) *
(float)(1.0 + ((float)(newP->out_nord.s_D_TAX)) + ((float)(newP->out_nord.s_W_TAX)));
   newP->out_nord.s_total_amount = newP->out_nord.s_total_amount/100;

   return (newP->out_nord.terror);

}


void tkvcndone (ora_cn_data_t *ora_SlotDataP)

{
   newctx *nctx = (newctx *)ora_SlotDataP->nctx;


   if (nctx)
   {
     DISCARD OCIHandleFree((dvoid *)nctx->curn1,OCI_HTYPE_STMT);
     DISCARD OCIHandleFree((dvoid *)nctx->curn2,OCI_HTYPE_STMT);
     free (nctx);
   }
}
```

plord.cpp

```
#include "../../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"
#include "tpccflags.h"

extern void oralogprintf(FILE *,char *format, ...);


/*----------------------------------------------------------------------
                PRIVATE TYPES AND CONSTANTS
  ----------------------------------------------------------------------*/

/*----------------------------------------------------------------------
                STATIC FUNCTION DECLARATIONS
  ----------------------------------------------------------------------*/









#define SQLCUR0 "SELECT rowid FROM cust \
              WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last = :c_last \
              ORDER BY c_last, c_d_id, c_w_id, c_first"

#define SQLCUR1 "SELECT /*+ USE_NL(cust) INDEX_DESC(ordr iordr2) */  \
              c_id, c_balance, c_first, c_middle, c_last, \
              o_id, o_entry_d, o_carrier_id, o_ol_cnt \
              FROM cust, ordr \
              WHERE cust.rowid = :cust_rowid \
              AND   o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id = c_id \
              ORDER BY o_c_id DESC, o_d_id DESC, o_w_id DESC, o_id DESC"

#define SQLCUR2 "SELECT /*+ USE_NL(cust)  INDEX_DESC (ordr iordr2) */   \
              c_balance, c_first, c_middle, c_last, \
              o_id, o_entry_d, o_carrier_id, o_ol_cnt \
              FROM cust, ordr \
              WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id \
              AND o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id = c_id \
              ORDER BY o_c_id DESC, o_d_id DESC, o_w_id DESC , o_id DESC"

#define SQLCUR3 "SELECT  /*+ INDEX(ordl) */ \
              ol_i_id, ol_supply_w_id, ol_quantity, ol_amount, ol_delivery_d \
              FROM ordl \
              WHERE ol_o_id = :o_id AND  ol_d_id = :d_id AND ol_w_id = :w_id"

#define SQLCUR4 "SELECT count(c_last) FROM cust \
               WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last = :c_last "


struct ordctx {

   ub2 c_rowid_len[100];
   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];
   ub2 ol_w_id_len;
   ub2 ol_d_id_len;
   ub2 ol_o_id_len;


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

   OCIStmt *curo0;
   OCIStmt *curo1;
   OCIStmt *curo2;
   OCIStmt *curo3;
   OCIStmt *curo4;
   OCIBind *c_id_bp;
   OCIBind *w_id_bp[4];
   OCIBind *d_id_bp[4];
   OCIBind *c_last_bp[2];
   OCIBind *o_id_bp;
   OCIBind *c_rowid_bp;
   OCIDefine *c_rowid_dp;
   OCIDefine *c_last_dp[2];
   OCIDefine *c_id_dp;
   OCIDefine *c_first_dp[2];
   OCIDefine *c_middle_dp[2];
   OCIDefine *c_balance_dp[2];
   OCIDefine *o_id_dp[2];
   OCIDefine *o_entry_d_dp[2];
   OCIDefine *o_cr_id_dp[2];
   OCIDefine *o_ol_cnt_dp[2];
   OCIDefine *ol_d_d_dp;
   OCIDefine *ol_i_id_dp;
   OCIDefine *ol_supply_w_id_dp;
   OCIDefine *ol_quantity_dp;
   OCIDefine *ol_amount_dp;
   OCIDefine *ol_d_base_dp;
   OCIDefine *c_count_dp;
   OCIRowid *c_rowid_ptr[100];
   OCIRowid *c_rowid_cust;
   int cs;
   int cust_idx;
   int norow;
   int rcount;
   int somerows;
};

typedef struct ordctx ordctx;

struct defctx
{
 boolean reexec;
 ub4 count;
};
typedef struct defctx defctx;

static ordcount = 0;

tkvcoinit (ora_cn_data_t *ora_SlotDataP)

{
   FILE *server_logtrans = ora_SlotDataP->server_logtrans;
   ordctx *octx;
   defctx *cbctx;

   int i;
   text stmbuf[SQL_BUF_SIZE];

   ords_wrapper *ordP;
   OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
   OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
   dvoid *errhp = ora_SlotDataP->errhp;
   OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
   OCISession *tpcusr = ora_SlotDataP->tpcusr;
   OCIStmt *curi = ora_SlotDataP->curi;

   octx = (ordctx *) malloc (sizeof(ordctx));
   DISCARD memset(octx,(char)0,sizeof(ordctx));
   octx->cs = 1;
   octx->norow = 0;
   octx->somerows = 10;
   ora_SlotDataP->octx = (void *)octx;

   cbctx = (defctx *) malloc (sizeof(defctx));
   DISCARD memset(cbctx,(char)0,sizeof(defctx));
   ora_SlotDataP->cbctx = (void *)cbctx;

   memset(&ora_SlotDataP->ordP,(char)0,sizeof(ords_wrapper));
   ordP = &ora_SlotDataP->ordP;

   for(i=0;i<100;i++) {
    DISCARD OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,
               (dvoid**)&octx->c_rowid_ptr[i], OCI_DTYPE_ROWID,0,(dvoid**)0));
  }


   DISCARD OCIERROR(errhp,
     OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
   DISCARD OCIERROR(errhp,
     OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo1,OCI_HTYPE_STMT,0,(dvoid**)0));
   DISCARD OCIERROR(errhp,
     OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo2,OCI_HTYPE_STMT,0,(dvoid**)0));
   DISCARD OCIERROR(errhp,
     OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo3,OCI_HTYPE_STMT,0,(dvoid**)0));
   DISCARD OCIERROR(errhp,
     OCIHandleAlloc(tpcenv,(dvoid**)&octx->curo4,OCI_HTYPE_STMT,0,(dvoid**)0));

/*  c_id = 0, use find customer by lastname. Get an array or rowid's back*/
   DISCARD sprintf((char *) stmbuf, SQLCUR0);
   DISCARD OCIERROR(errhp,
     OCIStmtPrepare(octx->curo0,(OCIError *)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
               OCI_NTV_SYNTAX,OCI_DEFAULT));
   DISCARD OCIERROR(errhp,
     OCIAttrSet(octx->curo0,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));
/* get order/customer info back based on rowid */
   DISCARD sprintf((char *) stmbuf, SQLCUR1);
   DISCARD OCIERROR(errhp,
     OCIStmtPrepare(octx->curo1,(OCIError *)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
               OCI_NTV_SYNTAX,OCI_DEFAULT));
```

```
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo1,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));

/*  c_id == 0, use lastname to find customer */
  DISCARD sprintf((char *) stmbuf, SQLCUR2);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo2,(OCIError *)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo2,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));

  DISCARD sprintf((char *) stmbuf, SQLCUR3);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo3,(OCIError *)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo3,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));

  DISCARD sprintf((char *) stmbuf, SQLCUR4);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo4,(OCIError *)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo4,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));

  for (i = 0; i < NITEMS; i++) {

    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(ordP->out_ords.s_OL_DELIVERY_D_base[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_o_id_csize = NITEMS;
  octx->ol_d_id_csize = NITEMS;
  octx->ol_w_id_len = sizeof(int);
  octx->ol_d_id_len = sizeof(int);
  octx->ol_o_id_len = sizeof(int);

  /* bind variables */

  /* c_id (customer id) is not known */
  OCIBND(octx->curo0,octx->w_id_bp[0],(OCIError *)errhp,":w_id",ADR(ordP-
>in_ords.s_W_ID),
        SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->d_id_bp[0],(OCIError *)errhp,":d_id",ADR(ordP-
>in_ords.s_D_ID),
        SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->c_last_bp[0],(OCIError *)errhp,":c_last",ordP-
>in_ords.s_C_LAST,
      SIZ(ordP->in_ords.s_C_LAST), SQLT_STR);
  OCIDFNRA(octx->curo0,octx->c_rowid_dp,(OCIError *)errhp,1,octx->c_rowid_ptr,
         SIZ(OCIRowid*), SQLT_RDD, NULL, octx->c_rowid_len, NULL);


  OCIBND(octx->curo1,octx->c_rowid_bp,(OCIError *)errhp,":cust_rowid", &octx-
>c_rowid_cust,
       sizeof( octx->c_rowid_ptr[0]),SQLT_RDD);
  OCIDEF(octx->curo1,octx->c_id_dp,(OCIError *)errhp,1,ADR(ordP-
>in_ords.s_C_ID),SIZ(int),SQLT_INT);
```

```
#ifdef USE_IEEE_NUMBER
  OCIDEF(octx->curo1,octx->c_balance_dp[0],(OCIError *)errhp,2,ADR(ordP-
>out_ords.s_C_BALANCE),
        SIZ(double),SQLT_BDOUBLE);
#else
  OCIDEF(octx->curo1,octx->c_balance_dp[0],(OCIError *)errhp,2,ADR(ordP-
>out_ords.s_C_BALANCE),
        SIZ(double),SQLT_FLT);
#endif
  OCIDEF(octx->curo1,octx->c_first_dp[0],(OCIError *)errhp,3,ordP-
>out_ords.s_C_FIRST,SIZ(ordP->out_ords.s_C_FIRST)-1,
        SQLT_CHR);
  OCIDEF(octx->curo1,octx->c_middle_dp[0],(OCIError *)errhp,4,ordP-
>out_ords.s_C_MIDDLE,
        SIZ(ordP->out_ords.s_C_MIDDLE)-1,SQLT_AFC);
  OCIDEF(octx->curo1,octx->c_last_dp[0],(OCIError *)errhp,5,ordP-
>out_ords.s_C_LAST,SIZ(ordP->out_ords.s_C_LAST)-1,
        SQLT_CHR);
  OCIDEF(octx->curo1,octx->o_id_dp[0],(OCIError *)errhp,6,ADR(ordP-
>out_ords.s_O_ID),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_entry_d_dp[0],(OCIError *)errhp,7,
        &ordP->out_ords.s_OL_DELIVERY_D_base,SIZ(OCIDate),SQLT_ODT);
  OCIDEF(octx->curo1,octx->o_cr_id_dp[0],(OCIError *)errhp,8,ADR(ordP-
>out_ords.s_O_CARRIER_ID),
        SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_ol_cnt_dp[0],(OCIError *)errhp,9,ADR(ordP-
>out_ords.s_ol_cnt),
        SIZ(int),SQLT_INT);


/* Bind for third cursor , no-zero customer id */
  OCIBND(octx->curo2,octx->w_id_bp[1],(OCIError *)errhp,":w_id",ADR(ordP-
>in_ords.s_W_ID),
       SIZ(int),SQLT_INT);
  OCIBND(octx->curo2,octx->d_id_bp[1],(OCIError *)errhp,":d_id",ADR(ordP-
>in_ords.s_D_ID),
        SIZ(int),SQLT_INT);
  OCIBND(octx->curo2,octx->c_id_bp,(OCIError *)errhp,":c_id",ADR(ordP-
>in_ords.s_C_ID),
        SIZ(int),SQLT_INT);
#ifdef USE_IEEE_NUMBER
  OCIDEF(octx->curo2,octx->c_balance_dp[1],(OCIError *)errhp,1,ADR(ordP-
>out_ords.s_C_BALANCE),
        SIZ(double),SQLT_BDOUBLE);
#else
  OCIDEF(octx->curo2,octx->c_balance_dp[1],(OCIError *)errhp,1,ADR(ordP-
>out_ords.s_C_BALANCE),
        SIZ(double),SQLT_FLT);
#endif
  OCIDEF(octx->curo2,octx->c_first_dp[1],(OCIError *)errhp,2,ordP-
>out_ords.s_C_FIRST,SIZ(ordP->out_ords.s_C_FIRST)-1,
        SQLT_CHR);
  OCIDEF(octx->curo2,octx->c_middle_dp[1],(OCIError *)errhp,3,ordP-
>out_ords.s_C_MIDDLE,
        SIZ(ordP->out_ords.s_C_MIDDLE)-1,SQLT_AFC);
  OCIDEF(octx->curo2,octx->c_last_dp[1],(OCIError *)errhp,4,ordP-
>out_ords.s_C_LAST,SIZ(ordP->out_ords.s_C_LAST)-1,
       SQLT_CHR);
  OCIDEF(octx->curo2,octx->o_id_dp[1],(OCIError *)errhp,5,ADR(ordP-
>out_ords.s_O_ID),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo2,octx->o_entry_d_dp[1],(OCIError *)errhp,6, &ordP-
>out_ords.s_OL_DELIVERY_D_base,
        SIZ(OCIDate),SQLT_ODT);
  OCIDEF(octx->curo2, octx->o_cr_id_dp[1],(OCIError *)errhp,7,ADR(ordP-
>out_ords.s_O_CARRIER_ID),
        SIZ(int), SQLT_INT);
  OCIDEF(octx->curo2,octx->o_ol_cnt_dp[1],(OCIError *)errhp,8,ADR(ordP-
>out_ords.s_ol_cnt),
        SIZ(int),SQLT_INT);
```

```
/* Bind for last cursor */


  OCIBND(octx->curo3,octx->w_id_bp[2],(OCIError *)errhp,":w_id",ADR(ordP-
>in_ords.s_W_ID), SIZ(int),SQLT_INT);
  OCIBND(octx->curo3,octx->d_id_bp[2],(OCIError *)errhp,":d_id",ADR(ordP-
>in_ords.s_D_ID), SIZ(int),SQLT_INT);
  OCIBND(octx->curo3,octx->o_id_bp,(OCIError *)errhp,":o_id",ADR(ordP-
>out_ords.s_O_ID), SIZ(int),SQLT_INT);
/*
  OCIBND(octx->curo3,octx->c_id_bp,(OCIError *)errhp,":c_id",ADR(ordP->ordin.c_id),
SIZ(int),SQLT_INT);
*/


  OCIDFNRA(octx->curo3, octx->ol_i_id_dp, (OCIError *)errhp, 1, ordP-
>out_ords.s_OL_I_ID,SIZ(int),SQLT_INT,
        NULL,octx->ol_i_id_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_supply_w_id_dp,(OCIError *)errhp,2, ordP-
>out_ords.s_OL_SUPPLY_W_ID,
        SIZ(int),SQLT_INT, NULL,
        octx->ol_supply_w_id_len, NULL);
#ifdef USE_IEEE_NUMBER
  OCIDFNRA(octx->curo3, octx->ol_quantity_dp,(OCIError *)errhp,3, ordP-
>out_ords.s_OL_QUANTITY,SIZ(float),
        SQLT_FLT, NULL,octx->ol_quantity_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_amount_dp,(OCIError *)errhp,4,ordP-
>out_ords.s_OL_AMOUNT, SIZ(float),
        SQLT_FLT,NULL, octx->ol_amount_len, NULL);
#else
  OCIDFNRA(octx->curo3, octx->ol_quantity_dp,(OCIError *)errhp,3, ordP-
>out_ords.s_OL_QUANTITY,SIZ(int),
        SQLT_INT, NULL,octx->ol_quantity_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_amount_dp,(OCIError *)errhp,4,ordP-
>out_ords.s_OL_AMOUNT, SIZ(int),
        SQLT_INT,NULL, octx->ol_amount_len, NULL);
#endif
  OCIDFNRA(octx->curo3,octx->ol_d_base_dp,(OCIError *)errhp,5,ordP-
>out_ords.s_OL_DELIVERY_D_base,SIZ(OCIDate),
        SQLT_ODT, NULL,octx->ol_delivery_d_len,NULL);

  OCIBND(octx->curo4,octx->w_id_bp[3],(OCIError *)errhp,":w_id",ADR(ordP-
>in_ords.s_W_ID),
       SIZ(int),SQLT_INT);
  OCIBND(octx->curo4,octx->d_id_bp[3],(OCIError *)errhp,":d_id",ADR(ordP-
>in_ords.s_D_ID),
       SIZ(int),SQLT_INT);
  OCIBND(octx->curo4,octx->c_last_bp[1],(OCIError *)errhp,":c_last",ordP-
>out_ords.s_C_LAST,
       SIZ(ordP->out_ords.s_C_LAST), SQLT_STR);
  OCIDEF(octx->curo4,octx->c_count_dp,(OCIError *)errhp,1,ADR(octx->rcount),SIZ(int),
       SQLT_INT);

  return (0);

}

tkvco (ora_cn_data_t *ora_SlotDataP)

{
 FILE *server_logtrans = ora_SlotDataP->server_logtrans;

  int i;
  int rcount;

  ordctx *octx = (ordctx *)ora_SlotDataP->octx;
  defctx *cbctx = (defctx *)ora_SlotDataP->cbctx;
  ords_wrapper *ordP = &ora_SlotDataP->ordP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
```

```
   dvoid *errhp = ora_SlotDataP->errhp;
   OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
   OCISession *tpcusr = ora_SlotDataP->tpcusr;
   OCIStmt *curi = ora_SlotDataP->curi;


#if defined(ISO9)
   int secondread = 0;
   char sdate[30];
   ub4  datelen;
   sysdate(sdate);
   printf("Order Status started at: %s\n", sdate);
#endif

#ifdef DEBUG
   if (bylastname) tkvc_trace_on();
#endif

#ifdef BLANK_PAD_C_LAST
   for (i = strlen(c_last); i < sizeof(c_last)-1; i++)
   {
      c_last[i] = ' ';
   }
   c_last[i] = '\0';
#endif

   for (i = 0; i < NITEMS; i++) {
      octx->ol_supply_w_id_len[i] = sizeof(int);
      octx->ol_i_id_len[i] = sizeof(int);
      octx->ol_quantity_len[i] = sizeof(float);
      octx->ol_amount_len[i] = sizeof(float);
      octx->ol_quantity_len[i] = sizeof(int);
      octx->ol_amount_len[i] = sizeof(int);
      octx->ol_delivery_d_len[i] = sizeof(OCIDate);
   }
   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;
retry:
   if(ordP->in_ords.bylastname)
   {
     ordcount++;
     cbctx->reexec = FALSE;
#undef STRIP_BLANKS_C_LAST
#ifdef STRIP_BLANKS_C_LAST
     for (i = strlen(ordP->in_ords.s_C_LAST)-1; i >= 0 && (ordP->in_ords.s_C_LAST[i] == ' ');
i--)
     {
        ordP->in_ords.s_C_LAST[i] = '\0';
     }
#endif
     ordP->out_ords.terror=OCIStmtExecute(tpcsvc,octx->curo0,(OCIError *)errhp,100,0,
             NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
     /* will get OCI_NO_DATA if <100 found */
     if ((ordP->out_ords.terror != OCI_NO_DATA) && (ordP->out_ords.terror !=
OCI_SUCCESS))
     {
        ordP->out_ords.terror=OCIERROR(errhp, ordP->out_ords.terror);
        if((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP->out_ords.terror ==
RECOVERR))
       {
         DISCARD OCITransCommit(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
         ordP->out_ords.retry++;
         goto retry;
       } else {
         return -1;
       }
     }
     if (ordP->out_ords.terror == OCI_NO_DATA) /* there are no more rows */
     {
      /* get rowcount, find middle one */
       DISCARD OCIAttrGet(octx->curo0,OCI_HTYPE_STMT,&rcount,NULL,
                    OCI_ATTR_ROW_COUNT,(OCIError *)errhp);

       if (rcount <1)
       {
          oralogprintf(server_logtrans,"ORDERSTATUS:  rcount=%d\n",rcount);
          oralogprintf(server_logtrans,"ORDERSTATUS: c_d_id %d c_w_id %d c_last
%s\n",ordP->in_ords.s_D_ID,ordP->in_ords.s_W_ID,ordP->in_ords.s_C_LAST);
         return (-1);
       }
       octx->cust_idx=(rcount)/2 ;
     }
     else
     {
       /* count the number of rows */
         ordP->out_ords.terror=OCIStmtExecute(tpcsvc,octx->curo4,(OCIError *)errhp,1,0,
                NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
        if ((ordP->out_ords.terror != OCI_NO_DATA) && (ordP->out_ords.terror !=
OCI_SUCCESS))
       {
          ordP->out_ords.terror=OCIERROR(errhp, ordP->out_ords.terror);
          if ((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP->out_ords.terror ==
RECOVERR))
         {
           DISCARD OCITransCommit(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
           ordP->out_ords.retry++;
           goto retry;
         } else {
           return -1;
         }
       }
       cbctx->reexec = TRUE;
       cbctx->count = (octx->rcount+1)/2 ;
       ordP->out_ords.terror=OCIStmtExecute(tpcsvc,octx->curo0,(OCIError *)errhp,cbctx-
>count,
                            0,NULLP(CONST OCISnapshot),
                            NULLP(OCISnapshot),OCI_DEFAULT);

       DISCARD OCIAttrGet(octx->curo0,OCI_HTYPE_STMT,&rcount,NULL,
                          OCI_ATTR_ROW_COUNT,(OCIError *)errhp);

       /* will get OCI_NO_DATA if <100 found */
       if (cbctx->count != rcount)
       {
         userlog ("did not get all rows ");
         return (-1);
       }

       if ((ordP->out_ords.terror != OCI_NO_DATA) && (ordP->out_ords.terror !=
OCI_SUCCESS))
       {
         ordP->out_ords.terror=OCIERROR(errhp, ordP->out_ords.terror);
         if((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP->out_ords.terror ==
RECOVERR))
         {
           DISCARD OCITransCommit(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
           ordP->out_ords.retry++;
           goto retry;
         } else {
           return -1;
         }
       }
       octx->cust_idx=cbctx->count - 1 ;
     }

     octx->c_rowid_cust = octx->c_rowid_ptr[octx->cust_idx];
     ordP->out_ords.terror=OCIStmtExecute(tpcsvc,octx->curo1,(OCIError *)errhp,1,0,
                NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if (ordP->out_ords.terror != OCI_SUCCESS)
    {
       ordP->out_ords.terror=OCIERROR(errhp,ordP->out_ords.terror);
       DISCARD OCITransCommit(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
       if((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP->out_ords.terror ==
RECOVERR)
          || (ordP->out_ords.terror == SNAPSHOT_TOO_OLD))
       {
         ordP->out_ords.retry++;
         goto retry;
         } else {
         return -1;
       }
     }
   } /* lastname */
   else
   {
       ordP->out_ords.terror=OCIStmtExecute(tpcsvc,octx->curo2,(OCIError *)errhp,1,0,
                    NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
                    OCI_DEFAULT);
       if (ordP->out_ords.terror != OCI_SUCCESS)
     {
        ordP->out_ords.terror=OCIERROR(errhp,ordP->out_ords.terror);
       DISCARD OCITransCommit(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
       if((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP->out_ords.terror ==
RECOVERR)
          || (ordP->out_ords.terror == SNAPSHOT_TOO_OLD))
       {
         ordP->out_ords.retry++;
         goto retry;
       }
       else
       {
         return -1;
       }
     }
   }
   octx->ol_w_id_len = sizeof(int);
   octx->ol_d_id_len = sizeof(int);
   octx->ol_o_id_len = sizeof(int);

    ordP->out_ords.terror = OCIStmtExecute(tpcsvc,octx->curo3,(OCIError *)errhp,ordP-
>out_ords.s_ol_cnt,0,
                     NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
                     OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
   if (ordP->out_ords.terror != OCI_SUCCESS )
   {
      ordP->out_ords.terror=OCIERROR(errhp,ordP->out_ords.terror);
     DISCARD OCITransCommit(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
     if((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP->out_ords.terror ==
RECOVERR)
        || (ordP->out_ords.terror == SNAPSHOT_TOO_OLD))
     {
       ordP->out_ords.retry++;
       goto retry;
     }
     else
     {
       return -1;
     }

   }
   /* clean up and convert the delivery dates */
   for (i = 0; i < ordP->out_ords.s_ol_cnt; i++)
   {
      ordP->out_ords.ol_del_len[i]=sizeof(ordP->out_ords.s_OL_DELIVERY_D_time[i]);
     DISCARD OCIERROR(errhp,OCIDateToText((OCIError *)errhp,&ordP-
>out_ords.s_OL_DELIVERY_D_base[i],
        (const text*)SHORTDATE,(ub1)strlen(SHORTDATE),(text*)0,0,
```

```
        (ub4 *)&ordP->out_ords.ol_del_len[i], (text *)ordP-
>out_ords.s_OL_DELIVERY_D_time[i]));
    }
   ordP->out_ords.terror = 0;
   return (0);

}


void tkvcodone (ora_cn_data_t *ora_SlotDataP)
{

   if (ora_SlotDataP->octx) {
      free (ora_SlotDataP->octx);
      ora_SlotDataP->octx = NULL;
   }
}

/* end of file tkvcord.c */
```

plpay.cpp

```
#include "../../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"

extern char OracleHome[256];
extern int sqlfile(FILE *server_logtrans,char *fnam, text *linebuf);

extern void oralogprintf(FILE *,char *format, ...);

#define SQLTXT_INIT "BEGIN inittpcc.init_pay; END;"

struct payctx {
  OCIStmt *curpi;
  OCIStmt *curp0;
  OCIStmt *curp1;
  OCIBind *w_id_bp[2];
  ub2 w_id_len;

  OCIBind *d_id_bp[2];
  ub2 d_id_len;

  OCIBind *c_w_id_bp[2];
  ub2 c_w_id_len;

  OCIBind *c_d_id_bp[2];
  ub2 c_d_id_len;

  OCIBind *c_id_bp[2];
  ub2 c_id_len;

  OCIBind *h_amount_bp[2];
  ub2 h_amount_len;

  OCIBind *c_last_bp[2];
  ub2 c_last_len;

  OCIBind *w_street_1_bp[2];
  ub2 w_street_1_len;

  OCIBind *w_street_2_bp[2];
  ub2 w_street_2_len;

  OCIBind *w_city_bp[2];
  ub2 w_city_len;

  OCIBind *w_state_bp[2];
  ub2 w_state_len;

  OCIBind *w_zip_bp[2];
  ub2 w_zip_len;

  OCIBind *d_street_1_bp[2];
  ub2 d_street_1_len;

  OCIBind *d_street_2_bp[2];
  ub2 d_street_2_len;

  OCIBind *d_city_bp[2];
  ub2 d_city_len;

  OCIBind *d_state_bp[2];
  ub2 d_state_len;

  OCIBind *d_zip_bp[2];
  ub2 d_zip_len;

  OCIBind *c_first_bp[2];
  ub2 c_first_len;

  OCIBind *c_middle_bp[2];
  ub2 c_middle_len;

  OCIBind *c_street_1_bp[2];
  ub2 c_street_1_len;

  OCIBind *c_street_2_bp[2];
  ub2 c_street_2_len;

  OCIBind *c_city_bp[2];
  ub2 c_city_len;

  OCIBind *c_state_bp[2];
  ub2 c_state_len;

  OCIBind *c_zip_bp[2];
  ub2 c_zip_len;

  OCIBind *c_phone_bp[2];
  ub2 c_phone_len;

  OCIBind *c_since_bp[2];
  ub2 c_since_len;

  OCIBind *c_credit_bp[2];
  ub2 c_credit_len;

  OCIBind *c_credit_lim_bp[2];
  ub2 c_credit_lim_len;

  OCIBind *c_discount_bp[2];
  ub2 c_discount_len;

  OCIBind *c_balance_bp[2];
  ub2 c_balance_len;

  OCIBind *c_data_bp[2];
  ub2 c_data_len;

  OCIBind *h_date_bp[2];
  ub2 h_date_len;

  OCIBind *retries_bp[2];
  ub2 retries_len;

  OCIBind *cr_date_bp[2];
  ub2 cr_date_len;

  OCIBind *byln_bp[2];
  ub2 byln_len;
};

typedef struct payctx payctx;

int tkvcpinit (ora_cn_data_t *ora_SlotDataP)
{
   char sqlfilename[256];
   FILE *server_logtrans = ora_SlotDataP->server_logtrans;
   /*  */
   payctx *pctx;
   paym_wrapper *payP;
   OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
   OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
   dvoid *errhp = ora_SlotDataP->errhp;
   OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
   OCISession *tpcusr = ora_SlotDataP->tpcusr;
   OCIStmt *curi = ora_SlotDataP->curi;

   oralogprintf(ora_SlotDataP->server_logtrans,">>tkvcpinit\n");
   text stmbuf[SQL_BUF_SIZE];

   pctx = (payctx *)malloc(sizeof(payctx));
   memset(pctx,(char)0,sizeof(payctx));
   ora_SlotDataP->pctx = (void *)pctx;

   payP = &ora_SlotDataP->payP;
   memset(&ora_SlotDataP->payP,(char)0,sizeof(paym_wrapper));

/* cursor for init */
   DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curpi)),
          OCI_HTYPE_STMT,0,(dvoid**)0));

   DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curp0)),
          OCI_HTYPE_STMT,0,(dvoid**)0));
   DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)(&(pctx->curp1)),
          OCI_HTYPE_STMT,0,(dvoid**)0));

   /*  build the init statement  and execute it */

   sprintf ((char*)stmbuf, SQLTXT_INIT);
   DISCARD OCIERROR(errhp,OCIStmtPrepare(pctx->curpi, (OCIError *)errhp, stmbuf,
          strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
   DISCARD OCIERROR(errhp, OCIStmtExecute(tpcsvc,pctx->curpi,(OCIError *)errhp,1,0,
                NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT));

   /* customer id !=  0, go by cust id */
   sprintf(sqlfilename,"%s/tpcc-kit/benchrun/blocks/paynz.sql",OracleHome);
   sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuf);
   DISCARD OCIERROR(errhp,OCIStmtPrepare(pctx->curp0, (OCIError *)errhp, stmbuf,
          strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

   /* customer id ==  0, go by last name */

   sprintf(sqlfilename,"%s/tpcc-kit/benchrun/blocks/payz.sql",OracleHome);
   sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuf);
   DISCARD OCIERROR(errhp,OCIStmtPrepare(pctx->curp1, (OCIError *)errhp, stmbuf,
          strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

  pctx->w_id_len = SIZ(payP->in_paym.s_W_ID);
  pctx->d_id_len = SIZ(payP->in_paym.s_D_ID);
  pctx->c_w_id_len = SIZ(payP->in_paym.s_C_W_ID);
  pctx->c_d_id_len = SIZ(payP->in_paym.s_C_D_ID);
  pctx->c_id_len = 0;
  pctx->h_amount_len = SIZ(float);
  pctx->c_last_len = 0;
  pctx->w_street_1_len = 0;
```

```
  pctx->w_street_2_len = 0;
  pctx->w_city_len = 0;
  pctx->w_state_len = 0;
  pctx->w_zip_len = 0;
  pctx->d_street_1_len = 0;
  pctx->d_street_2_len = 0;
  pctx->d_city_len = 0;
  pctx->d_state_len = 0;
  pctx->d_zip_len = 0;
  pctx->c_first_len = 0;
  pctx->c_middle_len = 0;
  pctx->c_street_1_len = 0;
  pctx->c_street_2_len = 0;
  pctx->c_city_len = 0;
  pctx->c_state_len = 0;
  pctx->c_zip_len = 0;
  pctx->c_phone_len = 0;
  pctx->c_since_len = 7;
  pctx->c_credit_len = 0;
  pctx->c_credit_lim_len = 0;
  pctx->c_discount_len = 0;
  pctx->c_balance_len = sizeof(double);
  pctx->c_data_len = 0;
  pctx->h_date_len = 0;
  pctx->retries_len =SIZ(payP->out_paym.retry) ;
  pctx->cr_date_len = 7;



   /* bind variables */


   OCIBNDPL(pctx->curp0, pctx->w_id_bp[0], (OCIError *)errhp,":w_id",ADR(payP-
>in_paym.s_W_ID),SIZ(int),
          SQLT_INT, NULL);

   OCIBNDPL(pctx->curp0, pctx->d_id_bp[0], (OCIError *)errhp,":d_id",ADR(payP-
>in_paym.s_D_ID),SIZ(int),
          SQLT_INT, NULL);
   OCIBND(pctx->curp0, pctx->c_w_id_bp[0], (OCIError *)errhp,":c_w_id",ADR(payP-
>in_paym.s_C_W_ID),SIZ(int),
          SQLT_INT);
   OCIBND(pctx->curp0, pctx->c_d_id_bp[0], (OCIError *)errhp,":c_d_id",ADR(payP-
>in_paym.s_C_D_ID),SIZ(int),
          SQLT_INT);
   OCIBND(pctx->curp0, pctx->c_id_bp[0], (OCIError *)errhp,":c_id",ADR(payP-
>in_paym.s_C_ID),SIZ(int),
          SQLT_INT);

   OCIBNDPL(pctx->curp0, pctx->h_amount_bp[0], (OCIError
*)errhp,":h_amount",ADR(payP->in_paym.s_H_AMOUNT),
          SIZ(payP->in_paym.s_H_AMOUNT),SQLT_FLT,  &pctx->h_amount_len);

   OCIBNDPL(pctx->curp0, pctx->c_last_bp[0], (OCIError *)errhp,":c_last",payP-
>out_paym.s_C_LAST,SIZ(payP->out_paym.s_C_LAST),
          SQLT_STR, &pctx->c_last_len);

   OCIBNDPL(pctx->curp0, pctx->w_street_1_bp[0], (OCIError *)errhp,":w_street_1",payP-
>out_paym.s_W_STREET_1,
          SIZ(payP->out_paym.s_W_STREET_1),SQLT_STR, &pctx->w_street_1_len);
   OCIBNDPL(pctx->curp0, pctx->w_street_2_bp[0], (OCIError *)errhp,":w_street_2",payP-
>out_paym.s_W_STREET_2,
          SIZ(payP->out_paym.s_W_STREET_2),SQLT_STR, &pctx->w_street_2_len);
   OCIBNDPL(pctx->curp0, pctx->w_city_bp[0], (OCIError *)errhp,":w_city",payP-
>out_paym.s_W_CITY,SIZ(payP->out_paym.s_W_CITY),
          SQLT_STR, &pctx->w_city_len);
   OCIBNDPL(pctx->curp0, pctx->w_state_bp[0], (OCIError *)errhp,":w_state",payP-
>out_paym.s_W_STATE,
          SIZ(payP->out_paym.s_W_STATE), SQLT_STR, &pctx->w_state_len);

   OCIBNDPL(pctx->curp0, pctx->w_zip_bp[0], (OCIError *)errhp,":w_zip",payP-
>out_paym.s_W_ZIP,SIZ(payP->out_paym.s_W_ZIP),
          SQLT_STR, &pctx->w_zip_len);

   OCIBNDPL(pctx->curp0, pctx->d_street_1_bp[0], (OCIError *)errhp,":d_street_1",payP-
>out_paym.s_D_STREET_1,
          SIZ(payP->out_paym.s_D_STREET_1),SQLT_STR, &pctx->d_street_1_len);
   OCIBNDPL(pctx->curp0, pctx->d_street_2_bp[0], (OCIError *)errhp,":d_street_2",payP-
>out_paym.s_D_STREET_2,
          SIZ(payP->out_paym.s_D_STREET_2),SQLT_STR, &pctx->d_street_2_len);
   OCIBNDPL(pctx->curp0, pctx->d_city_bp[0], (OCIError *)errhp,":d_city",payP-
>out_paym.s_D_CITY,SIZ(payP->out_paym.s_D_CITY),
          SQLT_STR, &pctx->d_city_len);
   OCIBNDPL(pctx->curp0, pctx->d_state_bp[0], (OCIError *)errhp,":d_state",payP-
>out_paym.s_D_STATE,
          SIZ(payP->out_paym.s_D_STATE), SQLT_STR, &pctx->d_state_len);
   OCIBNDPL(pctx->curp0, pctx->d_zip_bp[0], (OCIError *)errhp,":d_zip",payP-
>out_paym.s_D_ZIP,SIZ(payP->out_paym.s_D_ZIP),
          SQLT_STR, &pctx->d_zip_len);
   OCIBNDPL(pctx->curp0, pctx->c_first_bp[0], (OCIError *)errhp,":c_first",payP-
>out_paym.s_C_FIRST,
          SIZ(payP->out_paym.s_C_FIRST), SQLT_STR, &pctx->c_first_len);
   OCIBNDPL(pctx->curp0, pctx->c_middle_bp[0], (OCIError *)errhp,":c_middle",payP-
>out_paym.s_C_MIDDLE,2,
          SQLT_AFC, &pctx->c_middle_len);
   OCIBNDPL(pctx->curp0, pctx->c_street_1_bp[0], (OCIError *)errhp,":c_street_1",payP-
>out_paym.s_C_STREET_1,
          SIZ(payP->out_paym.s_C_STREET_1),SQLT_STR, &pctx->c_street_1_len);
   OCIBNDPL(pctx->curp0, pctx->c_street_2_bp[0], (OCIError *)errhp,":c_street_2",payP-
>out_paym.s_C_STREET_2,
          SIZ(payP->out_paym.s_C_STREET_2),SQLT_STR, &pctx->c_street_2_len);
   OCIBNDPL(pctx->curp0, pctx->c_city_bp[0], (OCIError *)errhp,":c_city",payP-
>out_paym.s_C_CITY,SIZ(payP->out_paym.s_C_CITY),
          SQLT_STR, &pctx->c_city_len);
   OCIBNDPL(pctx->curp0, pctx->c_state_bp[0], (OCIError *)errhp,":c_state",payP-
>out_paym.s_C_STATE,
          SIZ(payP->out_paym.s_C_STATE), SQLT_STR, &pctx->c_state_len);
   OCIBNDPL(pctx->curp0, pctx->c_zip_bp[0], (OCIError *)errhp,":c_zip",payP-
>out_paym.s_C_ZIP,SIZ(payP->out_paym.s_C_ZIP),
          SQLT_STR,&pctx->c_zip_len);
   OCIBNDPL(pctx->curp0, pctx->c_phone_bp[0], (OCIError *)errhp,":c_phone",payP-
>out_paym.s_C_PHONE,
          SIZ(payP->out_paym.s_C_PHONE), SQLT_STR, &pctx->c_phone_len);
   OCIBNDPL(pctx->curp0, pctx->c_since_bp[0], (OCIError *)errhp,":c_since",&payP-
>out_paym.c_since,
          SIZ(OCIDate), SQLT_ODT, &pctx->c_since_len);
   OCIBNDPL(pctx->curp0, pctx->c_credit_bp[0], (OCIError *)errhp,":c_credit",payP-
>out_paym.s_C_CREDIT,
          SIZ(payP->out_paym.s_C_CREDIT),SQLT_CHR, &pctx->c_credit_len);
   OCIBNDPL(pctx->curp0, pctx->c_credit_lim_bp[0], (OCIError *)errhp,":c_credit_lim",
          ADR(payP->out_paym.s_C_CREDIT_LIM),SIZ(int), SQLT_INT, &pctx-
>c_credit_lim_len);
   OCIBNDPL(pctx->curp0, pctx->c_discount_bp[0], (OCIError *)errhp,":c_discount",
          ADR(payP->out_paym.s_C_DISCOUNT),SIZ(payP->out_paym.s_C_DISCOUNT),
SQLT_FLT, &pctx->c_discount_len);
   OCIBNDPL(pctx->curp0, pctx->c_balance_bp[0], (OCIError *)errhp,":c_balance",
          ADR(payP->out_paym.s_C_BALANCE), SIZ(double),SQLT_FLT, &pctx-
>c_balance_len);

   OCIBNDPL(pctx->curp0, pctx->c_data_bp[0], (OCIError *)errhp,":c_data",payP-
>out_paym.s_C_DATA,SIZ(payP->out_paym.s_C_DATA),
          SQLT_STR, &pctx->c_data_len);

   OCIBNDPL(pctx->curp0, pctx->retries_bp[0], (OCIError *)errhp,":retry",ADR(payP-
>out_paym.retry),
          SIZ(int), SQLT_INT, &pctx->retries_len);

   OCIBNDPL(pctx->curp0, pctx->cr_date_bp[0], (OCIError *)errhp,":cr_date",ADR(payP-
>in_paym.cr_date),
          SIZ(OCIDate),SQLT_ODT, &pctx->cr_date_len);


          /* ---- Binds for  the second cursor   */


   OCIBNDPL(pctx->curp1, pctx->w_id_bp[1], (OCIError *)errhp,":w_id",ADR(payP-
>in_paym.s_W_ID),SIZ(int),
          SQLT_INT, &pctx->w_id_len);
   OCIBNDPL(pctx->curp1, pctx->d_id_bp[1], (OCIError *)errhp,":d_id",ADR(payP-
>in_paym.s_D_ID),SIZ(int),
          SQLT_INT, &pctx->d_id_len);
   OCIBND(pctx->curp1, pctx->c_w_id_bp[1], (OCIError *)errhp,":c_w_id",ADR(payP-
>in_paym.s_C_W_ID),SIZ(int),
          SQLT_INT);
   OCIBND(pctx->curp1, pctx->c_d_id_bp[1], (OCIError *)errhp,":c_d_id",ADR(payP-
>in_paym.s_C_D_ID),SIZ(int),
          SQLT_INT);
   OCIBNDPL(pctx->curp1, pctx->c_id_bp[1], (OCIError *)errhp,":c_id",ADR(payP-
>out_paym.s_C_ID),SIZ(int),
          SQLT_INT, &pctx->c_id_len);
#ifdef USE_IEEE_NUMBER
   OCIBNDPL(pctx->curp1, pctx->h_amount_bp[1], (OCIError
*)errhp,":h_amount",ADR(payP->in_paym.s_H_AMOUNT),
          SIZ(float),SQLT_FLT, &pctx->h_amount_len);
#else
   OCIBNDPL(pctx->curp1, pctx->h_amount_bp[1], (OCIError
*)errhp,":h_amount",ADR(payP->in_paym.s_H_AMOUNT),
          SIZ(int),SQLT_FLT, &pctx->h_amount_len);
#endif
   OCIBND(pctx->curp1, pctx->c_last_bp[1], (OCIError *)errhp,":c_last",payP-
>in_paym.s_C_LAST,SIZ(payP->in_paym.s_C_LAST),
          SQLT_STR);
   OCIBNDPL(pctx->curp1, pctx->w_street_1_bp[1], (OCIError *)errhp,":w_street_1",payP-
>out_paym.s_W_STREET_1,
          SIZ(payP->out_paym.s_W_STREET_1),SQLT_STR, &pctx->w_street_1_len);
   OCIBNDPL(pctx->curp1, pctx->w_street_2_bp[1], (OCIError *)errhp,":w_street_2",payP-
>out_paym.s_W_STREET_2,
          SIZ(payP->out_paym.s_W_STREET_2),SQLT_STR, &pctx->w_street_2_len);
   OCIBNDPL(pctx->curp1, pctx->w_city_bp[1], (OCIError *)errhp,":w_city",payP-
>out_paym.s_W_CITY,SIZ(payP->out_paym.s_W_CITY),
          SQLT_STR, &pctx->w_city_len);
   OCIBNDPL(pctx->curp1, pctx->w_state_bp[1], (OCIError *)errhp,":w_state",payP-
>out_paym.s_W_STATE,
          SIZ(payP->out_paym.s_W_STATE), SQLT_STR, &pctx->w_state_len);
   OCIBNDPL(pctx->curp1, pctx->w_zip_bp[1], (OCIError *)errhp,":w_zip",payP-
>out_paym.s_W_ZIP,SIZ(payP->out_paym.s_W_ZIP),
          SQLT_STR, &pctx->w_zip_len);
   OCIBNDPL(pctx->curp1, pctx->d_street_1_bp[1], (OCIError *)errhp,":d_street_1",payP-
>out_paym.s_D_STREET_1,
          SIZ(payP->out_paym.s_D_STREET_1),SQLT_STR, &pctx->d_street_1_len);
   OCIBNDPL(pctx->curp1, pctx->d_street_2_bp[1], (OCIError *)errhp,":d_street_2",payP-
>out_paym.s_D_STREET_2,
          SIZ(payP->out_paym.s_D_STREET_2),SQLT_STR, &pctx->d_street_2_len);
   OCIBNDPL(pctx->curp1, pctx->d_city_bp[1], (OCIError *)errhp,":d_city",payP-
>out_paym.s_D_CITY,SIZ(payP->out_paym.s_D_CITY),
          SQLT_STR, &pctx->d_city_len);
   OCIBNDPL(pctx->curp1, pctx->d_state_bp[1], (OCIError *)errhp,":d_state",payP-
>out_paym.s_D_STATE,
          SIZ(payP->out_paym.s_D_STATE), SQLT_STR, &pctx->d_state_len);
   OCIBNDPL(pctx->curp1, pctx->d_zip_bp[1], (OCIError *)errhp,":d_zip",payP-
>out_paym.s_D_ZIP,SIZ(payP->out_paym.s_D_ZIP),
          SQLT_STR, &pctx->d_zip_len);
   OCIBNDPL(pctx->curp1, pctx->c_first_bp[1], (OCIError *)errhp,":c_first",payP-
>out_paym.s_C_FIRST,
          SIZ(payP->out_paym.s_C_FIRST), SQLT_STR, &pctx->c_first_len);
   OCIBNDPL(pctx->curp1, pctx->c_middle_bp[1], (OCIError *)errhp,":c_middle",payP-
>out_paym.s_C_MIDDLE,2,
          SQLT_AFC, &pctx->c_middle_len);

   OCIBNDPL(pctx->curp1, pctx->c_street_1_bp[1], (OCIError *)errhp,":c_street_1",payP-
>out_paym.s_C_STREET_1,
```

```
      SIZ(payP->out_paym.s_C_STREET_1),SQLT_STR, &pctx->c_street_1_len);
   OCIBNDPL(pctx->curp1, pctx->c_street_2_bp[1], (OCIError *)errhp,":c_street_2",payP-
>out_paym.s_C_STREET_2,
      SIZ(payP->out_paym.s_C_STREET_2),SQLT_STR, &pctx->c_street_2_len);
   OCIBNDPL(pctx->curp1, pctx->c_city_bp[1], (OCIError *)errhp,":c_city",payP-
>out_paym.s_C_CITY,
      SIZ(payP->out_paym.s_C_CITY),SQLT_STR, &pctx->c_city_len);
   OCIBNDPL(pctx->curp1, pctx->c_state_bp[1], (OCIError *)errhp,":c_state",payP-
>out_paym.s_C_STATE,
      SIZ(payP->out_paym.s_C_STATE), SQLT_STR, &pctx->c_state_len);
   OCIBNDPL(pctx->curp1, pctx->c_zip_bp[1], (OCIError *)errhp,":c_zip",payP-
>out_paym.s_C_ZIP,SIZ(payP->out_paym.s_C_ZIP),
         SQLT_STR, &pctx->c_zip_len);
   OCIBNDPL(pctx->curp1, pctx->c_phone_bp[1], (OCIError *)errhp,":c_phone",payP-
>out_paym.s_C_PHONE,
      SIZ(payP->out_paym.s_C_PHONE), SQLT_STR, &pctx->c_phone_len);
   OCIBNDPL(pctx->curp1, pctx->c_since_bp[1], (OCIError *)errhp,":c_since",&payP-
>out_paym.c_since,
         SIZ(OCIDate), SQLT_ODT, &pctx->c_since_len);
   OCIBNDPL(pctx->curp1, pctx->c_credit_bp[1], (OCIError *)errhp,":c_credit",payP-
>out_paym.s_C_CREDIT,
         SIZ(payP->out_paym.s_C_CREDIT),SQLT_CHR, &pctx->c_credit_len);
   OCIBNDPL(pctx->curp1, pctx->c_credit_lim_bp[1], (OCIError *)errhp,":c_credit_lim",
         ADR(payP->out_paym.s_C_CREDIT_LIM),SIZ(int), SQLT_INT, &pctx-
>c_credit_lim_len);
   OCIBNDPL(pctx->curp1, pctx->c_discount_bp[1], (OCIError *)errhp,":c_discount",
         ADR(payP->out_paym.s_C_DISCOUNT),SIZ(double), SQLT_FLT, &pctx-
>c_discount_len);
#ifdef USE_IEEE_NUMBER
   OCIBNDPL(pctx->curp1, pctx->c_balance_bp[1], (OCIError *)errhp,":c_balance",
         ADR(payP->out_paym.s_C_BALANCE), SIZ(double),SQLT_BDOUBLE, &pctx-
>c_balance_len);
#else
   OCIBNDPL(pctx->curp1, pctx->c_balance_bp[1], (OCIError *)errhp,":c_balance",
       ADR(payP->out_paym.s_C_BALANCE), SIZ(double),SQLT_FLT, &pctx-
>c_balance_len);
#endif
   OCIBNDPL(pctx->curp1, pctx->c_data_bp[1], (OCIError *)errhp,":c_data",payP-
>out_paym.s_C_DATA,SIZ(payP->out_paym.s_C_DATA),
         SQLT_STR, &pctx->c_data_len);
   OCIBNDPL(pctx->curp1, pctx->retries_bp[1], (OCIError *)errhp,":retry",ADR(payP-
>out_paym.retry),
         SIZ(int), SQLT_INT, &pctx->retries_len);
   OCIBNDPL(pctx->curp1, pctx->cr_date_bp[1], (OCIError *)errhp,":cr_date",ADR(payP-
>in_paym.cr_date),
         SIZ(OCIDate),SQLT_ODT, &pctx->cr_date_len);
   oralogprintf(ora_SlotDataP->server_logtrans,"<<tkvcpinit\n");
   return (0);

}


tkvcp (ora_cn_data_t *ora_SlotDataP)

{
   FILE *server_logtrans = ora_SlotDataP->server_logtrans;

  /*  */
  payctx *pctx = (payctx *)ora_SlotDataP->pctx;
  paym_wrapper *payP = &ora_SlotDataP->payP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  dvoid *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;

retry:
  pctx->w_id_len = SIZ(payP->in_paym.s_W_ID);
  pctx->d_id_len = SIZ(payP->in_paym.s_D_ID);
  pctx->c_w_id_len = 0;
  pctx->c_d_id_len = 0;
  pctx->c_id_len = 0;
  pctx->h_amount_len = SIZ(payP->in_paym.s_H_AMOUNT);
  pctx->c_last_len = SIZ(payP->in_paym.s_C_LAST);
  pctx->w_street_1_len = 0;
  pctx->w_street_2_len = 0;
  pctx->w_city_len = 0;
  pctx->w_state_len = 0;
  pctx->w_zip_len = 0;
  pctx->d_street_1_len = 0;
  pctx->d_street_2_len = 0;
  pctx->d_city_len = 0;
  pctx->d_state_len = 0;
  pctx->d_zip_len = 0;
  pctx->c_first_len = 0;
  pctx->c_middle_len = 0;
  pctx->c_street_1_len = 0;
  pctx->c_street_2_len = 0;
  pctx->c_city_len = 0;
  pctx->c_state_len = 0;
  pctx->c_zip_len = 0;
  pctx->c_phone_len = 0;
  pctx->c_since_len = 7;
  pctx->c_credit_len = 0;
  pctx->c_credit_lim_len = 0;
  pctx->c_discount_len = 0;
  pctx->c_balance_len = sizeof(double);
  pctx->c_data_len = 0;
  pctx->h_date_len = 0;
  pctx->retries_len = SIZ(payP->out_paym.retry);
  pctx->cr_date_len = 7;

  if(payP->in_paym.bylastname) {
     payP->out_paym.terror=OCIStmtExecute(tpcsvc,pctx->curp1,(OCIError *)errhp,1,0,
           NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
           OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
  } else {
     payP->out_paym.terror=OCIStmtExecute(tpcsvc,pctx->curp0,(OCIError *)errhp,1,0,
           NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
           OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
  }

  if(payP->out_paym.terror != OCI_SUCCESS) {
     oralogprintf(ora_SlotDataP->server_logtrans,"payP->out_paym.terror = %d\n",payP-
>out_paym.terror);
     OCITransRollback(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
     payP->out_paym.terror = OCIERROR(errhp,payP->out_paym.terror);
     oralogprintf(ora_SlotDataP->server_logtrans,"payP->out_paym.terror = %d\n",payP-
>out_paym.terror);
     if(payP->out_paym.terror == NOT_SERIALIZABLE) {
       payP->out_paym.retry++;
       goto retry;
     } else if (payP->out_paym.terror == RECOVERR) {
       payP->out_paym.retry++;
       goto retry;
     } else if (payP->out_paym.terror == SNAPSHOT_TOO_OLD) {
       payP->out_paym.retry++;
       goto retry;
     } else {
       return -1;
     }
   }
   return 0;
}



void tkvcpdone (ora_cn_data_t *ora_SlotDataP)
{
  if(ora_SlotDataP->pctx) {
    free(ora_SlotDataP->pctx);
    ora_SlotDataP->pctx = NULL;
  }
}
```

plsto.cpp

```
#include "../../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"

extern void oralogprintf(FILE *,char *format, ...);

#ifdef PLSQLSTO
#define SQLTXT "BEGIN stocklevel.getstocklevel (:w_id, :d_id, :threshold, \
  :low_stock); END;"
#else
#define SQLTXT "SELECT /*+ nocache (stok) */ count (DISTINCT s_i_id) \
         FROM ordl, stok, dist \
         WHERE d_id = :d_id AND d_w_id = :w_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) \
        order by ol_o_id desc"
#endif

struct stoctx {
  OCIStmt *curs;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *threshold_bp;
#ifdef PLSQLSTO
  OCIBind *low_stock_bp;
#else
  OCIDefine *low_stock_bp;
#endif
  int norow;
};

typedef struct stoctx stoctx;

tkvcsinit (ora_cn_data_t *ora_SlotDataP)

{
  FILE *server_logtrans = ora_SlotDataP->server_logtrans;

  /*  */
  stoctx *sctx;
  stok_wrapper *stoP;
  OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
  OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
  dvoid *errhp = ora_SlotDataP->errhp;
  OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
  OCISession *tpcusr = ora_SlotDataP->tpcusr;
  OCIStmt *curi = ora_SlotDataP->curi;

  text stmbuf[SQL_BUF_SIZE];
  sctx = (stoctx *)malloc(sizeof(stoctx));
  memset(sctx,(char)0,sizeof(stoctx));
  ora_SlotDataP->sctx = (void *)sctx;

  memset(&ora_SlotDataP->stoP,(char)0,sizeof(stok_wrapper));
  stoP = &ora_SlotDataP->stoP;

  sctx->norow=0;
```

```
   OCIERROR((OCIError *)errhp,
     OCIHandleAlloc(tpcenv,(dvoid**)&sctx->curs,OCI_HTYPE_STMT,0,(dvoid**)0));
   sprintf ((char *) stmbuf, SQLTXT);
   OCIERROR((OCIError *)errhp,OCIStmtPrepare(sctx->curs,(OCIError
*)errhp,stmbuf,strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
#ifndef PLSQLSTO
   OCIERROR((OCIError *)errhp,
     OCIAttrSet(sctx->curs,OCI_HTYPE_STMT,(dvoid*)&sctx->norow,0,
             OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));
#endif

   /* bind variables */

   OCIBND(sctx->curs,sctx->w_id_bp,(OCIError *)errhp, ":w_id", ADR(stoP-
>in_stok.s_W_ID),sizeof(int),
        SQLT_INT);
   OCIBND(sctx->curs,sctx->d_id_bp,(OCIError *)errhp, ":d_id", ADR(stoP-
>in_stok.s_D_ID),sizeof(int),
     SQLT_INT);
#ifdef USE_IEEE_NUMBER
   OCIBND(sctx->curs,sctx->threshold_bp,(OCIError *)errhp, ":threshold", ADR(stoP-
>in_stok.s_threshold),
     sizeof(float),SQLT_FLT);
#else
   OCIBND(sctx->curs,sctx->threshold_bp,(OCIError *)errhp, ":threshold", ADR(stoP-
>in_stok.s_threshold),
     sizeof(int),SQLT_INT);
#endif
#ifdef PLSQLSTO
   OCIBND(sctx->curs,sctx->low_stock_bp,(OCIError *)errhp,":low_stock" , ADR(stoP-
>out_stok.s_low_stock),
     sizeof(int), SQLT_INT);
#else
   OCIDEFINE(sctx->curs,sctx->low_stock_bp,(OCIError *)errhp, 1, ADR(stoP-
>out_stok.s_low_stock),
     sizeof(int), SQLT_INT);
#endif

   return (0);

}

tkvcs (ora_cn_data_t *ora_SlotDataP)
{
   FILE *server_logtrans = ora_SlotDataP->server_logtrans;
    stoctx *sctx = (stoctx *)ora_SlotDataP->sctx;
    stok_wrapper *stoP = &ora_SlotDataP->stoP;
    OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;
    OCIStmt *curi = ora_SlotDataP->curi;

retry:
    stoP->out_stok.terror= OCIStmtExecute(tpcsvc,sctx->curs,(OCIError *)errhp,1,0,0,0,
       OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
    if (stoP->out_stok.terror != OCI_SUCCESS)
    {
       stoP->out_stok.terror=OCIERROR((OCIError *)errhp,stoP->out_stok.terror);
       OCITransRollback(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
       if((stoP->out_stok.terror == NOT_SERIALIZABLE) || (stoP->out_stok.terror ==
RECOVERR)
           || (stoP->out_stok.terror == SNAPSHOT_TOO_OLD))
       {
          stoP->out_stok.retry++;
          goto retry;
       } else {
          return -1;
       }
    }

    return (0);
}



void tkvcsdone (ora_cn_data_t *ora_SlotDataP)

{
  /*  */
  stoctx *sctx = (stoctx *)ora_SlotDataP->sctx;
  if (sctx) {
    free(sctx);
    ora_SlotDataP->sctx = NULL;
  }

}
```

stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
// tpccOracleGlue.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STDAFX.H
// and not in this file
```

tpccOracleGlue.cpp

```
// tpccOracleGlue.cpp : Defines the entry point for the DLL application.
//

#include "stdafx.h"
#include "../tpccCom/tpccCom.h"
#include "../tpccCom/tpccCom_i.c"
#include "tpccOracleGlue.h"
#include <time.h>
#include <winsock.h>
#include "../tpccisapi/tpccisapi.hpp"

CRITICAL_SECTION debugMutex;
CRITICAL_SECTION errorMutex;
char OracleHome[256];

int loc_count = 0;
CRITICAL_SECTIONcrit_sec;

#define INVALID_ITEM 100

#ifdef _DEBUG
    int debugFlag = 1;
#else
    int debugFlag = 0;
#endif

    void print_time_prefix(FILE *file)
{
    time_t cur_timet;
    char time_str[30];

    cur_timet = time(&cur_timet);
    strftime(time_str, 29, "%X", localtime(&cur_timet));

    fprintf(file, "%s - ",
            time_str);
}

char *get_time_prefix(char *buffer)
{
    time_t cur_timet;
    char time_str[30];
    int len;

    cur_timet = time(&cur_timet);
    strftime(time_str, 29, "%X", localtime(&cur_timet));

    len = sprintf(buffer, "%s - ",
                time_str);
    if (len >= TIME_PREFIX_LEN) {
      fprintf(stderr, "TIME_PREFIX_LEN (%d) too small: %d\n",
              TIME_PREFIX_LEN, len);
      exit(12);
    }
    return(buffer);
}

void oralogprintf(FILE* server_logtrans,char *format, ...)
{
    char formatBuffer[200];
    char *fmt = formatBuffer;
    int fmtLen;
    va_list ap;
    va_start(ap, format);

    fmtLen = TIME_PREFIX_LEN + (int)strlen(format) + 2;
    if (fmtLen > sizeof(formatBuffer)) {
       fmt = (char *)malloc(fmtLen);
    }
    get_time_prefix(fmt);
    strcat(fmt, format);
    if (server_logtrans) {
       vfprintf(server_logtrans, fmt, ap);
       fflush(server_logtrans);
    } else {
       vfprintf(stderr, fmt, ap);
    }
    if (fmt != formatBuffer) free(fmt);
    va_end(ap);
}

/*
 * gettimeofday is not available in the Microsoft C/C++ Run Time
 * and the Win32 API.
 */

/*
 * It is not used and just for unix compatibility.
 */
struct timezone {
       char a;
};

void get_time_init();

int gettimeofday(struct timeval *curTimeP, struct timezone *timezoneP);

#define Li2Double(x) ((double)((x).HighPart) * 4.294967296E9 + (double)((x).LowPart))

LARGE_INTEGER pFreq;
double sFreq;

void get_time_init()
{
    QueryPerformanceFrequency(&pFreq);
    sFreq=Li2Double(pFreq);
```

```
}

void get_local_time(struct timeval *timeP)
{
    double cur_t;
    LARGE_INTEGER counter;

    QueryPerformanceCounter(&counter);
    cur_t = Li2Double(counter) / sFreq;
    timeP->tv_sec = (long)cur_t;
    timeP->tv_usec = ((long)cur_t - timeP->tv_sec) * 1000000;
}

int gettimeofday(struct timeval *curTimeP, struct timezone *timezoneP)
{
    get_local_time(curTimeP);
    return 1;
}

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                       )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        loc_count = 0;
        InitializeCriticalSection(&crit_sec);


    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTERY_SUB_KEY,0,KEY_READ,&r
egisteryKey) == ERROR_SUCCESS)
        {
            regValueSize = 256;
            if (RegQueryValueEx(registeryKey,"OracleHome",0,&regType,(BYTE
*)&value,&regValueSize) == ERROR_SUCCESS) {
                strcpy(OracleHome,value);
            }
            regValueSize = sizeof(value);
            if (RegQueryValueEx(registeryKey,"dbUserName",0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
                strcpy(userName,value);
            else
                return ERR_INVALID_USERNAME;

            regValueSize = sizeof(value);
            if (RegQueryValueEx(registeryKey,"dbPassword",0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
                strcpy(userPassword,value);
            else
                return ERR_INVALID_PASSWORD;
        }
        else
        {
            return ERR_INVALID_REGISTERY_KEY;
        }
        break;
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}

extern "C" TPCCORACLEGLUE_API int connect_db(char *dbName,void **ctx)
{
    FILE    *server_logtrans;
    InitializeCriticalSection(&crit_sec);
    LeaveCriticalSection(&crit_sec);

    server_logtrans = fopen("C:\\inetpub\\wwwroot\\tpcc\\my_debug_gluecode.txt", "w+");
    oralogprintf(server_logtrans,"Opening logfile\n");
    fflush(server_logtrans);
    fclose(server_logtrans);

//  Sleep(30000);
    *ctx = (void *)get_db_ready((ora_cn_data_t *)*ctx, userName, userPassword, dbName);
//  char buf[256];
//  sprintf(buf,"context handler %0X\n",(long long)*ctx);
//  oralogprintf(server_logtrans,buf);
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    if (ctx == NULL)
    {
        oralogprintf(curP->server_logtrans,"Object get_db_ready() failed\n");

        return ERR_SAVING_CONTEXT;
    }
    return OK;
}

extern "C" TPCCORACLEGLUE_API int disconnect_db(void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    clean_connection(curP->server_logtrans, ctx);
    return OK;
}

extern "C" TPCCORACLEGLUE_API int do_nord(nord_wrapper *nord,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    static int numCalls = 0;

    memcpy(&curP->globals,nord,sizeof(nord_wrapper));
    int rc = TPCnew((void *)curP);
    if (rc != 0 && rc != INVALID_ITEM)
    {
        int i;
        oralogprintf(curP->server_logtrans,"Error TPCnew : terror %d, rc %d, retry %d, w_id
%d, d_id %d, c_id %d, o_ol_cnt %d (out cnt %d)\n",
            curP->globals.out_nord.terror, rc, curP->globals.out_nord.retry,
            curP->globals.in_nord.s_W_ID, curP->globals.in_nord.s_D_ID,
            curP->globals.in_nord.s_C_ID, curP->globals.in_nord.s_O_OL_CNT, curP-
>globals.out_nord.s_O_OL_CNT);
        for (i=0; i<15; i++)
        {
            oralogprintf(curP->server_logtrans,"ol_i_id %d, ol_supply_w_id %d, ol_quantity
%d\n",
                curP->globals.in_nord.s_OL_I_ID[i], curP-
>globals.in_nord.s_OL_SUPPLY_W_ID[i],
                curP->globals.in_nord.s_OL_QUANTITY[i]);
        }
    }
    memcpy(nord,&curP->globals,sizeof(nord_wrapper));
    nord->out_nord.s_transtatus = rc == 0 ? TPCC_SUCCESS : curP-
>globals.out_nord.terror;
    if (++numCalls % 10000 == 0) {
        oralogprintf(curP->server_logtrans,"doNewOrder so far %d\n", numCalls);
    }

    return OK;
}

extern "C" TPCCORACLEGLUE_API int do_pymt(paym_wrapper *pymt,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    static int numCalls = 0;
    pymt->in_paym.bylastname = ((pymt->in_paym.s_C_ID == 0) ? 1 : 0);
    memcpy(&curP->payP,pymt,sizeof(paym_wrapper));

    int rc = TPCpay((void *)curP);
    if (rc != 0)
    {
        oralogprintf(curP->server_logtrans,"Error TPCpay: terror %d, rc %d, retry %d, w_id
%d, D_id %d, C_w_id %d, c_id %d, bylastname %d, amount %.2f\n",
            curP->payP.out_paym.terror, rc, curP->payP.out_paym.retry,
            curP->payP.in_paym.s_W_ID,
            curP->payP.in_paym.s_D_ID,
            curP->payP.in_paym.s_C_W_ID,
            curP->payP.in_paym.s_C_ID,
            curP->payP.in_paym.bylastname,
            curP->payP.in_paym.s_H_AMOUNT);
    }
    memcpy(pymt,&curP->payP,sizeof(paym_wrapper));
    pymt->out_paym.s_transtatus = rc == 0 ? TPCC_SUCCESS : curP-
>payP.out_paym.terror;
    if (++numCalls % 10000 == 0) {
        oralogprintf(curP->server_logtrans,"doPayment so far %d\n", numCalls);
    }

    return OK;
}
extern "C"  TPCCORACLEGLUE_API int do_ords(ords_wrapper *ords,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    static int numCalls = 0;

    //  oralogprintf(curP->server_logtrans,"OrderStatus, I'm tid %d\n",ThreadNum);
    ords->in_ords.bylastname = ((ords->in_ords.s_C_ID == 0) ? 1 : 0);
    memcpy(&curP->ordP,ords,sizeof(ords_wrapper));
    int rc = TPCord((void *)curP);
    if (rc != 0)
    {
        oralogprintf(curP->server_logtrans,"Error TPCord: terror %d, rc %d, retry %d, w_id
%d, d_id %d,c_id %d, bylastname %d, c_last %s\n ",
            curP->ordP.out_ords.terror, rc, curP->ordP.out_ords.retry,
            curP->ordP.in_ords.s_W_ID, curP->ordP.in_ords.s_D_ID, curP-
>ordP.in_ords.s_C_ID,
            curP->ordP.in_ords.bylastname, curP->ordP.in_ords.s_C_LAST);
    }
    memcpy(ords,&curP->ordP,sizeof(ords_wrapper));
    ords->out_ords.s_transtatus = rc == 0 ? TPCC_SUCCESS : curP->ordP.out_ords.terror;
    if (++numCalls % 10000 == 0) {
        oralogprintf(curP->server_logtrans,"doOrderStatus so far %d\n", numCalls);
    }

    return OK;
}


extern "C" TPCCORACLEGLUE_API int do_dlvy(dlvy_wrapper *dlvy,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    dlvy_wrapper *delP=&(curP->delP);
    FILE *server_logtrans = curP->server_logtrans;
    dvoid *errhp = curP->errhp;
    int rc;
    static int numCalls = 0;

    memcpy(&curP->delP,dlvy,sizeof(dlvy_wrapper));
    delP->out_dlvy.retry = 0;
    delP->in_dlvy.plsqlflag = 1;

    OCIERROR(errhp,OCIDateSysDate((OCIError *)errhp,&delP->in_dlvy.cr_date));

    if (rc = tkvcd (curP)) {
        if(delP->out_dlvy.terror == DEL_ERROR)
```

```
            return DEL_ERROR;
        return (-1);
    }
    memcpy(dlvy,&curP->delP,sizeof(dlvy_wrapper));
    dlvy->out_dlvy.s_transtatus = rc == 0 ? TPCC_SUCCESS : delP->out_dlvy.terror;
    if (++numCalls % 10000 == 0) {
        oralogprintf(curP->server_logtrans,"doOrderStatus so far %d\n", numCalls);
    }

    return OK;
}

extern "C" TPCCORACLEGLUE_API int do_stok(stok_wrapper *stok,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    static int numCalls = 0;
    FILE *server_logtrans = curP->server_logtrans;
    memcpy(&curP->stoP,stok,sizeof(stok_wrapper));
    //  oralogprintf(curP->server_logtrans,"StockLevel, I'm tid %d\n",ThreadNum);

    //call stock level txn
    int rc = TPCsto((void *)curP);
    if ( rc != 0 ) {
        oralogprintf(curP->server_logtrans,"Error TPCsto : terror %d, rc %d, retry %d, w_id
%d, d_id %d, threshold %d\n",
            curP->stoP.out_stok.terror, rc, curP->stoP.out_stok.retry,
            curP->stoP.in_stok.s_W_ID, curP->stoP.in_stok.s_D_ID,
            curP->stoP.in_stok.s_threshold);
    }
    memcpy(stok,&curP->stoP,sizeof(stok_wrapper));
    stok->out_stok.s_transtatus = rc == 0 ? TPCC_SUCCESS : curP->stoP.out_stok.terror;
    if (++numCalls % 10000 == 0) {
        oralogprintf(curP->server_logtrans,"doStockLevel so far %d\n", numCalls);
    }

    return OK;
}
```

tpccpl.cpp

```
#include "../stdafx.h"
#define ORA_NT
#include <stdio.h>
#include <time.h>
#include "../../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"
#include <stdarg.h>

extern int server_null_test;
extern int loc_count;

extern CRITICAL_SECTION debugMutex;

#define INVALID_ITEM 100

#define SQLTXT "alter session set isolation_level = serializable"
#define SQLTXTTRC "alter session set sql_trace = true"
#define SQLTXTTIM "alter session set timed_statistics = true"
#define SQLTXTIMU "alter session set \"_in_memory_undo\" = true"
#define SQLTXTNCP "alter session set plsql_compiler_flags=anonymous_block_native"

#ifdef ORA_NT
#undef boolean
#include "dpbcore.h"
#define gettime dpbtimef
#else
extern double gettime ();
#endif

static int init_cn_data(struct ora_cn_data_t *dataP);
static void initOCIhandles(struct ora_cn_data_t *cn_dataP, char* uid, char *pwd);

extern void oralogprintf(FILE *,char *format, ...);

int proc_no;
static char *db_uid;
static char *db_pwd;

#ifndef AVOID_DEADLOCK
void swap(nord_wrapper *newP, int i, int j, int *indx, int in);
void q_sort(nord_wrapper *newP,int left, int right, int *indx, int in);
#endif

#ifndef TUX
void userlog (char* fmtp, ...)
{
  va_list va;
  va_start(va,fmtp);
  vfprintf(stderr,fmtp,va);
  va_end(va);
}
#endif


/* vmm313 void ocierror(fname, lineno, errhp, status)  */
int ocierror(FILE *server_logtrans,char *fname, int lineno, dvoid *errhp, sword status)
{
  text errbuf[512];
  sb4 errcode;
  sb4 lstat;
  ub4 recno=2;


  switch (status) {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname, lineno);
    oralogprintf(server_logtrans,"Error - OCI_SUCCESS_WITH_INFO\n");
    lstat = OCIErrorGet ((OCIError *)errhp, recno++, (text *) NULL, &errcode, errbuf,
                         (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    oralogprintf(server_logtrans,"Error - %s\n", errbuf);
    break;
  case OCI_NEED_DATA:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname, lineno);
    oralogprintf(server_logtrans,"Error - OCI_NEED_DATA\n");
    return (IRRECERR);
  case OCI_NO_DATA:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname, lineno);
    oralogprintf(server_logtrans,"Error - OCI_NO_DATA\n");
    return (IRRECERR);
  case OCI_ERROR:
    lstat = OCIErrorGet ((OCIError *)errhp, (ub4) 1,
          (text *) NULL, &errcode, errbuf,
           (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    if (errcode == NOT_SERIALIZABLE) return (errcode);
    if (errcode == SNAPSHOT_TOO_OLD) return (errcode);
    while (lstat != OCI_NO_DATA)
    {
      oralogprintf(server_logtrans,"Module %s Line %d\n", fname, lineno);
      oralogprintf(server_logtrans,"Error - %s\n", errbuf);
      lstat = OCIErrorGet ((OCIError *)errhp, recno++, (text *) NULL, &errcode, errbuf,
                           (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    }
    return (errcode);
/* vmm313    TPCexit(1); */
/* vmm313    exit(1); */
  case OCI_INVALID_HANDLE:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname, lineno);
    oralogprintf(server_logtrans,"Error - OCI_INVALID_HANDLE\n");
    TPCexit();
    exit(-1);
  case OCI_STILL_EXECUTING:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname, lineno);
    oralogprintf(server_logtrans,"Error - OCI_STILL_EXECUTE\n");
    return (IRRECERR);
  case OCI_CONTINUE:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname, lineno);
    oralogprintf(server_logtrans,"Error - OCI_CONTINUE\n");
    return (IRRECERR);
  default:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname, lineno);
    oralogprintf(server_logtrans,"Status - %s\n", status);
    return (IRRECERR);
  }
  return (RECOVERR);
}


FILE  *vopen(char *fnam,char *mode)
{
FILE *fd;

#ifdef  DEBUG
  fprintf(stderr, "tkvuopen() fnam: %s,   mode: %s\n", fnam, mode);
#endif

    fd = fopen((char *)fnam,(char *)mode);
    if (!fd){
        fprintf(stderr," fopen on %s failed\n",fnam);
        exit(-1);
    }
    return(fd);
}

int sqlfile(FILE* server_logtrans,char *fnam, text *linebuf)
{
FILE *fd;
int nulpt = 0;
char realfile[512];

//DEBUGP(( "%s: %d sqlfile() fnam: %s,  linebuf: %#x\n", __FILE__, __LINE__, fnam,
linebuf));

/*
    sprintf(realfile,"%s/bench/tpc/tpcc/blocks/%s",oracle_home,fnam);
*/
    sprintf(realfile,"%s",fnam);
    if ((fd = vopen(realfile,"r")) == NULL) {
        oralogprintf(server_logtrans,"%s: %d >sqlfile vopen failed\n", __FILE__, __LINE__);
        return(-1);
    }
    while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
    {
        nulpt = (int)strlen((char *)linebuf);
    }
    oralogprintf(server_logtrans,"%s: %d <sqlfile\n", __FILE__, __LINE__);
    return(nulpt);
}

#ifdef NOT
void vgetdate (unsigned char *oradt)
{
 struct tm *loctime;
 time_t  int_time;

 struct ORADATE {
 unsigned char  century;
 unsigned char  year;
```

```
 unsigned char  month;
 unsigned char  day;
 unsigned char  hour;
 unsigned char  minute;
 unsigned char  second;
 } Date;
int century;
int cnvrtOK;

/* assume convert is successful */
cnvrtOK = 1;

/* get the current date and time as an integer */
time( &int_time);

/* Convert the current date and time into local time */
loctime = localtime( &int_time);

century = (1900+loctime->tm_year) / 100;

Date.century = (unsigned char)(century + 100);
if (Date.century < 119 || Date.century > 120) cnvrtOK = 0;
Date.year  = (unsigned char)(loctime->tm_year+100);
if (Date.year < 100 || Date.year > 199) cnvrtOK = 0;
Date.month = (unsigned char)(loctime->tm_mon + 1);
if (Date.month < 1 || Date.month > 12) cnvrtOK = 0;
Date.day   = (unsigned char)loctime->tm_mday;
if (Date.day < 1 || Date.day > 31) cnvrtOK = 0;
Date.hour  = (unsigned char)(loctime->tm_hour + 1);
if (Date.hour < 1 || Date.hour > 24) cnvrtOK = 0;
Date.minute= (unsigned char)(loctime->tm_min + 1);
if (Date.minute < 1 || Date.minute > 60) cnvrtOK = 0;
Date.second= (unsigned char)(loctime->tm_sec + 1);
if (Date.second < 1 || Date.second > 60) cnvrtOK = 0;


if (cnvrtOK)
  memcpy(oradt,&Date,7);
  else
    *oradt = '\0';

 return;

}
void cvtdmy (unsigned char *oradt, char *outdate)
{

       struct ORADATE {
              unsigned char   century;
              unsigned char   year;
              unsigned char   month;
              unsigned char   day;
              unsigned char   hour;
              unsigned char   minute;
              unsigned char   second;
       } Date;


       int day,month,year;

       memcpy(&Date,oradt,7);

       year = (Date.century-100)*100 + Date.year-100;
       month = Date.month;
       day = Date.day;
       sprintf(outdate,"%02d-%02d-%4d\0",day,month,year);

       return;
}
```

```
void cvtdmyhms (unsigned char *oradt, char *outdate)
{

       struct ORADATE {
              unsigned char   century;
              unsigned char   year;
              unsigned char   month;
              unsigned char   day;
              unsigned char   hour;
              unsigned char   minute;
              unsigned char   second;
       } Date;


       int day,month,year;
       int hour,min,sec;

       memcpy(&Date,oradt,7);

       year = (Date.century-100)*100 + Date.year-100;
       month = Date.month;
       day = Date.day;
       hour = Date.hour - 1;
       min = Date.minute - 1;
       sec = Date.second - 1;

       sprintf(outdate,"%02d-%02d-%4d %02d:%02d:%02d\0",
                     day,month,year,hour,min,sec);

       return;
}
#endif


void TPCexit (void)
{
//  free(Ctpcc_com::get_cur());
}

void clean_connection(FILE* server_logtrans,void *ptr)
{

  /* free trans specific cursor handles first and later the ora handles */
  ora_cn_data_t *cn_dataP = (ora_cn_data_t *)ptr;

  if (cn_dataP != NULL) {
       OCIServer *tpcsrv;
       OCISession *tpcusr;
       OCIEnv *tpcenv;
       dvoid *errhp;
       OCISvcCtx *tpcsvc;

       oralogprintf(server_logtrans, "clean_connection, Freeing OCI handles\n");

        tkvcpdone(cn_dataP);
       tkvcsdone(cn_dataP);
       tkvcodone(cn_dataP);
       tkvcndone(cn_dataP);
       /* free OCI handles */
       if (tpcusr = cn_dataP->tpcusr) {
              oralogprintf(server_logtrans, "free_handles> OCIHandleFree tpcusr\n");
              OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
       }
       if (tpcsvc = cn_dataP->tpcsvc) {
              oralogprintf(server_logtrans, "free_handles> OCIHandleFree tpcsvc\n");
              OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
       }
       if (errhp = cn_dataP->errhp) {
              oralogprintf(server_logtrans, "free_handles> OCIHandleFree errhp\n");
              OCIHandleFree((OCIError *)errhp, OCI_HTYPE_ERROR);
       }
       if (tpcsrv = cn_dataP->tpcsrv) {
              oralogprintf(server_logtrans, "free_handles> OCIHandleFree tpcsrv\n");
              OCIHandleFree((OCIError *)tpcsrv, OCI_HTYPE_SERVER);
       }
       if (tpcenv = cn_dataP->tpcenv) {
              oralogprintf(server_logtrans, "free_handles> OCIHandleFree tpcenv\n");
              OCIHandleFree((OCIError *)tpcenv, OCI_HTYPE_ENV);
       }

       oralogprintf(server_logtrans, "free_handles> free cn_dataP\n");
  }
}



static void initOCIhandles(struct ora_cn_data_t *cn_dataP, char *uid, char *pwd,char
*dbName)
{

  int tracelevel = 0;  /* new define */
  OCIDate cr_date;
  text stmbuf[100];
  OCIEnv *tpcenv=NULL;
  OCIServer *tpcsrv=NULL;
  OCIError *errhp;
  OCISvcCtx *tpcsvc;
  OCISession *tpcusr;
  OCIStmt *curi;

  oralogprintf(cn_dataP->server_logtrans,">> initOCIhandles uid %s pwd %s dbName
%s\n",uid,pwd,dbName);
  OCIEnvCreate (&tpcenv,OCI_OBJECT|OCI_THREADED, (dvoid *)0, 0, 0, 0, (size_t)0,
(dvoid **)0);
  if (tpcenv==NULL) {
     oralogprintf(cn_dataP->server_logtrans,"Cannot get handle to environment,
OCIEnvCreate failed\n");
  }
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid
**)0);

  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid **)0);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid
**)0);
//EnterCriticalSection(&debugMutex);
  ocierror(cn_dataP->server_logtrans,__FILE__,__LINE__,errhp, OCIServerAttach(tpcsrv,
(OCIError *)errhp, (text *)dbName,0,OCI_DEFAULT));
  if (tpcsrv==NULL) {
     oralogprintf(cn_dataP->server_logtrans,"Cannot get handle to server,
OCIServerAttach failed\n");
  }
   oralogprintf(cn_dataP->server_logtrans,"OCIServerAttach returned %x\n",tpcsrv);
//LeaveCriticalSection(&debugMutex);
  OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SERVER, (OCIError *)errhp);

  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);

  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)4,OCI_ATTR_USERNAME, (OCIError *)errhp);

  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)4,
           OCI_ATTR_PASSWORD, (OCIError *)errhp);
  ocierror(cn_dataP->server_logtrans,__FILE__,__LINE__,errhp, OCISessionBegin(tpcsvc,
(OCIError *)errhp, tpcusr, OCI_CRED_RDBMS, OCI_DEFAULT));

  OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_SESSION, (OCIError
*)errhp);
```

```
   /* run all transaction in serializable mode */

   OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT);
   OCIStmtPrepare(curi, (OCIError *)errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
   ocierror(cn_dataP->server_logtrans,__FILE__,__LINE__,errhp,OCIStmtExecute(tpcsvc,
curi, (OCIError *)errhp,1,0,0,0,OCI_DEFAULT));
   OCIHandleFree(curi, OCI_HTYPE_STMT);


   if (getenv("USE_IMU")) {
      printf("Use in_memory_undo\n");
      OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
      sprintf ((char *) stmbuf, SQLTXTIMU);
      OCIStmtPrepare(curi, (OCIError *)errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
      ocierror(cn_dataP->server_logtrans,__FILE__,__LINE__,errhp,OCIStmtExecute(tpcsvc,
curi, (OCIError *)errhp,1,0,0,0,OCI_DEFAULT));
      OCIHandleFree(curi, OCI_HTYPE_STMT);
   }

   if (getenv("USE_NCOMP")) {
      printf("Use NCOMP\n");
      OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
      sprintf ((char *) stmbuf, SQLTXTNCP);
      OCIStmtPrepare(curi, (OCIError *)errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
      ocierror(cn_dataP->server_logtrans,__FILE__,__LINE__,errhp,OCIStmtExecute(tpcsvc,
curi, (OCIError *)errhp,1,0,0,0,OCI_DEFAULT));
      OCIHandleFree(curi, OCI_HTYPE_STMT);
   }

   if (tracelevel == 3) {
     OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
     memset(stmbuf,0,100);
     sprintf ((char *) stmbuf, SQLTXTTIM);
     OCIStmtPrepare(curi, (OCIError *)errhp, stmbuf, strlen((char *)stmbuf),
                    OCI_NTV_SYNTAX, OCI_DEFAULT);
     ocierror(cn_dataP->server_logtrans,__FILE__,__LINE__,errhp,  OCIStmtExecute(tpcsvc,
curi, (OCIError *)errhp,1,0,0,0,OCI_DEFAULT));
     OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
   }


   ocierror(cn_dataP-
>server_logtrans,__FILE__,__LINE__,errhp,OCIDateSysDate((OCIError *)errhp,&cr_date));

    /*  Store the handles just initialized in the thread slot
    */
   cn_dataP->tpcenv = tpcenv;
   cn_dataP->tpcsrv = tpcsrv;
   cn_dataP->errhp  = errhp;
   cn_dataP->tpcsvc = tpcsvc;
   cn_dataP->tpcusr = tpcusr;
   cn_dataP->curi   = curi;
   oralogprintf(cn_dataP->server_logtrans,"<< initOCIhandles\n");
}

/*
 *  init_cn_data
 *      Initializes all the transactions for a single connection
 */
static int init_cn_data(ora_cn_data_t *cnP,char *uid,char *pwd,char *dbName)
{
    int status;

    oralogprintf(cnP->server_logtrans,">> init_cn_data\n");
    initOCIhandles(cnP, uid,pwd,dbName);

    if (status = tkvcninit (cnP)) {
       fprintf(stderr, "tkvcninit failed: %d\n", status);
       TPCexit ();
       return (status);
    }

    if (status = tkvcpinit (cnP)) {
       fprintf(stderr, "tkvcpinit failed: %d\n", status);
       TPCexit ();
       return (status);
    }

    if (status = tkvcoinit (cnP)) {
       fprintf(stderr, "tkvcoinit failed: %d\n", status);
       TPCexit ();
       return (status);
    }

    if (status = tkvcdinit (cnP)) {
       fprintf(stderr, "tkvcdinit failed: %d\n", status);
       TPCexit ();
       return (status);
    }

    if (status = tkvcsinit (cnP)) {
       fprintf(stderr, "tkvcsinit failed: %d\n", status);
       TPCexit ();
       return (status);
    }
return 0;
}


void *create_ora_connection(char *uid, char *pwd, char *dbName) {
    HKEY Key;
    char  value[256];
    DWORD   regValueSize = 256;
    DWORD   regType;
    FILE    *server_logtrans;

    ora_cn_data_t *cnP = (ora_cn_data_t *)malloc(sizeof(ora_cn_data_t));

    if (RegOpenKeyEx(HKEY_LOCAL_MACHINE,"SOFTWARE\\TPCC", 0, KEY_READ,
&Key) == ERROR_SUCCESS)
    {
        regValueSize = 256;
        if (RegQueryValueEx(Key,"OraComServLog",0,&regType,(BYTE
*)&value,&regValueSize) == ERROR_SUCCESS)
        {
           sprintf(value,"%s%d.log",value,loc_count++);
           server_logtrans = fopen(value, "w+");
           oralogprintf(server_logtrans,"Opening logfile %s\n",value);
        }
    }
    cnP->server_logtrans = server_logtrans;
    oralogprintf(server_logtrans,">> create_ora_connection\n");
    if (cnP!=NULL)
      init_cn_data(cnP,uid,pwd,dbName);
    else
       oralogprintf(server_logtrans,"Cannot allocate ora_cn_data_t.\n");
    return (void *)cnP;
}

ora_cn_data_t *get_db_ready(ora_cn_data_t *curP, char *uid, char *pwd, char *dbName)
{

    curP = (ora_cn_data_t *)create_ora_connection(uid,pwd,dbName);
    if (curP==NULL) {
       oralogprintf(curP->server_logtrans,"create_ora_connection failed!\n");
    }
    return(curP);
}

TPCnew (void *cnP)
{
   int i;
   ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
   FILE *server_logtrans = cn_dataP->server_logtrans;
   nord_wrapper *newP = &cn_dataP->globals;
   int indx[NITEMS], ordl_cnt=newP->out_nord.s_O_OL_CNT;
   dvoid *errhp = cn_dataP->errhp;

   newP->out_nord.retry = 0;

#ifndef AVOID_DEADLOCK

   for (i = NITEMS; i > 0; i--) {
      if (newP->in_nord.s_OL_I_ID[i-1] > 0) {
        ordl_cnt = i;
        break;
      }
   }

   for (i = 0; i < NITEMS; i++) indx[i] = i;

    q_sort(newP, 0, ordl_cnt-1, indx, 1);

#endif


   OCIERROR(errhp,OCIDateSysDate((OCIError *)errhp,&newP->in_nord.cr_date));

   if (tkvcn (cn_dataP)) {
      if (newP->out_nord.terror != RECOVERR)
      {
        if (newP->out_nord.status) {
          return(newP->out_nord.terror);
        }
        else {
          newP->out_nord.terror = INVALID_STATUS;
          return (INVALID_STATUS);
        }
      }
   }


   newP->out_nord.datelen = sizeof(newP->out_nord.s_O_ENTRY_D_time);
    OCIERROR(errhp,
       OCIDateToText((OCIError *)errhp,&newP-
>in_nord.cr_date,(text*)FULLDATE,SIZ(FULLDATE),(text*)0,0,
        (ub4 *)&newP->out_nord.datelen,(text *)newP->out_nord.s_O_ENTRY_D_time));



#ifndef AVOID_DEADLOCK
    q_sort(newP, 0, ordl_cnt-1, indx, 0);
#endif
     newP->out_nord.terror = 0;

   return (0);
}

TPCpay (void *cnP)
{
   ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
   FILE *server_logtrans = cn_dataP->server_logtrans;
   paym_wrapper *payP = &cn_dataP->payP;
   dvoid *errhp = cn_dataP->errhp;
```

```
   char months[13][4] = {"",
"JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP","OCT","NOV","DEC"};

   OCIERROR(errhp,OCIDateSysDate((OCIError *)errhp,&payP->in_paym.cr_date));

   sprintf(payP->out_paym.s_H_DATE_time,"%d-%s-%d %d:%d:%d",payP-
>in_paym.cr_date.OCIDateDD,
    months[payP->in_paym.cr_date.OCIDateMM],   payP->in_paym.cr_date.OCIDateYYYY,
    payP->in_paym.cr_date.OCIDateTime.OCITimeHH,   payP-
>in_paym.cr_date.OCIDateTime.OCITimeMI,
    payP->in_paym.cr_date.OCIDateTime.OCITimeSS);

   if (payP->in_paym.bylastname) {
     payP->in_paym.s_C_ID = 0;
   }
   else {
      payP->in_paym.s_C_LAST[0] = ' ';
   }

   if (tkvcp (cn_dataP)) {
      if (payP->out_paym.terror != RECOVERR)
        payP->out_paym.terror = IRRECERR;
     return (-1);
   }

    sprintf(payP->out_paym.c_since_d,"%d-%s-%d",payP-
>out_paym.c_since.OCIDateDD,months[payP->out_paym.c_since.OCIDateMM],
    payP->out_paym.c_since.OCIDateYYYY);

    payP->out_paym.terror = 0;

   return (0);

}


TPCord (void *cnP)
{
   ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
   ords_wrapper *ordP = &cn_dataP->ordP;

   if (tkvco (cn_dataP)) {
     if (ordP->out_ords.terror != RECOVERR)
       ordP->out_ords.terror = IRRECERR;
     return (-1);
   }

   ordP->out_ords.terror = 0;
   if ( ordP->out_ords.s_O_CARRIER_ID == 11 )
      ordP->out_ords.s_O_CARRIER_ID = 0;

   return (0);

}

int TPCdel (void *cnP)
{
   ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
   FILE *server_logtrans = cn_dataP->server_logtrans;
   dlvy_wrapper *str = &cn_dataP->delP;
   dvoid *errhp = cn_dataP->errhp;
   OCIDate cr_date;

   OCIERROR(errhp,OCIDateSysDate((OCIError *)errhp,&cr_date));

   if (tkvcd (cn_dataP)) {
     if(str->out_dlvy.terror == DEL_ERROR)
       return DEL_ERROR;
     if (str->out_dlvy.terror != RECOVERR)
       str->out_dlvy.terror = IRRECERR;
     return (-1);
   }

   str->out_dlvy.terror = 0;
   return (0);
}


TPCsto (void *cnP)
{
   ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
   stok_wrapper *stoP = &cn_dataP->stoP;

   stoP->out_stok.retry = 0;

   if (tkvcs (cn_dataP)) {
     if (stoP->out_stok.terror != RECOVERR)
       stoP->out_stok.terror = IRRECERR;
     return (-1);
   }

   stoP->out_stok.terror = 0;

   return (0);

}

#ifndef AVOID_DEADLOCK

void q_sort(nord_wrapper *newP,int left, int right, int *indx, int in)
{
  int i, last;

  if(left >= right)
    return;
  swap(newP,left,(left+right)/2, indx, in);
  last = left;
  for(i=left+1;i<=right;i++)
     if(newP->in_nord.s_OL_I_ID[i] < newP->in_nord.s_OL_I_ID[left])
     swap(newP,last,i, indx, in);
  swap(newP, left,last, indx, in);
  q_sort(newP,left,last-1, indx, in);
  q_sort(newP,last+1,right, indx, in);
}

void swap(struct nord_wrapper *newP, int i, int j, int *indx, int in)
{
  int tempd;
  int tempi;
  char tmpstr[25];
  char tmpch;

  tempi = indx[i];
  indx[i] = indx[j];
  indx[j] = tempi;

  if (in) {

     tempi = newP->in_nord.s_OL_I_ID[i];
      newP->in_nord.s_OL_I_ID[i] = newP->in_nord.s_OL_I_ID[j];
      newP->in_nord.s_OL_I_ID[j] = tempi;
      tempi = newP->in_nord.s_OL_SUPPLY_W_ID[i];
      newP->in_nord.s_OL_SUPPLY_W_ID[i] = newP->in_nord.s_OL_SUPPLY_W_ID[j];
      newP->in_nord.s_OL_SUPPLY_W_ID[j] = tempi;

      tempi = newP->in_nord.s_OL_QUANTITY[i];
      newP->in_nord.s_OL_QUANTITY[i] = newP->in_nord.s_OL_QUANTITY[j];
      newP->in_nord.s_OL_QUANTITY[j] = (short)tempi;
  } else {
      strcpy(tmpstr,newP->out_nord.s_I_NAME[i]);
      strcpy(newP->out_nord.s_I_NAME[i],newP->out_nord.s_I_NAME[j]);
      strcpy(newP->out_nord.s_I_NAME[j],tmpstr);

      tempi = newP->out_nord.s_S_QUANTITY[i];
      newP->out_nord.s_S_QUANTITY[i] = newP->out_nord.s_S_QUANTITY[j];
      newP->out_nord.s_S_QUANTITY[j] = (short)tempi;

      tmpch = newP->out_nord.s_brand_generic[i];
      newP->out_nord.s_brand_generic[i] = newP->out_nord.s_brand_generic[j];

      newP->out_nord.s_brand_generic[j] = tmpch;

      tempd = newP->out_nord.s_I_PRICE[i];
      newP->out_nord.s_I_PRICE[i] = newP->out_nord.s_I_PRICE[j];
      newP->out_nord.s_I_PRICE[j] = tempd;

      tempd = newP->out_nord.s_OL_AMOUNT[i];
      newP->out_nord.s_OL_AMOUNT[i] = newP->out_nord.s_OL_AMOUNT[j];
      newP->out_nord.s_OL_AMOUNT[j] = tempd;
  }
}

#endif
```

ora_tpcc.h

```
/*
 * $Header: tpcc.h 7030100.1 95/07/19 15:10:55 plai Generic<base> $ Copyr (c) 1993
Oracle
 */
/*=============================================================================+
|        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
|                 OPEN SYSTEMS PERFORMANCE GROUP                  |
|                      All Rights Reserved                        |
 +=============================================================================+
| FILENAME
|    tpcc.h
| DESCRIPTION
|    Include file for TPC-C benchmark programs.
 +=============================================================================*/

#ifndef TPCC_H
#define TPCC_H


#ifndef FALSE
# define FALSE 0
#endif

#ifndef TRUE
# define TRUE 1
#endif

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

#ifndef boolean
#define boolean int
#endif

#include <tpccflags.h>

#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>
/*
#ifdef __STDC__
```

```
#include "ociapr.h"
#else
#include "ocikpr.h"
#endif
*/

typedef struct cda_def csrdef;
typedef struct cda_def ldadef;

typedef __int16 int16_t;
typedef __int32 int32_t;
typedef __int64 int64_t;


/* TPC-C transaction functions */

extern int TPCinit ();
extern int TPCnew ();
extern int TPCpay ();
extern int TPCord ();
extern int TPCdel ();
extern int TPCsto ();
extern void TPCexit ();
extern int TPCdumpinit ();
extern void TPCdumpnew ();
extern void TPCdumppay ();
extern void TPCdumpord ();
extern void TPCdumpdel ();
extern void TPCdumpsto ();
extern void TPCdumpexit ();
extern void userlog(char* fmtp, ...);


/* Error codes */

#define RECOVERR -10
#define IRRECERR -20
#define NOERR    111
#define DEL_ERROR -666
#define DEL_DATE_LEN 7
#define NDISTS 10
#define NITEMS 15
#define SQL_BUF_SIZE 8192

#define FULLDATE "dd-mon-yy.hh24:mi:ss"
#define SHORTDATE "dd-mm-yyyy"

#define DELRT 80.0


extern int tkvcss (); /* for alter session to get memory size and trace */
extern boolean multitranx;
extern int ord_init;


extern void errrpt ();
extern int ocierror(FILE *server_logtrans,char *fname, int lineno,dvoid *errhp, sword status);
//extern int sqlfile(char *fname, text *linebuf);

extern FILE *lfp;
//extern FILE *fopen ();
extern int proc_no;
extern int doid[];

/* Miscellaneous */

#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 */
#define SNAPSHOT_TOO_OLD  1555  /* ORA-01555: snapshot too old */

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

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

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

#define min(x,y)  (((x) < (y)) ? (x) : (y))

#define OCIERROR(errp,function)\
       ocierror(server_logtrans,__FILE__,__LINE__,(dvoid *)(errp),(function));

#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
      ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
           OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      ocierror(server_logtrans,__FILE__,__LINE__, (errp), \
           OCIBindByName((stmp), &(bndp), (errp), \
           (text *)(sqlvar), strlen((sqlvar)),\
           (progv), (progvl), (ftype),0,0,0,0,0,OCI_DEFAULT));

/* bind arrays for sql */
#define OCIBNDRA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
      DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
        OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
                (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));

/* use with callback data */
#define OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,\
                  cbf_nodata,cbf_data) \
      DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
          OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar), \
                    strlen((sqlvar)),0,(progvl),(ftype), \
                    indp,0,0,0,0,OCI_DATA_AT_EXEC)); \
      DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
         OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)));

/* bind in/out for plsql without indicator and rcode */
#define OCIBNDPL(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen) \
    DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
     OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
    DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
      OCIBindByName((stmp),&(bndp),(errp),(CONST text *)(sqlvar), \
       (sb4)strlen((CONST char *)(sqlvar)), (dvoid*)(progv),(progvl),(ftype),\
            NULLP(dvoid),(alen), NULLP(ub2), 0,NULLP(ub4),OCI_DEFAULT));

/* bind  in values for plsql with indicator and rcode */
#define OCIBNDR(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
      DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
       OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
          (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0, \
               OCI_DEFAULT));

/* bind in/out for plsql arrays witout indicator and rcode */
#define OCIBNDPLA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen,ms,cu) \
      DISCARD ocierror(server_logtrans,__FILE__,__LINE__, (errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));\
      DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp),\
       OCIBindByName((stmp),&(bndp),(errp),(CONST text *)(sqlvar), \
           (sb4)strlen((CONST char *) (sqlvar)),(void *)(progv), \
           (progvl),(ftype),NULL,(alen),NULL,(ms),(cu),OCI_DEFAULT));

/* bind in/out values for plsql with indicator and rcode */
#define OCIBNDRAA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode,\
              ms,cu) \
      ocierror(server_logtrans,__FILE__,__LINE__, (errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));\
      ocierror(server_logtrans,__FILE__,__LINE__,(errp),\
       OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
      (progv),(progvl),(ftype),(indp),(alen),(arcode),(ms),(cu),OCI_DEFAULT));

#define OCIDEFINE(stmp,dfnp,errp,pos,progv,progvl,ftype)\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),(ftype),\
           0,0,0,OCI_DEFAULT);


#define OCIDEF(stmp,dfnp,errp,pos,progv,progvl,ftype) \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                  (dvoid**)0);\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),\
                  (ftype),NULL,NULL,NULL,OCI_DEFAULT); \


#define OCIDFNRA(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,alen,arcode) \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                  (dvoid**)0);\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
                  (progvl),(ftype),(indp),(alen),\
                  (arcode),OCI_DEFAULT);

#define OCIDFNDYN(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,ctxp,cbf_data) \
      ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                  (dvoid**)0));\
      ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv), (progvl),(ftype),\
                          (indp),NULL,NULL, OCI_DYNAMIC_FETCH));\
      ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
      OCIDefineDynamic((dfnp),(errp),(ctxp),(cbf_data)));


struct out_stocklev_struct {
   int32_t terror;
   int32_t s_low_stock;
   int32_t retry;
   int16_t  s_transtatus;
};

struct in_stocklev_struct {
   int32_t  s_threshold;
   int32_t  s_D_ID;
   int32_t    s_W_ID;
};


struct in_neword_struct {
  int32_t  s_OL_I_ID[15];
  int32_t  s_OL_SUPPLY_W_ID[15];
  int32_t  s_OL_QUANTITY[15];
```

```
  int32_t  s_C_ID;
  int32_t  s_W_ID;
  int32_t  s_D_ID;
  int32_t  s_O_OL_CNT;
  int32_t  s_all_local;
  int32_t  duplicate_items;
  OCIDate cr_date;
};

struct out_neword_struct {
  int32_t  s_I_PRICE[15];
  int32_t  s_OL_AMOUNT[15];
  int32_t  s_S_QUANTITY[15];
  char     s_I_NAME[15][25];
  char     s_brand_generic[15];
  char  s_O_ENTRY_D_time[22];
  float  s_W_TAX;
  float  s_D_TAX;
  float  s_C_DISCOUNT;
  float  s_total_amount;
  int32_t  s_O_ID;
  int32_t  s_O_OL_CNT;
  int16_t  s_transtatus;
  int16_t  deadlocks;
  int16_t  datelen;
  char     s_C_LAST[17];
  char     s_C_CREDIT[3];
  int32_t  terror;
  int32_t  status;
  int32_t  retry;
  int16_t  items_valid;
};

struct in_payment_struct {
  float  s_H_AMOUNT;
  int32_t  s_W_ID;
  int32_t  s_C_W_ID;
  int32_t  s_C_ID;
  int32_t  s_C_D_ID;
  int32_t  s_D_ID;
  char     s_C_LAST[17];
  int32_t  bylastname;
  OCIDate cr_date;
};

struct out_payment_struct {
  char  s_H_DATE_time[22];
  OCIDate c_since;
  char c_since_d[12];
  double  s_C_CREDIT_LIM;
  double  s_C_BALANCE;
  float  s_C_DISCOUNT;
  int32_t  s_C_ID;
  int16_t  s_transtatus;
  int16_t  since_len;
  int32_t  terror;
  int16_t  hlen;
  int32_t  retry;
  char     s_W_STREET_1[21];
  char     s_W_STREET_2[21];
  char     s_W_CITY[21];
  char     s_W_STATE[3];
  char     s_W_ZIP[10];
  char     s_D_STREET_1[21];
  char     s_D_STREET_2[21];
  char     s_D_CITY[21];
  char     s_D_STATE[3];
  char     s_D_ZIP[10];
  char     s_C_FIRST[17];
  char     s_C_MIDDLE[3];
  char     s_C_LAST[17];
  char     s_C_STREET_1[21];
  char     s_C_STREET_2[21];
  char     s_C_CITY[21];
  char     s_C_STATE[3];
  char     s_C_ZIP[10];
  char     s_C_PHONE[17];
  char     s_C_CREDIT[3];
  char     s_C_DATA[201];
};

struct in_ordstat_struct {
  int32_t  s_C_ID;
  int32_t  s_W_ID;
  int32_t  s_D_ID;
  int32_t  bylastname;
  char     s_C_LAST[17];
  OCIDate  cr_date;
};

struct out_ordstat_struct {
  double   s_C_BALANCE;
  char     s_O_ENTRY_D_time[22];
  int32_t  s_C_ID;
  int32_t  s_O_ID;
  int32_t  s_O_CARRIER_ID;
  int32_t  s_ol_cnt;
  int32_t  terror;
  char     s_OL_DELIVERY_D_time[15][22];
  OCIDate  s_OL_DELIVERY_D_base[15];
  int32_t  s_OL_AMOUNT[15];
  int32_t  s_OL_I_ID[15];
  int32_t  s_OL_SUPPLY_W_ID[15];
  int32_t  s_OL_QUANTITY[15];
  int32_t  ol_del_len[15];
  int16_t  s_transtatus;
  int32_t  retry;
  char     s_C_FIRST[17];
  char     s_C_MIDDLE[3];
  char     s_C_LAST[17];
};

struct in_delivery_struct {
  double  s_O_DELIVERY_D_time;
  int32_t  s_W_ID;
  int32_t  s_O_CARRIER_ID;
  int16_t  plsqlflag;
  OCIDate cr_date;
};

struct out_delivery_struct {
  int32_t  s_O_ID[10];
  int32_t  terror;
  int16_t  s_transtatus;
  int32_t  retry;
};

#endif
```

oratpcc.h

```
#ifndef ORATPCC_H
#define ORATPCC_H
    typedef struct {
        long int sec;
        long int usec;
    } time_type;
    typedef struct  {
        OCIDate cr_date;
        int  dtype;
        int  returncode;
        long int   sql_code;
        long int   isam_code;
        long int   num_rms;

        short int  stats;       /*  For instrument only */
        time_type  start_time;  /*  For Debug Purposes only  */
        time_type  end_time;    /*  For Debug Purposes only  */
    } data_header;

    struct stoinstruct {
        int w_id;
        int d_id;
#ifdef USE_IEEE_NUMBER
        float threshold;
#else
        int threshold;
#endif

    };

    struct stooutstruct {
        int terror;
        int low_stock;
        int retry;
    };

    struct stostruct {
        data_header header;
        struct stoinstruct stoin;
        struct stooutstruct stoout;
        int out_s_transtatus;
    };

    struct newinstruct {
        int w_id;
        int d_id;
        int c_id;
        int ol_i_id[15];
        int ol_supply_w_id[15];
#ifdef USE_IEEE_NUMBER
        float ol_quantity[15];
#else
        int ol_quantity[15];
#endif
    };

    struct newoutstruct {
        int terror;
        int o_id;
        int o_ol_cnt;
        char c_last[17];
        char c_credit[3];
        float c_discount;
        float w_tax;
        float d_tax;
        char o_entry_d[20];
        int datelen;
#ifdef USE_IEEE_NUMBER
        float i_price[15];
        float ol_amount[15];
        float s_quantity[15];
#else
        int i_price[15];
        int ol_amount[15];
        int s_quantity[15];
#endif
        float total_amount;
        char i_name[15][25];
```

```
        char brand_generic[15];
        int status;
        int retry;
        int o_all_local;
    };

    struct newstruct {
        data_header header;
        struct newinstruct newin;
        struct newoutstruct newout;
        int out_s_transtatus;
        int items_valid;
    };

    struct payinstruct {
        int w_id;
        int d_id;
        int c_w_id;
        int c_d_id;
        int c_id;
        int bylastname;
#ifdef USE_IEEE_NUMBER
        float h_amount;
#else
        int h_amount;
#endif
        char c_last[17];
    };

    struct payoutstruct {
        int terror;
        int hlen;
        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];
        int  c_id;
        char c_first[17];
        char c_middle[3];
        char c_last[17];
        char c_street_1[21];
        char c_street_2[21];
        char c_city[21];
        char c_state[3];
        char c_zip[10];
        char c_phone[17];
        OCIDate c_since;
        char c_since_d[11];
        int since_len;
        char c_credit[3];
        double c_credit_lim;
        float  c_discount;
        double c_balance;
        char c_data[201];
        char h_date[20];
        int retry;
    };

    struct paystruct {
        data_header header;
        struct payinstruct payin;
        struct payoutstruct payout;
        int out_s_transtatus;
    };

    struct ordinstruct {
        int w_id;
        int d_id;
        int c_id;
        int bylastname;
        char c_last[17];
    };

    struct ordoutstruct {
        int terror;
        int c_id;
        char c_last[17];
        char c_first[17];
        char c_middle[3];
        double c_balance;
        int o_id;
        OCIDate o_entry_d_base;
        OCIDate ol_d_base[15];
        char o_entry_d[20];
        int o_carrier_id;
        int o_ol_cnt;
        int ol_supply_w_id[15];
        int ol_i_id[15];
#ifdef USE_IEEE_NUMBER
        float ol_quantity[15];
        float ol_amount[15];
#else
        int ol_quantity[15];
        int ol_amount[15];
#endif
        char ol_delivery_d[15][11];
        int ol_del_len[15];
        int retry;
    };

    struct ordstruct {
        data_header header;
        struct ordinstruct ordin;
        struct ordoutstruct ordout;
        int out_s_transtatus;
    };

struct delinstruct {
   int w_id;
   int o_carrier_id;
   double qtime;
   int in_timing_int;
   int plsqlflag;
};

struct deloutstruct {
   int del_o_id[10];
   int terror;
   int retry;
};

struct delstruct {
    data_header header;
   struct delinstruct delin;
   struct deloutstruct delout;
   OCIDate cr_date;
   int out_s_transtatus;
};
#endif /* ORATPCC_H */
```

plora.h

```
/*
 * $Header: plora.h for Encina
 */
/*+============================================================+
 |                Copyright (c) 2001 Oracle Corporation       |
 |                   All Rights Reserved                      |
 +============================================================+
 | FILENAME
 |    plora.h
 | DESCRIPTION
 |    Include file for TPC-C benchmark programs.
 +============================================================*/


#ifndef TPCC_PLORA_H
#define TPCC_PLORA_H

#define serverDebug 1
//extern void logprintf(char *format, ...);
//#define DEBUGP(list) if (serverDebug) logprintf list
#define TIME_PREFIX_LEN 50
//extern FILE *server_logtrans;

#define NEWO_TRANS (1)
#define PAYMENT_TRANS (2)
#define ORDER_STAT_TRANS (3)
#define DELIVERY_TRANS (4)
#define STOCK_TRANS (5)
#define MAX_TRAN_TYPE (5)

/* Oracle handles and rest of thread specific vars(thread slot data ) */

struct ora_cn_data_t {
  OCIEnv *tpcenv;
  OCIServer *tpcsrv;
  dvoid *errhp;
  OCISvcCtx *tpcsvc;
  OCISession *tpcusr;
  OCIStmt *curi;
  dvoid *xmem;

  nord_wrapper globals;
  ords_wrapper ordP;
  stok_wrapper stoP;
  paym_wrapper payP;
  dlvy_wrapper delP;

  int tid;
  FILE* server_logtrans;

  void *nctx;
  void *pctx;
  void *octx;
  void *sctx;
  void *dctx;
  void *pldctx;
  void *actx;  /* for #ifdef DMLRETDEL */
  void *cbctx; /* for orderstatus */
};

struct thread_info
{
    ora_cn_data_t           *curP;
    FILE                *server_logtrans;
    int                 tid;
    int                 server_null_test;
    char                oracle_home[255];
    char                dbName[20];
};

typedef struct ora_cn_data_t ora_cn_data_t;
extern int tkvcninit (ora_cn_data_t *ora_SlotDataP);
extern int tkvcpinit (ora_cn_data_t *ora_SlotDataP);
```

```
extern int tkvcoinit (ora_cn_data_t *ora_SlotDataP);
extern int tkvcdinit (ora_cn_data_t *ora_SlotDataP);
extern int tkvcsinit (ora_cn_data_t *ora_SlotDataP);

extern int tkvcn (ora_cn_data_t *ora_SlotDataP);
extern int tkvcp (ora_cn_data_t *ora_SlotDataP);
extern int tkvco (ora_cn_data_t *ora_SlotDataP);
extern int tkvcd (ora_cn_data_t *ora_SlotDataP);
extern int tkvcs (ora_cn_data_t *ora_SlotDataP);

extern void tkvcndone (ora_cn_data_t *ora_SlotDataP);
extern void tkvcpdone (ora_cn_data_t *ora_SlotDataP);
extern void tkvcodone (ora_cn_data_t *ora_SlotDataP);
extern void tkvcddone (ora_cn_data_t *ora_SlotDataP);
extern void tkvcsdone (ora_cn_data_t *ora_SlotDataP);

#endif /* TPCC_PLORA_H */
```

stdafx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once


#define WIN32_LEAN_AND_MEAN        // Exclude rarely-used stuff from Windows headers
// Windows Header Files:
#include <windows.h>

// TODO: reference additional headers your program requires here
```

tpccflags.h

```
#define PLSQLNO
#define DMLRETDEL
```

tpcc_info.h

```
/*
 * $Header: tpcc_info.h 7030100.1 95/07/19 15:11:37 plai Generic<base> $ Copyr (c) 1995
Oracle
 */
/*==============================================================+
 |       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA     |
 |                OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                   All Rights Reserved                         |
 +==============================================================+
 | FILENAME
 |    tpcc_info.h
 | DESCRIPTION
 |    Include file for TPC-C benchmark programs.
 +==============================================================*/

#ifndef TPCC_INFO_H
#define TPCC_INFO_H

/* this set is duplicated in c_Defs.h, c_Defs.h is used for batch driver */
#define MENTXN    0         /* menu txn */
#define NEWTXN    1         /* new order transaction */
#define PAYTXN    2         /* payment transaction */
#define ORDTXN    3         /* order status transaction */
#define DELTXN    4         /* delivery transaction */
#define STOTXN    5         /* stock level transaction */
#define ALLTXN    6         /* for processing all txns */
#define ALLTXNNODEL 7       /* for processing all txns except delivery */
#endif
```

tpccOracleGlue.h

```
// The following ifdef block is the standard way of creating macros which make exporting
// from a DLL simpler. All files within this DLL are compiled with the
TPCCORACLEGLUE_EXPORTS
// symbol defined on the command line. this symbol should not be defined on any project
// that uses this DLL. This way any other project whose source files include this file see
// TPCCORACLEGLUE_API functions as being imported from a DLL, whereas this DLL
sees symbols
// defined with this macro as being exported.
#ifdef TPCCORACLEGLUE_EXPORTS
#define TPCCORACLEGLUE_API __declspec(dllexport)
#else
#define TPCCORACLEGLUE_API __declspec(dllimport)
#endif

#include "../tpccisapi/tpcc.h"
#include "ora10_mt/ora_tpcc.h"
#include "ora10_mt/plora.h"

////////////////////////////////////////////////////////////////////
// Error/Debug log file defines
////////////////////////////////////////////////////////////////////
ofstream debugStream;
ofstream errorStream;

////////////////////////////////////////////////////////////////////
// Registery Values
////////////////////////////////////////////////////////////////////
#define     DB_USER_NAME                "dbUserName"
#define     DB_USER_PASSWORD            "dbPassword"
#define     DB_NAME                     "dbName"

char userName[16] = {NULL};
char userPassword[16] = {NULL};

HKEY   registeryKey;
DWORD   regType;
char  value[MAX_STRING_LEN];
DWORD   regValueSize = MAX_STRING_LEN;

////////////////////////////////////////////////////////////////////
// Oracle Glue Function Prototypes
////////////////////////////////////////////////////////////////////
extern "C" TPCCORACLEGLUE_API int connect_db(char *dbName,void **ctx);
extern "C" TPCCORACLEGLUE_API int getContext(void **ctx);
extern "C" TPCCORACLEGLUE_API int detachContext(void *ctx);
extern "C" TPCCORACLEGLUE_API int attachContext(void *ctx);
extern "C" TPCCORACLEGLUE_API int disconnect_db(void *ctx);

extern "C" TPCCORACLEGLUE_API int do_nord(nord_wrapper *nord,void *ctx);
extern "C" TPCCORACLEGLUE_API int do_pymt(paym_wrapper *pymt,void *ctx);
extern "C" TPCCORACLEGLUE_API int do_ords(ords_wrapper *ords,void *ctx);
extern "C" TPCCORACLEGLUE_API int do_dlvy(dlvy_wrapper *dlvy,void *ctx);
extern "C" TPCCORACLEGLUE_API int do_stok(stok_wrapper *stok,void *ctx);

////////////////////////////////////////////////////////////////////
// Function Prototypes
////////////////////////////////////////////////////////////////////
extern ora_cn_data_t *get_db_ready(ora_cn_data_t *ctx, char *uid, char *pwd, char
*dbName);
extern void clean_connection(FILE* server_logtrans,void *ptr);

extern int TPCnew (void *cnP);
extern int TPCpay (void *cnP);
extern int TPCord (void *cnP);
extern int TPCsto (void *cnP);
extern int TPCdel (void *cnP);

extern CRITICAL_SECTION debugMutex;

#define TPCC_SUCCESS 0

// This class is exported from the tpccOracleGlue.dll
class TPCCORACLEGLUE_API CtpccOracleGlue {
public:
    CtpccOracleGlue(void);

    // TODO: add your methods here.
};
```

tpccpl.h

```
/*
 * $Header: tpccpl.h 7030100.1 96/04/02 18:03:35 plai Generic<base> $ Copyr (c) 1994
Oracle
 */
/*==============================================================+
 |       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 "tpcc.h"

#ifdef TUX
#define DELRT 5.0
#else
#define DELRT 80.0
#endif

extern int plnewinit ();
extern int plpayinit ();
extern int plordinit ();
extern int pldelinit ();
extern int plstoinit ();

extern int plnew ();
extern int plpay ();
extern int plord ();
extern int pldel ();
extern int plsto ();

extern void plnewdone ();
extern void plpaydone ();
extern void plorddone ();
extern void pldeldone ();
extern void plstodone ();

extern errrpt ();
extern void logerr();
extern int ocierror(FILE *server_logtrans,char *fname, int lineno,OCIError *errhp, sword
status);


extern int sqlfile(char *fname, text *linebuf);

extern void cvtdmy ( unsigned char *orady, char *outdate);
extern void cvtdmyhms (unsigned char *orady, char *outdate);

extern FILE *lfp;
```

```
extern FILE *fopen ();
extern int proc_no;
extern int doid[];
extern int execstatus;
extern int errcode;

#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 */
#define SNAPSHOT_TOO_OLD  1555  /* ORA-01555: snapshot too old */


#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 OCIERROR(errp,function)\
   ocierror(FILE *server_logtrans,__FILE__,__LINE__,(void *)(errp),(function));

#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      ocierror(FILE *server_logtrans,__FILE__,__LINE__, (errp), \
       OCIBindByName((stmp), &(bndp), (errp), \
          (text *)(sqlvar), strlen((sqlvar)),\
          (progv), (progvl), (ftype),0,0,0,0,0,OCI_DEFAULT));

#define OCIBNDRA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
       OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
                (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));
#define OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,cbf_nodata,cbf_data) \
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
       OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar), \
            strlen((sqlvar)),0,(progvl),(ftype), \
            indp,0,0,0,0,OCI_DATA_AT_EXEC)); \
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
       OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)));


#define OCIBNDR(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
      OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
       ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
      OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
                (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT));

#define OCIBNDRAA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode,ms,cu) \
      ocierror(FILE *server_logtrans,__FILE__,__LINE__, (errp), \
      OCIHandleAlloc((stmp),&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
       ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp),\
      OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
       (progv),(progvl),(ftype),(indp),(alen),(arcode),(ms),(cu),OCI_DEFAULT));


#define OCIDEFINE(stmp,dfnp,errp,pos,progv,progvl,ftype)\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),(ftype),\
            0,0,0,OCI_DEFAULT);


#define OCIDEF(stmp,dfnp,errp,pos,progv,progvl,ftype) \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                  (dvoid**)0);\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),\
             (ftype),NULL,NULL,NULL,OCI_DEFAULT); \

#define OCIDFNRA(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,alen,arcode) \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                  (dvoid**)0);\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
                  (progvl),(ftype),(indp),(alen),\
                  (arcode),OCI_DEFAULT);\
/*
      OCIDefineArrayOfStruct((dfnp),(errp),(progvl),\
                        sizeof((indp)[0]),\
                        sizeof((alen)[0]),\
                        sizeof((arcode)[0]));
*/
/*
#define OCIDFNRA(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,alen,arcode) \
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
        OCIHandleAlloc(tpcenv,&(dfnp),OCI_HTYPE_DEFINE,0,\
                  (dvoid**)0));\
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
        OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
                  (progvl),(ftype),(indp),(alen),\
                  (arcode),OCI_DEFAULT));\
      ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp),\
        OCIDefineArrayOfStruct((dfnp),(errp),(progvl),\
                        sizeof((indp)[0]),\
                        sizeof((alen)[0]),\
                        sizeof((arcode)[0])));
*/

#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);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);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);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);return(-1);}\
   else\
      DISCARD 0

#define OEXFET(lda,cursor,nrows,cancel,exact)\
   if (oexfet((cursor),(nrows),(cancel),(exact)))\
      {if ((cursor)->rc == 1403) \
     {i=errrpt(lda,cursor); orol(lda); return(-1);} \
    else if (errrpt(lda,cursor)==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);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);return(-1);}\
   else\
      DISCARD 0

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

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


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


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

#endif
```

9.3. Server Stored Procedures

tkvcpdel.sql

```
declare
  TYPE numarray IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
  TYPE numlist is varray (10) of number;
  dist numarray;
  amt numarray ;
  cnt pls_integer;

  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
    FORALL d IN 1..10
      DELETE FROM nord N
       WHERE no_d_id = inittpcc.dist(d)
         AND no_w_id = :w_id
         AND no_o_id = (select min (no_o_id)
                        from nord
                        where no_d_id = N.no_d_id
                              and no_w_id = N.no_w_id)
      RETURNING no_d_id, no_o_id BULK COLLECT INTO :d_id, :order_id;

    :ordcnt := SQL%ROWCOUNT;

    FORALL o in 1.. :ordcnt
      UPDATE ordr SET o_carrier_id = :carrier_id
      WHERE o_id = :order_id (o)
        AND o_d_id = :d_id(o)
        AND o_w_id = :w_id
      RETURNING o_c_id BULK COLLECT INTO :o_c_id;

    FORALL o in 1.. :ordcnt
      UPDATE ordl SET ol_delivery_d = :now
      WHERE ol_w_id = :w_id
        AND ol_d_id = :d_id(o)
        AND ol_o_id = :order_id(o)
      RETURNING sum(ol_amount) BULK COLLECT INTO  :sums;

    FORALL c IN 1.. :ordcnt
      UPDATE cust
        SET c_balance = c_balance + :sums(c),
                    c_delivery_cnt = c_delivery_cnt + 1
       WHERE c_w_id = :w_id
         AND c_d_id = :d_id(c)
         AND c_id = :o_c_id(c);
    COMMIT;
    EXIT;
    EXCEPTION
      WHEN not_serializable OR deadlock OR snapshot_too_old
      THEN
        ROLLBACK;
        :retry := :retry + 1;
    END;

  END LOOP; -- for retry
END;
```

tkvcpnew.sql

```
-- New Order Anonymous block

  DECLARE
     idx                  PLS_INTEGER;
     dummy_local          PLS_INTEGER;
     cache_ol_cnt         PLS_INTEGER;
     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);

  PROCEDURE u1 IS
  BEGIN
     FORALL idx IN 1 .. cache_ol_cnt
       UPDATE  stock_item
       SET s_order_cnt = s_order_cnt + 1,
       s_ytd = s_ytd + :ol_quantity(idx),
       s_remote_cnt = s_remote_cnt + :s_remote(idx),
       s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                          THEN s_quantity +91
                          ELSE s_quantity
                     END) - :ol_quantity(idx)
       WHERE i_id = :ol_i_id(idx)
       AND s_w_id = :ol_supply_w_id(idx)
       RETURNING i_price, i_name, s_quantity, s_dist_01,
                 i_price*:ol_quantity(idx),
        CASE WHEN i_data NOT LIKE '%ORIGINAL%'
             THEN 'G'
             ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                        THEN 'G'
                        ELSE 'B'
                        END)
        END
      BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                        :ol_amount,:brand_generic;
  END u1;

  PROCEDURE u2 IS
  BEGIN
     FORALL idx IN 1 .. cache_ol_cnt
       UPDATE  stock_item
       SET s_order_cnt = s_order_cnt + 1,
       s_ytd = s_ytd + :ol_quantity(idx),
       s_remote_cnt = s_remote_cnt + :s_remote(idx),
       s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                          THEN s_quantity +91
                          ELSE s_quantity
                     END) - :ol_quantity(idx)
       WHERE i_id = :ol_i_id(idx)
       AND s_w_id = :ol_supply_w_id(idx)
       RETURNING i_price, i_name, s_quantity, s_dist_02,
                 i_price*:ol_quantity(idx),
        CASE WHEN i_data NOT LIKE '%ORIGINAL%'
             THEN 'G'
             ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                        THEN 'G'
                        ELSE 'B'
                        END)
        END
      BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                        :ol_amount,:brand_generic;
  END u2;

  PROCEDURE u3 IS
  BEGIN
     FORALL idx IN 1 .. cache_ol_cnt
       UPDATE  stock_item
       SET s_order_cnt = s_order_cnt + 1,
       s_ytd = s_ytd + :ol_quantity(idx),
       s_remote_cnt = s_remote_cnt + :s_remote(idx),
       s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                          THEN s_quantity +91
                          ELSE s_quantity
                     END) - :ol_quantity(idx)
       WHERE i_id = :ol_i_id(idx)
       AND s_w_id = :ol_supply_w_id(idx)
       RETURNING i_price, i_name, s_quantity, s_dist_03,
                 i_price*:ol_quantity(idx),
        CASE WHEN i_data NOT LIKE '%ORIGINAL%'
             THEN 'G'
             ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                        THEN 'G'
                        ELSE 'B'
                        END)
        END
      BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                        :ol_amount,:brand_generic;
  END u3;

  PROCEDURE u4 IS
  BEGIN
     FORALL idx IN 1 .. cache_ol_cnt
       UPDATE  stock_item
       SET s_order_cnt = s_order_cnt + 1,
       s_ytd = s_ytd + :ol_quantity(idx),
       s_remote_cnt = s_remote_cnt + :s_remote(idx),
       s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                          THEN s_quantity +91
                          ELSE s_quantity
                     END) - :ol_quantity(idx)
       WHERE i_id = :ol_i_id(idx)
       AND s_w_id = :ol_supply_w_id(idx)
       RETURNING i_price, i_name, s_quantity, s_dist_04,
                 i_price*:ol_quantity(idx),
        CASE WHEN i_data NOT LIKE '%ORIGINAL%'
             THEN 'G'
             ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                        THEN 'G'
                        ELSE 'B'
                        END)
        END
      BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                        :ol_amount,:brand_generic;
  END u4;

  PROCEDURE u5 IS
  BEGIN
     FORALL idx IN 1 .. cache_ol_cnt
       UPDATE  stock_item
       SET s_order_cnt = s_order_cnt + 1,
       s_ytd = s_ytd + :ol_quantity(idx),
       s_remote_cnt = s_remote_cnt + :s_remote(idx),
       s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                          THEN s_quantity +91
                          ELSE s_quantity
                     END) - :ol_quantity(idx)
       WHERE i_id = :ol_i_id(idx)
       AND s_w_id = :ol_supply_w_id(idx)
       RETURNING i_price, i_name, s_quantity, s_dist_05,
                 i_price*:ol_quantity(idx),
        CASE WHEN i_data NOT LIKE '%ORIGINAL%'
             THEN 'G'
             ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                        THEN 'G'
                        ELSE 'B'
                        END)
        END
      BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                        :ol_amount,:brand_generic;
  END u5;

  PROCEDURE u6 IS
  BEGIN
     FORALL idx IN 1 .. cache_ol_cnt
```

```
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                      THEN s_quantity +91
                      ELSE s_quantity
                 END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_06,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                 :ol_amount,:brand_generic;
END u6;

PROCEDURE u7 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                      THEN s_quantity +91
                      ELSE s_quantity
                 END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_07,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                 :ol_amount,:brand_generic;
END u7;

PROCEDURE u8 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                      THEN s_quantity +91
                      ELSE s_quantity
                 END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_08,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                 :ol_amount,:brand_generic;
END u8;

PROCEDURE u9 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                      THEN s_quantity +91
                      ELSE s_quantity
                 END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_09,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                 :ol_amount,:brand_generic;
END u9;

PROCEDURE u10 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                      THEN s_quantity +91
                      ELSE s_quantity
                 END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_10,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                 :ol_amount,:brand_generic;
END u10;

PROCEDURE fix_items IS
  rows_lost                PLS_INTEGER;
  max_index                PLS_INTEGER;
  temp_index               PLS_INTEGER;
BEGIN
  idx := 1;
  rows_lost := 0;
  max_index := dummy_local;

  WHILE (max_index != cache_ol_cnt) LOOP

   WHILE (idx <= sql%rowcount AND
            sql%bulk_rowcount(idx + rows_lost) = 1)
   LOOP
     idx := idx + 1;
   END LOOP;

   temp_index := max_index;
   WHILE (temp_index >= idx + rows_lost) LOOP
     :ol_amount(temp_index + 1)      :=  :ol_amount(temp_index);
     :i_price(temp_index + 1)        :=  :i_price(temp_index);
     :i_name(temp_index + 1)         :=  :i_name(temp_index);
     :s_quantity(temp_index + 1)     :=  :s_quantity(temp_index);
     inittpcc.s_dist(temp_index + 1) :=  inittpcc.s_dist(temp_index);
     :brand_generic(temp_index + 1)  :=  :brand_generic(temp_index);
     temp_index := temp_index - 1;
   END LOOP;

   IF (idx + rows_lost <= cache_ol_cnt) THEN
     :i_price(idx + rows_lost)        :=   0;
     :i_name(idx + rows_lost)         :=   'NO ITEM';
     :s_quantity(idx + rows_lost)     :=   0;
     inittpcc.s_dist(idx + rows_lost) := NULL;
     :brand_generic(idx + rows_lost)  := ' ';
     :ol_amount(idx + rows_lost)      := 0;
     rows_lost := rows_lost + 1;
     max_index := max_index + 1;
   END IF;

 END LOOP;
END fix_items;

BEGIN
   LOOP BEGIN
      cache_ol_cnt := :o_ol_cnt;

      UPDATE dist SET d_next_o_id = d_next_o_id + 1
       WHERE d_id = :d_id AND  d_w_id = :w_id
       RETURNING d_tax, d_next_o_id-1
       INTO :d_tax, :o_id;

      SELECT c_discount, c_last, c_credit
       INTO :c_discount, :c_last, :c_credit
       FROM cust
       WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id;

     SELECT w_tax
      INTO :w_tax
      FROM ware
      WHERE w_id = :w_id;

      INSERT INTO nord (no_o_id, no_d_id, no_w_id)
       VALUES (:o_id, :d_id, :w_id);

      INSERT INTO ordr  (o_id,o_d_id, o_w_id, o_c_id, o_entry_d,
                         o_carrier_id, o_ol_cnt, o_all_local)
       VALUES (:o_id, :d_id, :w_id, :c_id,
              :cr_date, 11, :o_ol_cnt, :o_all_local);


      dummy_local :=  :d_id;

      IF (dummy_local < 6) THEN
        IF (dummy_local < 3) THEN
          IF (dummy_local = 1) THEN
            u1;
          ELSE
            u2;
          END IF;
        ELSE
          IF (dummy_local = 3) THEN
            u3;
          ELSIF (dummy_local = 4) then
```

```
            u4;
          ELSE
            u5;
          END IF;
        END IF;
      ELSE
        IF (dummy_local < 8) THEN
          IF (dummy_local = 6) THEN
            u6;
          ELSE
            u7;
          END IF;
        ELSE
          IF (dummy_local = 8) THEN
            u8;
          ELSIF (dummy_local = 9) then
            u9;
          ELSE
            u10;
          END IF;
        END IF;
      END IF;

      dummy_local := sql%rowcount;

      IF (dummy_local != cache_ol_cnt )  THEN fix_items; END IF;

      FORALL idx IN 1..dummy_local
       INSERT INTO ordl
         (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_delivery_d, ol_i_id,
              ol_supply_w_id, ol_quantity,ol_amount,ol_dist_info)
       VALUES (:o_id, :d_id, :w_id, inittpcc.idx1arr(idx), inittpcc.nulldate,
              :ol_i_id(idx), :ol_supply_w_id(idx),
              :ol_quantity(idx), :ol_amount(idx), inittpcc.s_dist(idx));

      IF (dummy_local != :o_ol_cnt) THEN
        :o_ol_cnt := dummy_local;
        ROLLBACK;
      END IF;

    EXIT;

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

paynz.sql

```
DECLARE /* paynz */
      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
          UPDATE ware
            SET w_ytd = w_ytd + :h_amount
           WHERE w_id = :w_id
        RETURNING w_name, w_street_1, w_street_2, w_city, w_state, w_zip
            INTO inittpcc.ware_name, :w_street_1, :w_street_2, :w_city,
                :w_state, :w_zip;

          UPDATE  cust
            SET  c_balance = c_balance - :h_amount,
                 c_ytd_payment = c_ytd_payment + :h_amount,
                 c_payment_cnt = c_payment_cnt+1
           WHERE    c_id = :c_id AND c_d_id = :c_d_id AND
                 c_w_id = :c_w_id
         RETURNING rowid, c_first, c_middle, c_last, 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
              INTO inittpcc.cust_rowid,:c_first, :c_middle, :c_last, :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;
        IF SQL%NOTFOUND THEN
          raise NO_DATA_FOUND;
        END IF;


        IF :c_credit = 'BC' THEN
          UPDATE cust
          SET c_data = substr ((to_char (:c_id) || ' ' ||
                               to_char (:c_d_id) || ' ' ||
                               to_char (:c_w_id) || ' ' ||
                               to_char (:d_id) || ' ' ||
                               to_char (:w_id) || ' ' ||
                               to_char (:h_amount/100, '9999.99') || ' ')
                               || c_data, 1, 500)
            WHERE rowid = inittpcc.cust_rowid
          RETURNING substr(c_data,1, 200)
              INTO :c_data;

        END IF;

          UPDATE dist
             SET d_ytd = d_ytd + :h_amount
           WHERE d_id = :d_id
             AND d_w_id = :w_id
         RETURNING d_name, d_street_1, d_street_2, d_city,d_state, d_zip
              INTO inittpcc.dist_name,:d_street_1,:d_street_2,:d_city,:d_state,
                 :d_zip;
         IF SQL%NOTFOUND THEN
          raise NO_DATA_FOUND;
         END IF;


        INSERT INTO hist  (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                           h_amount, h_date, h_data)
        VALUES
          (:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
           :cr_date, inittpcc.ware_name || '    ' || inittpcc.dist_name);
        EXIT;

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

      END LOOP;
    END;
```

payz.sql

```
DECLARE /* payz */
      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
    UPDATE ware
      SET w_ytd = w_ytd+:h_amount
      WHERE w_id = :w_id
      RETURNING w_name,
            w_street_1, w_street_2, w_city, w_state, w_zip
       INTO inittpcc.ware_name,
           :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


    SELECT rowid
    BULK COLLECT INTO inittpcc.row_id
    FROM cust
    WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last = :c_last
    ORDER BY c_last, c_d_id, c_w_id, c_first;

  inittpcc.c_num := sql%rowcount;
  inittpcc.cust_rowid := inittpcc.row_id((inittpcc.c_num) / 2);

    UPDATE cust
     SET c_balance = c_balance - :h_amount,
         c_ytd_payment = c_ytd_payment+ :h_amount,
         c_payment_cnt = c_payment_cnt+1
   WHERE rowid = inittpcc.cust_rowid
   RETURNING
          c_id, c_first, c_middle, c_last, 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
       INTO :c_id, :c_first, :c_middle, :c_last,
           :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 := ' ';
    IF :c_credit = 'BC' THEN
      UPDATE cust
        SET c_data = substr ((to_char (:c_id) || ' ' ||
                          to_char (:c_d_id) || ' ' ||
                          to_char (:c_w_id) || ' ' ||
                          to_char (:d_id) || ' ' ||
                          to_char (:w_id) || ' ' ||
                          to_char (:h_amount/100, '9999.99') || ' ')
                          || c_data, 1, 500)
        WHERE rowid = inittpcc.cust_rowid
        RETURNING substr(c_data,1, 200)
        INTO :c_data;

    END IF;

    UPDATE dist
      SET d_ytd = d_ytd+:h_amount
      WHERE d_id = :d_id
        AND d_w_id = :w_id
      RETURNING  d_name, d_street_1, d_street_2, d_city,
       d_state, d_zip
      INTO inittpcc.dist_name, :d_street_1, :d_street_2, :d_city,
          :d_state, :d_zip;

      IF  SQL%NOTFOUND
    THEN
    raise NO_DATA_FOUND;
      END IF;

    INSERT INTO hist (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                      h_amount, h_date, h_data)
       VALUES (:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
       :cr_date, inittpcc.ware_name || '    ' || inittpcc.dist_name);

    EXIT;
```

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

  END LOOP;
END;
```

10 Appendix B: Tunable Parameters

10.1. Database Parameters

p_run.ora

```
compatible                  = 10.2.0.1.0
timed_statistics             = FALSE
query_rewrite_enabled          = FALSE
db_name                     = tpcc
control_files               =
(/home/oracle/tpcc_20k/dev/control_001,/home/oracle/tpcc_20k/dev/control_002)
recovery_parallelism          = 32
dml_locks                   = 700
log_buffer                  = 10485760
parallel_max_servers          = 16
db_files                    = 400
db_cache_size                = 19000M
db_8k_cache_size              = 456M
db_16k_cache_size             = 12750M
db_keep_cache_size            = 78870M
db_recycle_cache_size          = 8000M
shared_pool_size              = 3600M
processes                   = 556
sessions                    = 546
transactions                = 556
cursor_space_for_time         = TRUE
db_block_size               = 2048
undo_management               = auto
UNDO_TABLESPACE                 = undo_1
trace_enabled               = FALSE
db_block_checksum             = FALSE
trace_enabled               = FALSE
statistics_level            = basic
plsql_optimize_level          = 2
pga_aggregate_target          = 0
undo_retention              = 2
fast_start_mttr_target        = 0
db_writer_processes           = 4
log_checkpoint_interval       = 0
log_checkpoints_to_alert      = TRUE
log_checkpoint_timeout        = 1770 java_pool_size              = 0
remote_login_passwordfile     = shared
disk_asynch_io              = TRUE
filesystemio_options          = asynch
db_block_checking             = FALSE
lock_sga                    = TRUE
replication_dependency_tracking = FALSE
db_file_multiblock_read_count  = 1
max_dump_file_size            = 5M
aq_tm_processes               = 0
utl_file_dir=*
```

10.2. Transaction Monitor Parameters

inetInfo_registry.reg

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo\Parameters]
"ListenBackLog"=dword:00000040
"DispatchEntries"=hex(7):53,00,47,00,54,00,55 ,00,53,00,56,00,43,00,00,00,00,00
"PoolThreadLimit"=dword:00000320
"MaxPoolThreads"=dword:000000c8
"MaxConcurrency"=dword:00000320
```

tcpip_parameters_registery.reg

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters]
"NV Hostname"="client1"
"DataBasePath"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,\
  00,74,00,25,00,5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,\
  64,00,72,00,69,00,76,00,65,00,72,00,73,00,5c,00,65,00,74,00,63,00,00,00
"NameServer"=""
"ForwardBroadcasts"=dword:00000000
"IPEnableRouter"=dword:00000000
"Domain"=""
"Hostname"="client1"
"SearchList"=""
"UseDomainNameDevolution"=dword:00000001
"EnableICMPRedirect"=dword:00000001
"DeadGWDetectDefault"=dword:00000001
"DontAddDefaultGatewayDefault"=dword:00000000
"EnableSecurityFilters"=dword:00000000
"SynAttackProtect"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Adapte
rs]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Adapte
rs\NdisWanIp]
"LLInterface"="WANARP"

"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
  6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
  00,61,00,63,00,65,00,73,00,5c,00,7b,00,42,00,32,00,44,00,38,00,46,00,42,00,\
  45,00,36,00,2d,00,46,00,36,00,35,00,42,00,2d,00,34,00,36,00,41,00,35,00,2d,\
  00,42,00,31,00,32,00,35,00,2d,00,31,00,35,00,43,00,33,00,43,00,38,00,45,00,\
  43,00,32,00,37,00,46,00,39,00,7d,00,00,00,54,00,63,00,70,00,69,00,70,00,5c,\
  00,50,00,61,00,72,00,61,00,6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,\
  6e,00,74,00,65,00,72,00,66,00,61,00,63,00,65,00,73,00,5c,00,7b,00,45,00,32,\
  00,41,00,37,00,35,00,36,00,46,00,46,00,2d,00,37,00,34,00,31,00,37,00,2d,00,\
  34,00,43,00,34,00,41,00,2d,00,41,00,31,00,39,00,43,00,2d,00,34,00,42,00,41,\
  00,37,00,38,00,33,00,32,00,38,00,46,00,46,00,35,00,30,00,7d,00,00,00,00,00
"NumInterfaces"=dword:00000002
"IpInterfaces"=hex:e6,fb,d8,b2,5b,f6,a5,46,b1,25,15,c3,c8,ec,27,f9,ff,56,a7,e2,\
  17,74,4a,4c,a1,9c,4b,a7,83,28,ff,50

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Adapte
rs\{52C558C5-704A-4714-BAB8-A60B8D493D4E}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
  6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
  00,61,00,63,00,65,00,73,00,5c,00,7b,00,35,00,32,00,43,00,35,00,35,00,38,00,\
  43,00,35,00,2d,00,37,00,30,00,34,00,41,00,2d,00,34,00,37,00,31,00,34,00,2d,\
  00,42,00,41,00,42,00,38,00,2d,00,41,00,36,00,30,00,42,00,38,00,44,00,34,00,\
  39,00,33,00,44,00,34,00,45,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Adapte
rs\{9C21BC6A-969E-4AF3-A8EB-E6E8523FBD63}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
  6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
  00,61,00,63,00,65,00,73,00,5c,00,7b,00,39,00,43,00,32,00,31,00,42,00,43,00,\
  36,00,41,00,2d,00,39,00,36,00,39,00,45,00,2d,00,34,00,41,00,46,00,33 ,00,2d,\
  00,41,00,38,00,45,00,42,00,2d,00,45,00,36,00,45,00,38,00,35,00,32,00,33,00,\
  46,00,42,00,44,00,36,00,33,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\DNSR
egisteredAdapters]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfa
ces]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfa
ces\{52C558C5-704A-4714-BAB8-A60B8D493D4E}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):31,00,39,00,32,00,2e,00,31,00,36,00,38,00,2e,00,31,00,2e,00,31,00,0
0,00,00,00
  "SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,\
  00,35,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00,\
  32,00,00,00,00,00
"DhcpClassIdBin"=hex:
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000e10
"LeaseObtainedTime"=dword:4601a06a
"T1"=dword:4601b172
"T2"=dword:4601b6b8
"LeaseTerminatesTime"=dword:4601b87a
"IPAutoconfigurationAddress"="0.0.0.0"
"IPAutoconfigurationMask"="255.255.0.0"
"IPAutoconfigurationSeed"=dword:00000000
"AddressType"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfa
ces\{9C21BC6A-969E-4AF3-A8EB-E6E8523FBD63}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,32,00,2e,00,31,00,2e,00,32,00,00,00,00,00
"SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,\
  00,35,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00,\
  33,00,00,00,00,00
"DhcpClassIdBin"=hex:
"LeaseTerminatesTime"=dword:45884a89
"IPAutoconfigurationAddress"="0.0.0.0"
"IPAutoconfigurationMask"="255.255.0.0"
"IPAutoconfigurationSeed"=dword:00000000
"AddressType"=dword:00000000
"TcpWindowSize"=dword:00040000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Persist
entRoutes]
```

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Winsock]
"UseDelayedAcceptance"=dword:00000000
"HelperDllName"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,\
  6f,00,74,00,25,00,5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,\
  00,77,00,73,00,68,00,74,00,63,00,70,00,69,00,70,00,2e,00,64,00,6c,00,6c,00,\
  00,00
"MaxSockAddrLength"=dword:00000010
"MinSockAddrLength"=dword:00000010
"Mapping"=hex:0b,00,00,00,03,00,00,00,02,00,00,00,01,00,00,00,06,00,00,00,02,\
  00,00,00,01,00,00,00,00,00,00,00,02,00,00,00,00,00,00,00,06,00,00,00,00,00,\
  00,00,00,00,00,00,06,00,00,00,00,00,00,00,01,00,00,00,06,00,00,00,02,00,00,\
  00,02,00,00,00,11,00,00,00,02,00,00,00,02,00,00,00,00,00,00,00,02,00,00,00,\
  00,00,00,00,11,00,00,00,00,00,00,00,00,00,00,11,00,00,00,00,00,00,00,02,\
  00,00,00,11,00,00,00,02,00,00,00,03,00,00,00,00,00,00,00

tpccCom.tpcc_com_settings.txt

Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\TPCC]
"dbInterfacePath"="C:\\inetpub\\wwwroot\\tpcc\\tpccOracleglue.dll"
"dbName"="oratpcc.world"
"dbPassword"="tpcc"
"dbType"="ORACLE"
"dbUserName"="tpcc"
"dlvyLogPath"="C:\\inetpub\\wwwroot\\tpcc\\dlvy"
"dlvyQueueLen"=dword:00004e20
"dlvyThreads"=dword:00000002
"errorLogFile"="c:\\inetpub\\wwwroot\\tpcc\\isapi_err.log"
"htmlTrace"=dword:00000000
"htmlTraceLogFile"="c:\\inetpub\\wwwroot\\tpcc\\isapi.log"
"isapi_trace"=dword:00000000
"nullDB"=dword:00000000
"numPools"=dword:00000001
"numServers"=dword:00000001
"numUsers"=dword:00007530
"numWarehouse"=dword:000009c4
“OracleHome”=”C:\\ora10g”
“OracleComServLog”=”C:\\inetpub\\wwwroot\\tpcc\\oci_err.log”

tpcc_software_registry.reg client1

Transactions not supported
Enable Object pooling
Minimum pool size 30
Maximum pool size 30
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required

tpcc_software_registry.reg client2

Transactions not supported
Enable Object pooling
Minimum pool size 30
Maximum pool size 30
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required

tpcc_software_registry.reg client3

Transactions not supported
Enable Object pooling
Minimum pool size 30
Maximum pool size 30
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required

tpcc_software_registry.reg client4

Transactions not supported
Enable Object pooling
Minimum pool size 30
Maximum pool size 30
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required

tpcc_software_registry.reg client5

Transactions not supported
Enable Object pooling
Minimum pool size 30
Maximum pool size 30
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required

tpcc_software_registry.reg client6

Transactions not supported
Enable Object pooling
Minimum pool size 30
Maximum pool size 30
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required

tpcc_software_registry.reg client7

Transactions not supported
Enable Object pooling
Minimum pool size 30
Maximum pool size 30
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required

tpcc_software_registry.reg client8

Transactions not supported
Enable Object pooling
Minimum pool size 30
Maximum pool size 30
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required

10.3. Linux Parameters

B3. Linux Parameters

dev.parport.default.spintime = 500
dev.parport.default.timeslice = 200
dev.cdrom.check_media = 0
dev.cdrom.lock = 1
dev.cdrom.debug = 0
dev.cdrom.autoeject = 0
dev.cdrom.autoclose = 1
dev.cdrom.info = CD-ROM information, Id: cdrom.c 3.20 2003/12/17
dev.cdrom.info =
dev.cdrom.info = drive name:	hde
dev.cdrom.info = drive speed:	24
dev.cdrom.info = drive # of slots:	1
dev.cdrom.info = Can close tray:	1
dev.cdrom.info = Can open tray:	1
dev.cdrom.info = Can lock tray:	1
dev.cdrom.info = Can change speed:	1
dev.cdrom.info = Can select disk:	0
dev.cdrom.info = Can read multisession:	1
dev.cdrom.info = Can read MCN:	1
dev.cdrom.info = Reports media changed:	1
dev.cdrom.info = Can play audio:	1
dev.cdrom.info = Can write CD-R:	0
dev.cdrom.info = Can write CD-RW:	0
dev.cdrom.info = Can read DVD:	1
dev.cdrom.info = Can write DVD-R:	0
dev.cdrom.info = Can write DVD-RAM:	0
dev.cdrom.info = Can read MRW:	1
dev.cdrom.info = Can write MRW:	1
dev.cdrom.info = Can write RAM:	0
dev.cdrom.info =
dev.cdrom.info =
dev.scsi.logging_level = 0
dev.raid.speed_limit_max = 200000
dev.raid.speed_limit_min = 1000
net.ipv6.conf.eth4.router_probe_interval = 60
net.ipv6.conf.eth4.accept_ra_rtr_pref = 1
net.ipv6.conf.eth4.accept_ra_pinfo = 1
net.ipv6.conf.eth4.accept_ra_defrtr = 1
net.ipv6.conf.eth4.max_addresses = 16
net.ipv6.conf.eth4.max_desync_factor = 600
net.ipv6.conf.eth4.regen_max_retry = 5
net.ipv6.conf.eth4.temp_prefered_lft = 86400
net.ipv6.conf.eth4.temp_valid_lft = 604800
net.ipv6.conf.eth4.use_tempaddr = 0
net.ipv6.conf.eth4.force_mld_version = 0
net.ipv6.conf.eth4.router_solicitation_delay = 1
net.ipv6.conf.eth4.router_solicitation_interval = 4
net.ipv6.conf.eth4.router_solicitations = 3
net.ipv6.conf.eth4.dad_transmits = 1
net.ipv6.conf.eth4.autoconf = 1
net.ipv6.conf.eth4.accept_redirects = 1
net.ipv6.conf.eth4.accept_ra = 1
net.ipv6.conf.eth4.mtu = 1500
net.ipv6.conf.eth4.hop_limit = 64
net.ipv6.conf.eth4.forwarding = 0
net.ipv6.conf.eth3.router_probe_interval = 60
net.ipv6.conf.eth3.accept_ra_rtr_pref = 1
net.ipv6.conf.eth3.accept_ra_pinfo = 1
net.ipv6.conf.eth3.accept_ra_defrtr = 1
net.ipv6.conf.eth3.max_addresses = 16
net.ipv6.conf.eth3.max_desync_factor = 600
net.ipv6.conf.eth3.regen_max_retry = 5
net.ipv6.conf.eth3.temp_prefered_lft = 86400
net.ipv6.conf.eth3.temp_valid_lft = 604800
net.ipv6.conf.eth3.use_tempaddr = 0
net.ipv6.conf.eth3.force_mld_version = 0
net.ipv6.conf.eth3.router_solicitation_delay = 1
net.ipv6.conf.eth3.router_solicitation_interval = 4
net.ipv6.conf.eth3.router_solicitations = 3
net.ipv6.conf.eth3.dad_transmits = 1
net.ipv6.conf.eth3.autoconf = 1
net.ipv6.conf.eth3.accept_redirects = 1
net.ipv6.conf.eth3.accept_ra = 1
net.ipv6.conf.eth3.mtu = 1500
net.ipv6.conf.eth3.hop_limit = 64
net.ipv6.conf.eth3.forwarding = 0
net.ipv6.conf.eth2.router_probe_interval = 60
net.ipv6.conf.eth2.accept_ra_rtr_pref = 1
net.ipv6.conf.eth2.accept_ra_pinfo = 1
net.ipv6.conf.eth2.accept_ra_defrtr = 1
net.ipv6.conf.eth2.max_addresses = 16

net.ipv6.conf.eth2.max_desync_factor = 600
net.ipv6.conf.eth2.regen_max_retry = 5
net.ipv6.conf.eth2.temp_prefered_lft = 86400
net.ipv6.conf.eth2.temp_valid_lft = 604800
net.ipv6.conf.eth2.use_tempaddr = 0
net.ipv6.conf.eth2.force_mld_version = 0
net.ipv6.conf.eth2.router_solicitation_delay = 1
net.ipv6.conf.eth2.router_solicitation_interval = 4
net.ipv6.conf.eth2.router_solicitations = 3
net.ipv6.conf.eth2.dad_transmits = 1
net.ipv6.conf.eth2.autoconf = 1
net.ipv6.conf.eth2.accept_redirects = 1
net.ipv6.conf.eth2.accept_ra = 1
net.ipv6.conf.eth2.mtu = 1500
net.ipv6.conf.eth2.hop_limit = 64
net.ipv6.conf.eth2.forwarding = 0
net.ipv6.conf.eth1.router_probe_interval = 60
net.ipv6.conf.eth1.accept_ra_rtr_pref = 1
net.ipv6.conf.eth1.accept_ra_pinfo = 1
net.ipv6.conf.eth1.accept_ra_defrtr = 1
net.ipv6.conf.eth1.max_addresses = 16
net.ipv6.conf.eth1.max_desync_factor = 600
net.ipv6.conf.eth1.regen_max_retry = 5
net.ipv6.conf.eth1.temp_prefered_lft = 86400
net.ipv6.conf.eth1.temp_valid_lft = 604800
net.ipv6.conf.eth1.use_tempaddr = 0
net.ipv6.conf.eth1.force_mld_version = 0
net.ipv6.conf.eth1.router_solicitation_delay = 1
net.ipv6.conf.eth1.router_solicitation_interval = 4
net.ipv6.conf.eth1.router_solicitations = 3
net.ipv6.conf.eth1.dad_transmits = 1
net.ipv6.conf.eth1.autoconf = 1
net.ipv6.conf.eth1.accept_redirects = 1
net.ipv6.conf.eth1.accept_ra = 1
net.ipv6.conf.eth1.mtu = 1500
net.ipv6.conf.eth1.hop_limit = 64
net.ipv6.conf.eth1.forwarding = 0
net.ipv6.conf.eth0.router_probe_interval = 60
net.ipv6.conf.eth0.accept_ra_rtr_pref = 1
net.ipv6.conf.eth0.accept_ra_pinfo = 1
net.ipv6.conf.eth0.accept_ra_defrtr = 1
net.ipv6.conf.eth0.max_addresses = 16
net.ipv6.conf.eth0.max_desync_factor = 600
net.ipv6.conf.eth0.regen_max_retry = 5
net.ipv6.conf.eth0.temp_prefered_lft = 86400
net.ipv6.conf.eth0.temp_valid_lft = 604800
net.ipv6.conf.eth0.use_tempaddr = 0
net.ipv6.conf.eth0.force_mld_version = 0
net.ipv6.conf.eth0.router_solicitation_delay = 1
net.ipv6.conf.eth0.router_solicitation_interval = 4
net.ipv6.conf.eth0.router_solicitations = 3
net.ipv6.conf.eth0.dad_transmits = 1
net.ipv6.conf.eth0.autoconf = 1
net.ipv6.conf.eth0.accept_redirects = 1
net.ipv6.conf.eth0.accept_ra = 1
net.ipv6.conf.eth0.mtu = 1500
net.ipv6.conf.eth0.hop_limit = 64
net.ipv6.conf.eth0.forwarding = 0
net.ipv6.conf.default.router_probe_interval = 60
net.ipv6.conf.default.accept_ra_rtr_pref = 1
net.ipv6.conf.default.accept_ra_pinfo = 1
net.ipv6.conf.default.accept_ra_defrtr = 1
net.ipv6.conf.default.max_addresses = 16
net.ipv6.conf.default.max_desync_factor = 600
net.ipv6.conf.default.regen_max_retry = 5
net.ipv6.conf.default.temp_prefered_lft = 86400
net.ipv6.conf.default.temp_valid_lft = 604800
net.ipv6.conf.default.use_tempaddr = 0
net.ipv6.conf.default.force_mld_version = 0
net.ipv6.conf.default.router_solicitation_delay = 1
net.ipv6.conf.default.router_solicitation_interval = 4
net.ipv6.conf.default.router_solicitations = 3
net.ipv6.conf.default.dad_transmits = 1
net.ipv6.conf.default.autoconf = 1
net.ipv6.conf.default.accept_redirects = 1
net.ipv6.conf.default.accept_ra = 1
net.ipv6.conf.default.mtu = 1280
net.ipv6.conf.default.hop_limit = 64
net.ipv6.conf.default.forwarding = 0
net.ipv6.conf.all.router_probe_interval = 60
net.ipv6.conf.all.accept_ra_rtr_pref = 1
net.ipv6.conf.all.accept_ra_pinfo = 1
net.ipv6.conf.all.accept_ra_defrtr = 1
net.ipv6.conf.all.max_addresses = 16
net.ipv6.conf.all.max_desync_factor = 600
net.ipv6.conf.all.regen_max_retry = 5
net.ipv6.conf.all.temp_prefered_lft = 86400
net.ipv6.conf.all.temp_valid_lft = 604800
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.all.force_mld_version = 0
net.ipv6.conf.all.router_solicitation_delay = 1
net.ipv6.conf.all.router_solicitation_interval = 4
net.ipv6.conf.all.router_solicitations = 3
net.ipv6.conf.all.dad_transmits = 1
net.ipv6.conf.all.autoconf = 1
net.ipv6.conf.all.accept_redirects = 1
net.ipv6.conf.all.accept_ra = 1
net.ipv6.conf.all.mtu = 1280
net.ipv6.conf.all.hop_limit = 64
net.ipv6.conf.all.forwarding = 0
net.ipv6.conf.lo.router_probe_interval = 60
net.ipv6.conf.lo.accept_ra_rtr_pref = 1
net.ipv6.conf.lo.accept_ra_pinfo = 1
net.ipv6.conf.lo.accept_ra_defrtr = 1
net.ipv6.conf.lo.max_addresses = 16
net.ipv6.conf.lo.max_desync_factor = 600
net.ipv6.conf.lo.regen_max_retry = 5
net.ipv6.conf.lo.temp_prefered_lft = 86400
net.ipv6.conf.lo.temp_valid_lft = 604800
net.ipv6.conf.lo.use_tempaddr = -1
net.ipv6.conf.lo.force_mld_version = 0
net.ipv6.conf.lo.router_solicitation_delay = 1
net.ipv6.conf.lo.router_solicitation_interval = 4
net.ipv6.conf.lo.router_solicitations = 3
net.ipv6.conf.lo.dad_transmits = 1
net.ipv6.conf.lo.autoconf = 1
net.ipv6.conf.lo.accept_redirects = 1
net.ipv6.conf.lo.accept_ra = 1
net.ipv6.conf.lo.mtu = 16436
net.ipv6.conf.lo.hop_limit = 64
net.ipv6.conf.lo.forwarding = 0
net.ipv6.neigh.eth4.base_reachable_time_ms = 30000
net.ipv6.neigh.eth4.retrans_time_ms = 1000
net.ipv6.neigh.eth4.locktime = 0
net.ipv6.neigh.eth4.proxy_delay = 80
net.ipv6.neigh.eth4.anycast_delay = 100
net.ipv6.neigh.eth4.proxy_qlen = 64
net.ipv6.neigh.eth4.unres_qlen = 3
net.ipv6.neigh.eth4.gc_stale_time = 60
net.ipv6.neigh.eth4.delay_first_probe_time = 5
net.ipv6.neigh.eth4.retrans_time = 1000
net.ipv6.neigh.eth4.app_solicit = 0
net.ipv6.neigh.eth4.ucast_solicit = 3
net.ipv6.neigh.eth4.mcast_solicit = 3
net.ipv6.neigh.eth3.base_reachable_time_ms = 30000
net.ipv6.neigh.eth3.retrans_time_ms = 1000
net.ipv6.neigh.eth3.locktime = 0
net.ipv6.neigh.eth3.proxy_delay = 80
net.ipv6.neigh.eth3.anycast_delay = 100
net.ipv6.neigh.eth3.proxy_qlen = 64
net.ipv6.neigh.eth3.unres_qlen = 3
net.ipv6.neigh.eth3.gc_stale_time = 60
net.ipv6.neigh.eth3.delay_first_probe_time = 5
net.ipv6.neigh.eth3.retrans_time = 1000
net.ipv6.neigh.eth3.app_solicit = 0
net.ipv6.neigh.eth3.ucast_solicit = 3
net.ipv6.neigh.eth3.mcast_solicit = 3
net.ipv6.neigh.eth2.base_reachable_time_ms = 30000
net.ipv6.neigh.eth2.retrans_time_ms = 1000
net.ipv6.neigh.eth2.locktime = 0
net.ipv6.neigh.eth2.proxy_delay = 80
net.ipv6.neigh.eth2.anycast_delay = 100
net.ipv6.neigh.eth2.proxy_qlen = 64
net.ipv6.neigh.eth2.unres_qlen = 3
net.ipv6.neigh.eth2.gc_stale_time = 60
net.ipv6.neigh.eth2.delay_first_probe_time = 5
net.ipv6.neigh.eth2.retrans_time = 1000
net.ipv6.neigh.eth2.app_solicit = 0
net.ipv6.neigh.eth2.ucast_solicit = 3
net.ipv6.neigh.eth2.mcast_solicit = 3
net.ipv6.neigh.eth1.base_reachable_time_ms = 30000
net.ipv6.neigh.eth1.retrans_time_ms = 1000
net.ipv6.neigh.eth1.locktime = 0
net.ipv6.neigh.eth1.proxy_delay = 80
net.ipv6.neigh.eth1.anycast_delay = 100
net.ipv6.neigh.eth1.proxy_qlen = 64
net.ipv6.neigh.eth1.unres_qlen = 3
net.ipv6.neigh.eth1.gc_stale_time = 60
net.ipv6.neigh.eth1.delay_first_probe_time = 5
net.ipv6.neigh.eth1.retrans_time = 1000
net.ipv6.neigh.eth1.app_solicit = 0
net.ipv6.neigh.eth1.ucast_solicit = 3
net.ipv6.neigh.eth1.mcast_solicit = 3
net.ipv6.neigh.eth0.base_reachable_time_ms = 30000
net.ipv6.neigh.eth0.retrans_time_ms = 1000
net.ipv6.neigh.eth0.locktime = 0
net.ipv6.neigh.eth0.proxy_delay = 80
net.ipv6.neigh.eth0.anycast_delay = 100
net.ipv6.neigh.eth0.proxy_qlen = 64
net.ipv6.neigh.eth0.unres_qlen = 3
net.ipv6.neigh.eth0.gc_stale_time = 60
net.ipv6.neigh.eth0.delay_first_probe_time = 5
net.ipv6.neigh.eth0.retrans_time = 1000
net.ipv6.neigh.eth0.app_solicit = 0
net.ipv6.neigh.eth0.ucast_solicit = 3
net.ipv6.neigh.eth0.mcast_solicit = 3
net.ipv6.neigh.lo.base_reachable_time_ms = 30000
net.ipv6.neigh.lo.retrans_time_ms = 1000
net.ipv6.neigh.lo.locktime = 0
net.ipv6.neigh.lo.proxy_delay = 80
net.ipv6.neigh.lo.anycast_delay = 100
net.ipv6.neigh.lo.proxy_qlen = 64
net.ipv6.neigh.lo.unres_qlen = 3
net.ipv6.neigh.lo.gc_stale_time = 60
net.ipv6.neigh.lo.delay_first_probe_time = 5
net.ipv6.neigh.lo.retrans_time = 1000
net.ipv6.neigh.lo.app_solicit = 0
net.ipv6.neigh.lo.ucast_solicit = 3
net.ipv6.neigh.lo.mcast_solicit = 3
net.ipv6.neigh.default.base_reachable_time_ms = 30000
net.ipv6.neigh.default.retrans_time_ms = 1000
net.ipv6.neigh.default.gc_thresh3 = 1024
net.ipv6.neigh.default.gc_thresh2 = 512
net.ipv6.neigh.default.gc_thresh1 = 128
net.ipv6.neigh.default.gc_interval = 30
net.ipv6.neigh.default.locktime = 0
net.ipv6.neigh.default.proxy_delay = 80
net.ipv6.neigh.default.anycast_delay = 100
net.ipv6.neigh.default.proxy_qlen = 64
net.ipv6.neigh.default.unres_qlen = 3

net.ipv6.neigh.default.gc_stale_time = 60
net.ipv6.neigh.default.delay_first_probe_time = 5
net.ipv6.neigh.default.retrans_time = 1000
net.ipv6.neigh.default.app_solicit = 0
net.ipv6.neigh.default.ucast_solicit = 3
net.ipv6.neigh.default.mcast_solicit = 3
net.ipv6.mld_max_msf = 64
net.ipv6.ip6frag_secret_interval = 600
net.ipv6.ip6frag_time = 60
net.ipv6.ip6frag_low_thresh = 196608
net.ipv6.ip6frag_high_thresh = 262144
net.ipv6.bindv6only = 0
net.ipv6.icmp.ratelimit = 1000
net.ipv6.route.gc_min_interval_ms = 500
net.ipv6.route.min_adv_mss = 1
net.ipv6.route.mtu_expires = 600
net.ipv6.route.gc_elasticity = 0
net.ipv6.route.gc_interval = 30
net.ipv6.route.gc_timeout = 60
net.ipv6.route.gc_min_interval = 0
net.ipv6.route.max_size = 4096
net.ipv6.route.gc_thresh = 1024
net.unix.max_dgram_qlen = 10
net.token-ring.rif_timeout = 600000
net.ipv4.conf.eth4.promote_secondaries = 0
net.ipv4.conf.eth4.force_igmp_version = 0
net.ipv4.conf.eth4.disable_policy = 0
net.ipv4.conf.eth4.disable_xfrm = 0
net.ipv4.conf.eth4.arp_accept = 0
net.ipv4.conf.eth4.arp_ignore = 0
net.ipv4.conf.eth4.arp_announce = 0
net.ipv4.conf.eth4.arp_filter = 0
net.ipv4.conf.eth4.tag = 0
net.ipv4.conf.eth4.log_martians = 0
net.ipv4.conf.eth4.bootp_relay = 0
net.ipv4.conf.eth4.medium_id = 0
net.ipv4.conf.eth4.proxy_arp = 0
net.ipv4.conf.eth4.accept_source_route = 0
net.ipv4.conf.eth4.send_redirects = 1
net.ipv4.conf.eth4.rp_filter = 1
net.ipv4.conf.eth4.shared_media = 1
net.ipv4.conf.eth4.secure_redirects = 1
net.ipv4.conf.eth4.accept_redirects = 1
net.ipv4.conf.eth4.mc_forwarding = 0
net.ipv4.conf.eth4.forwarding = 0
net.ipv4.conf.eth3.promote_secondaries = 0
net.ipv4.conf.eth3.force_igmp_version = 0
net.ipv4.conf.eth3.disable_policy = 0
net.ipv4.conf.eth3.disable_xfrm = 0
net.ipv4.conf.eth3.arp_accept = 0
net.ipv4.conf.eth3.arp_ignore = 0
net.ipv4.conf.eth3.arp_announce = 0
net.ipv4.conf.eth3.arp_filter = 0
net.ipv4.conf.eth3.tag = 0
net.ipv4.conf.eth3.log_martians = 0
net.ipv4.conf.eth3.bootp_relay = 0
net.ipv4.conf.eth3.medium_id = 0
net.ipv4.conf.eth3.proxy_arp = 0
net.ipv4.conf.eth3.accept_source_route = 0
net.ipv4.conf.eth3.send_redirects = 1
net.ipv4.conf.eth3.rp_filter = 1
net.ipv4.conf.eth3.shared_media = 1
net.ipv4.conf.eth3.secure_redirects = 1
net.ipv4.conf.eth3.accept_redirects = 1
net.ipv4.conf.eth3.mc_forwarding = 0
net.ipv4.conf.eth3.forwarding = 0
net.ipv4.conf.eth2.promote_secondaries = 0
net.ipv4.conf.eth2.force_igmp_version = 0
net.ipv4.conf.eth2.disable_policy = 0
net.ipv4.conf.eth2.disable_xfrm = 0
net.ipv4.conf.eth2.arp_accept = 0
net.ipv4.conf.eth2.arp_ignore = 0
net.ipv4.conf.eth2.arp_announce = 0
net.ipv4.conf.eth2.arp_filter = 0
net.ipv4.conf.eth2.tag = 0
net.ipv4.conf.eth2.log_martians = 0
net.ipv4.conf.eth2.bootp_relay = 0
net.ipv4.conf.eth2.medium_id = 0
net.ipv4.conf.eth2.proxy_arp = 0
net.ipv4.conf.eth2.accept_source_route = 0
net.ipv4.conf.eth2.send_redirects = 1
net.ipv4.conf.eth2.rp_filter = 1
net.ipv4.conf.eth2.shared_media = 1
net.ipv4.conf.eth2.secure_redirects = 1
net.ipv4.conf.eth2.accept_redirects = 1
net.ipv4.conf.eth2.mc_forwarding = 0
net.ipv4.conf.eth2.forwarding = 0
net.ipv4.conf.eth1.promote_secondaries = 0
net.ipv4.conf.eth1.force_igmp_version = 0
net.ipv4.conf.eth1.disable_policy = 0
net.ipv4.conf.eth1.disable_xfrm = 0
net.ipv4.conf.eth1.arp_accept = 0
net.ipv4.conf.eth1.arp_ignore = 0
net.ipv4.conf.eth1.arp_announce = 0
net.ipv4.conf.eth1.arp_filter = 0
net.ipv4.conf.eth1.tag = 0
net.ipv4.conf.eth1.log_martians = 0
net.ipv4.conf.eth1.bootp_relay = 0
net.ipv4.conf.eth1.medium_id = 0
net.ipv4.conf.eth1.proxy_arp = 0
net.ipv4.conf.eth1.accept_source_route = 0
net.ipv4.conf.eth1.send_redirects = 1
net.ipv4.conf.eth1.rp_filter = 1
net.ipv4.conf.eth1.shared_media = 1
net.ipv4.conf.eth1.secure_redirects = 1
net.ipv4.conf.eth1.accept_redirects = 1
net.ipv4.conf.eth1.mc_forwarding = 0
net.ipv4.conf.eth1.forwarding = 0
net.ipv4.conf.eth0.promote_secondaries = 0
net.ipv4.conf.eth0.force_igmp_version = 0
net.ipv4.conf.eth0.disable_policy = 0
net.ipv4.conf.eth0.disable_xfrm = 0
net.ipv4.conf.eth0.arp_accept = 0
net.ipv4.conf.eth0.arp_ignore = 0
net.ipv4.conf.eth0.arp_announce = 0
net.ipv4.conf.eth0.arp_filter = 0
net.ipv4.conf.eth0.tag = 0
net.ipv4.conf.eth0.log_martians = 0
net.ipv4.conf.eth0.bootp_relay = 0
net.ipv4.conf.eth0.medium_id = 0
net.ipv4.conf.eth0.proxy_arp = 0
net.ipv4.conf.eth0.accept_source_route = 0
net.ipv4.conf.eth0.send_redirects = 1
net.ipv4.conf.eth0.rp_filter = 1
net.ipv4.conf.eth0.shared_media = 1
net.ipv4.conf.eth0.secure_redirects = 1
net.ipv4.conf.eth0.accept_redirects = 1
net.ipv4.conf.eth0.mc_forwarding = 0
net.ipv4.conf.eth0.forwarding = 0
net.ipv4.conf.lo.promote_secondaries = 0
net.ipv4.conf.lo.force_igmp_version = 0
net.ipv4.conf.lo.disable_policy = 1
net.ipv4.conf.lo.disable_xfrm = 1
net.ipv4.conf.lo.arp_accept = 0
net.ipv4.conf.lo.arp_ignore = 0
net.ipv4.conf.lo.arp_announce = 0
net.ipv4.conf.lo.arp_filter = 0
net.ipv4.conf.lo.tag = 0
net.ipv4.conf.lo.log_martians = 0
net.ipv4.conf.lo.bootp_relay = 0
net.ipv4.conf.lo.medium_id = 0
net.ipv4.conf.lo.proxy_arp = 0
net.ipv4.conf.lo.accept_source_route = 1
net.ipv4.conf.lo.send_redirects = 1
net.ipv4.conf.lo.rp_filter = 0
net.ipv4.conf.lo.shared_media = 1
net.ipv4.conf.lo.secure_redirects = 1
net.ipv4.conf.lo.accept_redirects = 1
net.ipv4.conf.lo.mc_forwarding = 0
net.ipv4.conf.lo.forwarding = 0
net.ipv4.conf.default.promote_secondaries = 0
net.ipv4.conf.default.force_igmp_version = 0
net.ipv4.conf.default.disable_policy = 0
net.ipv4.conf.default.disable_xfrm = 0
net.ipv4.conf.default.arp_accept = 0
net.ipv4.conf.default.arp_ignore = 0
net.ipv4.conf.default.arp_announce = 0
net.ipv4.conf.default.arp_filter = 0
net.ipv4.conf.default.tag = 0
net.ipv4.conf.default.log_martians = 0
net.ipv4.conf.default.bootp_relay = 0
net.ipv4.conf.default.medium_id = 0
net.ipv4.conf.default.proxy_arp = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.shared_media = 1
net.ipv4.conf.default.secure_redirects = 1
net.ipv4.conf.default.accept_redirects = 1
net.ipv4.conf.default.mc_forwarding = 0
net.ipv4.conf.default.forwarding = 0
net.ipv4.conf.all.promote_secondaries = 0
net.ipv4.conf.all.force_igmp_version = 0
net.ipv4.conf.all.disable_policy = 0
net.ipv4.conf.all.disable_xfrm = 0
net.ipv4.conf.all.arp_accept = 0
net.ipv4.conf.all.arp_ignore = 0
net.ipv4.conf.all.arp_announce = 0
net.ipv4.conf.all.arp_filter = 0
net.ipv4.conf.all.tag = 0
net.ipv4.conf.all.log_martians = 0
net.ipv4.conf.all.bootp_relay = 0
net.ipv4.conf.all.medium_id = 0
net.ipv4.conf.all.proxy_arp = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.send_redirects = 1
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.all.shared_media = 1
net.ipv4.conf.all.secure_redirects = 1
net.ipv4.conf.all.accept_redirects = 1
net.ipv4.conf.all.mc_forwarding = 0
net.ipv4.conf.all.forwarding = 0
net.ipv4.neigh.eth4.base_reachable_time_ms = 30000
net.ipv4.neigh.eth4.retrans_time_ms = 1000
net.ipv4.neigh.eth4.locktime = 100
net.ipv4.neigh.eth4.proxy_delay = 80
net.ipv4.neigh.eth4.anycast_delay = 100
net.ipv4.neigh.eth4.proxy_qlen = 64
net.ipv4.neigh.eth4.unres_qlen = 3
net.ipv4.neigh.eth4.gc_stale_time = 60
net.ipv4.neigh.eth4.delay_first_probe_time = 5
net.ipv4.neigh.eth4.base_reachable_time = 30
net.ipv4.neigh.eth4.retrans_time = 100
net.ipv4.neigh.eth4.app_solicit = 0
net.ipv4.neigh.eth4.ucast_solicit = 3
net.ipv4.neigh.eth4.mcast_solicit = 3
net.ipv4.neigh.eth3.base_reachable_time_ms = 30000
net.ipv4.neigh.eth3.retrans_time_ms = 1000
net.ipv4.neigh.eth3.locktime = 100
net.ipv4.neigh.eth3.proxy_delay = 80

```
net.ipv4.neigh.eth3.anycast_delay = 100
net.ipv4.neigh.eth3.proxy_qlen = 64
net.ipv4.neigh.eth3.unres_qlen = 3
net.ipv4.neigh.eth3.gc_stale_time = 60
net.ipv4.neigh.eth3.delay_first_probe_time = 5
net.ipv4.neigh.eth3.base_reachable_time = 30
net.ipv4.neigh.eth3.retrans_time = 100
net.ipv4.neigh.eth3.app_solicit = 0
net.ipv4.neigh.eth3.ucast_solicit = 3
net.ipv4.neigh.eth3.mcast_solicit = 3
net.ipv4.neigh.eth2.base_reachable_time_ms = 30000
net.ipv4.neigh.eth2.retrans_time_ms = 1000
net.ipv4.neigh.eth2.locktime = 100
net.ipv4.neigh.eth2.proxy_delay = 80
net.ipv4.neigh.eth2.anycast_delay = 100
net.ipv4.neigh.eth2.proxy_qlen = 64
net.ipv4.neigh.eth2.unres_qlen = 3
net.ipv4.neigh.eth2.gc_stale_time = 60
net.ipv4.neigh.eth2.delay_first_probe_time = 5
net.ipv4.neigh.eth2.base_reachable_time = 30
net.ipv4.neigh.eth2.retrans_time = 100
net.ipv4.neigh.eth2.app_solicit = 0
net.ipv4.neigh.eth2.ucast_solicit = 3
net.ipv4.neigh.eth2.mcast_solicit = 3
net.ipv4.neigh.eth1.base_reachable_time_ms = 30000
net.ipv4.neigh.eth1.retrans_time_ms = 1000
net.ipv4.neigh.eth1.locktime = 100
net.ipv4.neigh.eth1.proxy_delay = 80
net.ipv4.neigh.eth1.anycast_delay = 100
net.ipv4.neigh.eth1.proxy_qlen = 64
net.ipv4.neigh.eth1.unres_qlen = 3
net.ipv4.neigh.eth1.gc_stale_time = 60
net.ipv4.neigh.eth1.delay_first_probe_time = 5
net.ipv4.neigh.eth1.base_reachable_time = 30
net.ipv4.neigh.eth1.retrans_time = 100
net.ipv4.neigh.eth1.app_solicit = 0
net.ipv4.neigh.eth1.ucast_solicit = 3
net.ipv4.neigh.eth1.mcast_solicit = 3
net.ipv4.neigh.eth0.base_reachable_time_ms = 30000
net.ipv4.neigh.eth0.retrans_time_ms = 1000
net.ipv4.neigh.eth0.locktime = 100
net.ipv4.neigh.eth0.proxy_delay = 80
net.ipv4.neigh.eth0.anycast_delay = 100
net.ipv4.neigh.eth0.proxy_qlen = 64
net.ipv4.neigh.eth0.unres_qlen = 3
net.ipv4.neigh.eth0.gc_stale_time = 60
net.ipv4.neigh.eth0.delay_first_probe_time = 5
net.ipv4.neigh.eth0.base_reachable_time = 30
net.ipv4.neigh.eth0.retrans_time = 100
net.ipv4.neigh.eth0.app_solicit = 0
net.ipv4.neigh.eth0.ucast_solicit = 3
net.ipv4.neigh.eth0.mcast_solicit = 3
net.ipv4.neigh.lo.base_reachable_time_ms = 30000
net.ipv4.neigh.lo.retrans_time_ms = 1000
net.ipv4.neigh.lo.locktime = 100
net.ipv4.neigh.lo.proxy_delay = 80
net.ipv4.neigh.lo.anycast_delay = 100
net.ipv4.neigh.lo.proxy_qlen = 64
net.ipv4.neigh.lo.unres_qlen = 3
net.ipv4.neigh.lo.gc_stale_time = 60
net.ipv4.neigh.lo.delay_first_probe_time = 5
net.ipv4.neigh.lo.base_reachable_time = 30
net.ipv4.neigh.lo.retrans_time = 100
net.ipv4.neigh.lo.app_solicit = 0
net.ipv4.neigh.lo.ucast_solicit = 3
net.ipv4.neigh.lo.mcast_solicit = 3
net.ipv4.neigh.default.base_reachable_time_ms = 30000
net.ipv4.neigh.default.retrans_time_ms = 1000
net.ipv4.neigh.default.gc_thresh3 = 1024
net.ipv4.neigh.default.gc_thresh2 = 512
net.ipv4.neigh.default.gc_thresh1 = 128
net.ipv4.neigh.default.gc_interval = 30
net.ipv4.neigh.default.locktime = 100
net.ipv4.neigh.default.proxy_delay = 80
net.ipv4.neigh.default.anycast_delay = 100
net.ipv4.neigh.default.proxy_qlen = 64
net.ipv4.neigh.default.unres_qlen = 3
net.ipv4.neigh.default.gc_stale_time = 60
net.ipv4.neigh.default.delay_first_probe_time = 5
net.ipv4.neigh.default.base_reachable_time = 30
net.ipv4.neigh.default.retrans_time = 100
net.ipv4.neigh.default.app_solicit = 0
net.ipv4.neigh.default.ucast_solicit = 3
net.ipv4.neigh.default.mcast_solicit = 3
net.ipv4.cipso_rbm_strictvalid = 1
net.ipv4.cipso_rbm_optfmt = 0
net.ipv4.cipso_cache_bucket_size = 10
net.ipv4.cipso_cache_enable = 1
net.ipv4.tcp_slow_start_after_idle = 1
net.ipv4.tcp_dma_copybreak = 4096
net.ipv4.tcp_workaround_signed_windows = 0
net.ipv4.tcp_base_mss = 512
net.ipv4.tcp_mtu_probing = 0
net.ipv4.tcp_abc = 0
net.ipv4.tcp_congestion_control = bic
net.ipv4.tcp_tso_win_divisor = 3
net.ipv4.tcp_moderate_rcvbuf = 1
net.ipv4.tcp_no_metrics_save = 0
net.ipv4.ipfrag_max_dist = 64
net.ipv4.ipfrag_secret_interval = 600
net.ipv4.tcp_low_latency = 0
net.ipv4.tcp_frto = 0
net.ipv4.tcp_tw_reuse = 0
net.ipv4.icmp_ratemask = 6168
net.ipv4.icmp_ratelimit = 1000
net.ipv4.tcp_adv_win_scale = 2
net.ipv4.tcp_app_win = 31
net.ipv4.tcp_rmem = 65536    87380   4194304
net.ipv4.tcp_wmem = 65536   16384   4194304
net.ipv4.tcp_mem = 12288 16384   24576
net.ipv4.tcp_dsack = 1
net.ipv4.tcp_ecn = 0
net.ipv4.tcp_reordering = 3
net.ipv4.tcp_fack = 1
net.ipv4.tcp_orphan_retries = 0
net.ipv4.inet_peer_gc_maxtime = 120
net.ipv4.inet_peer_gc_mintime = 10
net.ipv4.inet_peer_maxttl = 600
net.ipv4.inet_peer_minttl = 120
net.ipv4.inet_peer_threshold = 65664
net.ipv4.igmp_max_msf = 10
net.ipv4.igmp_max_memberships = 20
net.ipv4.route.secret_interval = 600
net.ipv4.route.min_adv_mss = 256
net.ipv4.route.min_pmtu = 552
net.ipv4.route.mtu_expires = 600
net.ipv4.route.gc_elasticity = 8
net.ipv4.route.error_burst = 5000
net.ipv4.route.error_cost = 1000
net.ipv4.route.redirect_silence = 20480
net.ipv4.route.redirect_number = 9
net.ipv4.route.redirect_load = 20
net.ipv4.route.gc_interval = 60
net.ipv4.route.gc_timeout = 300
net.ipv4.route.gc_min_interval_ms = 500
net.ipv4.route.gc_min_interval = 0
net.ipv4.route.max_size = 33554432
net.ipv4.route.gc_thresh = 2097152
net.ipv4.route.max_delay = 10
net.ipv4.route.min_delay = 2
net.ipv4.icmp_errors_use_inbound_ifaddr = 0
net.ipv4.icmp_ignore_bogus_error_responses = 1
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.icmp_echo_ignore_all = 0
net.ipv4.ip_local_port_range = 1024   65000
net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_rfc1337 = 0
net.ipv4.tcp_stdurg = 0
net.ipv4.tcp_abort_on_overflow = 0
net.ipv4.tcp_tw_recycle = 0
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_fin_timeout = 60
net.ipv4.tcp_retries2 = 15
net.ipv4.tcp_retries1 = 3
net.ipv4.tcp_keepalive_intvl = 75
net.ipv4.tcp_keepalive_probes = 9
net.ipv4.tcp_keepalive_time = 7200
net.ipv4.ipfrag_time = 30
net.ipv4.ip_dynaddr = 0
net.ipv4.ipfrag_low_thresh = 196608
net.ipv4.ipfrag_high_thresh = 262144
net.ipv4.tcp_max_tw_buckets = 180000
net.ipv4.tcp_max_orphans = 4096
net.ipv4.tcp_synack_retries = 5
net.ipv4.tcp_syn_retries = 5
net.ipv4.ip_nonlocal_bind = 0
net.ipv4.ip_no_pmtu_disc = 0
net.ipv4.ip_default_ttl = 64
net.ipv4.ip_forward = 0
net.ipv4.tcp_retrans_collapse = 1
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 1
net.core.netdev_budget = 300
net.core.somaxconn = 128
net.core.xfrm_aevent_rseqth = 2
net.core.xfrm_aevent_etime = 10
net.core.optmem_max = 20480
net.core.message_burst = 10
net.core.message_cost = 5
net.core.netdev_max_backlog = 1000
net.core.dev_weight = 64
net.core.rmem_default = 262144
net.core.wmem_default = 262144
net.core.rmem_max = 262144
net.core.wmem_max = 262144
vm.min_slab_ratio = 5
vm.min_unmapped_ratio = 1
vm.zone_reclaim_mode = 0
vm.swap_token_timeout = 0   300
vm.legacy_va_layout = 0
vm.vfs_cache_pressure = 100
vm.block_dump = 0
vm.laptop_mode = 0
vm.max_map_count = 65536
vm.percpu_pagelist_fraction = 0
vm.min_free_kbytes = 45863
vm.drop_caches = 0
vm.lowmem_reserve_ratio = 256    256   32
vm.hugetlb_shm_group = 500
vm.nr_hugepages = 7691
vm.swappiness = 60
vm.nr_pdflush_threads = 2
vm.dirty_expire_centisecs = 3000
vm.dirty_writeback_centisecs = 500
vm.dirty_ratio = 40
vm.dirty_background_ratio = 10
vm.page-cluster = 3
vm.overcommit_ratio = 50
vm.panic_on_oom = 0
```

```
vm.overcommit_memory = 0
kernel.powersave-nap = 0
kernel.max_lock_depth = 1024
kernel.compat-log = 1
kernel.randomize_va_space = 1
kernel.ngroups_max = 65536
kernel.printk_ratelimit_burst = 10
kernel.printk_ratelimit = 5
kernel.panic_on_oops = 1
kernel.pid_max = 32768
kernel.overflowgid = 65534
kernel.overflowuid = 65534
kernel.pty.nr = 2
kernel.pty.max = 4096
kernel.random.uuid = f65d51d5-c3b6-45b5-b73a-9ecf6dca4c60
kernel.random.boot_id = a28fc76e-ba65-4224-ac75-80dd8b702108
kernel.random.write_wakeup_threshold = 128
kernel.random.read_wakeup_threshold = 64
kernel.random.entropy_avail = 186
kernel.random.poolsize = 4096
kernel.threads-max = 1027072
kernel.cad_pid = 1
kernel.sysrq = 0
kernel.sem = 512  32000  512  256
kernel.msgmnb = 65536
kernel.msgmni = 16
kernel.msgmax = 65536
kernel.shmmni = 4096
kernel.shmall = 134285099008
kernel.shmmax = 134285099008
kernel.acct = 4  2  30
kernel.hotplug =
kernel.modprobe = /sbin/modprobe
kernel.printk = 64  1  7
kernel.ctrl-alt-del = 0
kernel.real-root-dev = 0
kernel.cap-bound = -257
kernel.tainted = 0
kernel.core_pattern = core
kernel.core_uses_pid = 1
kernel.print-fatal-signals = 0
kernel.exec-shield = 1
kernel.panic = 180
kernel.domainname = (none)
kernel.hostname = itcopus121
kernel.version = #1 SMP Fri Jan 26 14:19:36 EST 2007
kernel.osrelease = 2.6.18-8.el5
kernel.ostype = Linux
fs.mqueue.msgsize_max = 8192
fs.mqueue.msg_max = 10
fs.mqueue.queues_max = 256
fs.quota.warnings = 1
fs.quota.syncs = 18
fs.quota.free_dquots = 0
fs.quota.allocated_dquots = 0
fs.quota.cache_hits = 0
fs.quota.writes = 0
fs.quota.reads = 0
fs.quota.drops = 0
fs.quota.lookups = 0
fs.suid_dumpable = 0
fs.inotify.max_queued_events = 16384
fs.inotify.max_user_watches = 8192
fs.inotify.max_user_instances = 128
fs.aio-max-nr = 1600000
fs.aio-nr = 1072000
fs.lease-break-time = 45
fs.dir-notify-enable = 1
fs.leases-enable = 1
fs.overflowgid = 65534
fs.overflowuid = 65534
fs.dentry-state = 22409  9694 450  0  0
fs.file-max = 131072
fs.file-nr = 51712  0  131072
fs.inode-state = 21181  146  0  0  0  0  0
fs.inode-nr = 21181  146
fs.binfmt_misc.status = enabled
```

11 Appendix C: Database Setup Code

11.1. Database Creation Scripts

createuser.sh

```
#!/bin/sh

echo Creating user tpcc...
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_sql_dir/createuser > junk 2>&1
if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

createts.sh

```
#created automatically by buildcreatets.sh Fri Mar 2 10:57:40 CST 2007

# Tablespace ware, ts size 50M (51200K)
# each file 50M (51200K)
# extents 43824K (43824K)
# 1 files

$tpcc_createts ware 1 1       50M 43824K unix 0       0 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for ware failed.  Exiting.
      exit 0
    fi

# Tablespace cust, ts size 537340M (550236160K)
# each file 8020M (8212480K)
# extents 110892K (110892K)
# 67 files

$tpcc_createts cust 67 1       8020M 110892K unix 0       1 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for cust failed.  Exiting.
      exit 0
    fi

# Tablespace dist, ts size 430M (440320K)
# each file 430M (440320K)
# extents 429024K (429024K)
# 1 files

$tpcc_createts dist 1 1       430M 429024K unix 0       68 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for dist failed.  Exiting.
      exit 0
    fi

# Tablespace hist, ts size 52080M (53329920K)
# each file 7440M (7618560K)
# extents 102822K (102822K)
# 7 files

$tpcc_createts hist 7 1       7440M 102822K unix 0       69 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for hist failed.  Exiting.
      exit 0
    fi

# Tablespace stok, ts size 602250M (616704000K)
# each file 8030M (8222720K)
# extents 124544K (124544K)
# 75 files

$tpcc_createts stok 75 1       8030M 124544K unix 0       76 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for stok failed.  Exiting.
      exit 0
    fi

# Tablespace item, ts size 20M (20480K)
# each file 20M (20480K)
# extents 16892K (16892K)
# 1 files

$tpcc_createts item 1 1       20M 16892K unix 0       151 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for item failed.  Exiting.
      exit 0
    fi

# Tablespace ordr, ts size 710710M (727767040K)
# each file 64610M (66160640K)
# extents 103360K (103360K)
# 11 files

$tpcc_createts ordr 11 1       64610M 103360K unix 0       152 8 16K t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for ordr failed.  Exiting.
      exit 0
    fi

# Tablespace nord, ts size 6980M (7147520K)
# each file 6980M (7147520K)
# extents 714358K (714358K)
# 1 files

$tpcc_createts nord 1 1       6980M 714358K unix 0       163 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for nord failed.  Exiting.
      exit 0
    fi

# Tablespace iware, ts size 30M (30720K)
# each file 30M (30720K)
# extents 26024K (26024K)
# 1 files

$tpcc_createts iware 1 1       30M 26024K unix 0       164 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for iware failed.  Exiting.
      exit 0
    fi

# Tablespace icust1, ts size 16130M (16517120K)
# each file 16130M (16517120K)
# extents 226032K (226032K)
# 1 files

$tpcc_createts icust1 1 1       16130M 226032K unix 0       165 8 16K t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for icust1 failed.  Exiting.
      exit 0
    fi

# Tablespace icust2, ts size 35500M (36352000K)
# each file 7100M (7270400K)
# extents 113524K (113524K)
# 5 files

$tpcc_createts icust2 5 1       7100M 113524K unix 0       166 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for icust2 failed.  Exiting.
      exit 0
    fi

# Tablespace idist, ts size 110M (112640K)
# each file 110M (112640K)
# extents 101024K (101024K)
# 1 files

$tpcc_createts idist 1 1       110M 101024K unix 0       171 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for idist failed.  Exiting.
      exit 0
    fi

# Tablespace istok, ts size 43740M (44513280K)
# each file 43740M (42741760K)
# extents 667696K (667696K)
# 1 files

$tpcc_createts istok 1 1       43740M 667696K unix 0       172 8 16K t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for istok failed.  Exiting.
      exit 0
    fi

# Tablespace iitem, ts size 20M (20480K)
# each file 20M (20480K)
# extents 11264K (11264K)
# 1 files

$tpcc_createts iitem 1 1       20M 11264K unix 0       173 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for iitem failed.  Exiting.
      exit 0
    fi

# Tablespace iordr2, ts size 31080M (31825920K)
# each file 7770M (7956480K)
# extents 103280K (103280K)
# 4 files

$tpcc_createts iordr2 4 1       7770M 103280K unix 0       174 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for iordr2 failed.  Exiting.
      exit 0
    fi

# Tablespace temp, ts size 104780M (107294720K)
# each file 8060M (8253440K)
# extents 201150K (201150K)
# 13 files

$tpcc_createts temp 13 1       8060M 201150K unix 1       178 8 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for temp failed.  Exiting.
      exit 0
    fi
```

addfile.sh

```
#!/bin/sh
# $1 = tablespace name
# $2 = filename
# $3 = size
# $4 = temporary ts (1) or not (0)
# global variable $tpcc_listfiles, does not execute sql

if expr x$tpcc_listfiles = xt > /dev/null; then
  echo $2 $3 >> $tpcc_bench/files.dat
  exit 0
fi

if expr $4 = 1 > /dev/null; then
  altersql="alter tablespace $1 add tempfile '$2' size $3 reuse;"
else
  altersql="alter tablespace $1 add datafile '$2' size $3 reuse autoextend on;"
fi

$tpcc_sqlplus $tpcc_user_pass <<!
   spool addfile_$1.log
   set echo on
  $altersql
   set echo off
   spool off
   exit ;
!
```

addts.sh

```
#!/bin/sh
# $1 = tablespace name
# $2 = filename
# $3 = size
# $4 = uniform size
# $5 = block size
# $6 = temporary ts (1) or not (0)
# $7 = bitmapped manage (t) or not (f) or (d) for dictionary
# global variable $tpcc_listfiles, does not execute sql

if expr x$tpcc_listfiles = xt > /dev/null; then
  echo $2 $3 >> $tpcc_bench/files.dat
  exit 0
fi

if expr $5 = auto > /dev/null; then
  bssql=
else
  bssql="blocksize $5"
fi

if expr $6 = 1 > /dev/null; then
  createsql="create temporary tablespace $1 tempfile '$2' size $3 reuse extent
management local uniform size $4;"
else
  if expr x$7 = xt > /dev/null; then
    createsql="create tablespace $1 datafile '$2' size $3 reuse extent management local
uniform size $4 segment space management auto $bssql nologging ;"
  else
    if expr x$7 = xd > /dev/null; then
      createsql="create tablespace $1 datafile '$2' size $3 reuse extent management
dictionary nologging $bssql;"
    else
      createsql="create tablespace $1 datafile '$2' size $3 reuse extent management local
uniform size $4 segment space management manual $bssql nologging ;"
    fi
  fi
fi

$tpcc_sqlplus $tpcc_user_pass <<!
   spool createts_$1.log
   set echo on
   drop tablespace $1 including contents;
 $createsql
   set echo off
   spool off
   exit ;
!
```

assigntemp.sh

```
#!/bin/sh

echo Assigning temporary tablespace to user tpcc...
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_sql_dir/assigntemp > junk 2>&1
if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

driver.sh

```
#!/bin/sh

. ./stepenv.sh

if expr $# \< 1 > /dev/null; then
  echo "$0 <starting stepname> <optional: only>"
  echo OR use:
  echo "$0 buildcreate  - to build the database creation scripts"
  echo "$0 create       - to create the database (after buildcreate)"
  echo "$0 steps        - to list individual steps"
  exit 1
fi

if expr x$1 = xsteps > /dev/null; then
  echo stepnames are from creation scripts: $tpcc_create_steps
  echo
  echo or running steps: $tpcc_steps
  echo "use the 'only' option to only do that step (otherwise all steps after will also be
executed.)"
  echo "   (e.g.  $0 listfiles only)"
  echo "use the 'through' option to do a sequence of steps (inclusively.)"
  echo "   (e.g.  $0 shutdowndb through startupdb-p_build)"
  exit 1
fi

startstep=$1
controlcmd=$2
endstep=$3

# Aliases for special steps
if test $startstep = buildcreate; then
  startstep=`echo $tpcc_create_steps | cut -d' ' -f1`
fi

if test $startstep = create; then
  startstep=`echo $tpcc_steps | cut -d' ' -f1`
fi

if test "x$controlcmd" = x; then
  endstep=
  # Since endstep is null it won't match any other steps, so we keep going.
elif test "x$controlcmd" = xonly; then
  controlcmd=only
  # this is allowed
elif test "x$controlcmd" = xthrough; then
  actualstep=f
  for step in $tpcc_create_steps $tpcc_steps ; do
    if test "x$step" = "x$endstep"; then
      actualstep=t
    fi
  done
  if test $actualstep = f; then
    echo "Invalid step $endstep.  Use $0 steps  to show steps."
    exit 1
  fi
else
  echo "Invalid syntax.  Use $0 by itself for help."
  exit 1
fi

echo Starting from step: $startstep

dostep=f
for step in $tpcc_create_steps $tpcc_steps ; do
  if expr $step = $startstep > /dev/null; then
    dostep=t
  fi

  if expr $dostep = t > /dev/null; then
    echo STEP: $step
    cd $tpcc_bench
    $tpcc_scripts/`echo $step | cut -d- -f1`.sh `echo $step | sed -e's/-*$/-/' | cut -d- -f2- | sed -
e's/-/ /g'`
    lasterror=$?
    cd $tpcc_bench
    if test -n "`find $tpcc_bench/scripts -name '*.log`"; then
    mv -f *.log `find $tpcc_bench/scripts -name '*.log'` $tpcc_bench/log/
    else
      if test -n "`find $tpcc_bench/ -name '*.log`"; then
        mv -f *.log $tpcc_bench/log/
      fi
    fi

    if expr $lasterror != 0 > /dev/null; then
      if expr $lasterror != 99 > /dev/null; then
        echo Step $step failed.  Stopping driver.
        exit 1
      else
        echo Step $step has completed and requested stop.  Stopping driver.
        exit 0
      fi
    fi
    if test "x$controlcmd" = xonly; then
      exit 0
    fi
    if test "x$endstep" = "x$step"; then
      echo The driver reached the last desired step.  Stopping driver.
      exit 0
    fi
  fi
done

if expr $dostep = f > /dev/null; then
  echo No such step: $1
fi
```

loadware.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -w > loadware.log 2>&1
```

loaddist.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -d > loaddist.log 2>&1
```

loaditem.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -i > loaditem.log 2>&1
```

loadhist.sh

```
#created automatically by evenload.sh Fri Mar 2 10:57:55 CST 2007
rm -f loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 20000 -h  -b 1 -e 1250 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 1251 -e 2500 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 2501 -e 3750 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 3751 -e 5000 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 5001 -e 6250 >> loadhist4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 6251 -e 7500 >> loadhist5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 7501 -e 8750 >> loadhist6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 8751 -e 10000 >> loadhist7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 10001 -e 11250 >> loadhist8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 11251 -e 12500 >> loadhist9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 12501 -e 13750 >> loadhist10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 13751 -e 15000 >> loadhist11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 15001 -e 16250 >> loadhist12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 16251 -e 17500 >> loadhist13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 17501 -e 18750 >> loadhist14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -h  -b 18751 -e 20000 >> loadhist15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadnord.sh

```
#created automatically by evenload.sh Fri Mar 2 10:57:55 CST 2007
rm -f loadnord*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 20000 -n  -b 1 -e 1250 >> loadnord0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 1251 -e 2500 >> loadnord1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 2501 -e 3750 >> loadnord2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 3751 -e 5000 >> loadnord3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 5001 -e 6250 >> loadnord4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 6251 -e 7500 >> loadnord5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 7501 -e 8750 >> loadnord6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 8751 -e 10000 >> loadnord7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 10001 -e 11250 >> loadnord8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 11251 -e 12500 >> loadnord9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 12501 -e 13750 >> loadnord10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 13751 -e 15000 >> loadnord11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 15001 -e 16250 >> loadnord12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 16251 -e 17500 >> loadnord13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 17501 -e 18750 >> loadnord14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -n  -b 18751 -e 20000 >> loadnord15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadordrordl.sh

```
#created automatically by evenload.sh Fri Mar 2 10:57:55 CST 2007
rm -f loadordrordl*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy0.dat -b 1 -e 1250 >>
loadordrordl0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy1.dat -b 1251 -e 2500 >>
loadordrordl1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy2.dat -b 2501 -e 3750 >>
loadordrordl2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy3.dat -b 3751 -e 5000 >>
loadordrordl3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy4.dat -b 5001 -e 6250 >>
loadordrordl4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy5.dat -b 6251 -e 7500 >>
loadordrordl5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy6.dat -b 7501 -e 8750 >>
loadordrordl6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy7.dat -b 8751 -e 10000 >>
loadordrordl7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy8.dat -b 10001 -e 11250 >>
loadordrordl8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy9.dat -b 11251 -e 12500 >>
loadordrordl9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy10.dat -b 12501 -e 13750 >>
loadordrordl10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy11.dat -b 13751 -e 15000 >>
loadordrordl11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy12.dat -b 15001 -e 16250 >>
loadordrordl12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy13.dat -b 16251 -e 17500 >>
loadordrordl13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy14.dat -b 17501 -e 18750 >>
loadordrordl14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -o ${tpcc_disks_location}dummy15.dat -b 18751 -e 20000 >>
loadordrordl15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadcust.sh

```
#created automatically by evenload.sh Fri Mar 2 10:57:55 CST 2007
rm -f loadcust*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 20000 -C  -l 1 -m 187 >> loadcust0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 188 -m 374 >> loadcust1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 375 -m 561 >> loadcust2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 562 -m 748 >> loadcust3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 749 -m 935 >> loadcust4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 936 -m 1122 >> loadcust5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 1123 -m 1309 >> loadcust6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 1310 -m 1496 >> loadcust7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 1497 -m 1684 >> loadcust8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 1685 -m 1872 >> loadcust9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 1873 -m 2060 >> loadcust10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 2061 -m 2248 >> loadcust11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 2249 -m 2436 >> loadcust12.log 2>&1 &
```

```
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 2437 -m 2624 >> loadcust13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 2625 -m 2812 >> loadcust14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -C  -l 2813 -m 3000 >> loadcust15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadstok.sh

```
#created automatically by evenload.sh Fri Mar 2 10:57:55 CST 2007
rm -f loadstok*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 20000 -S  -j 1 -k 6250 >> loadstok0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 6251 -k 12500 >> loadstok1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 12501 -k 18750 >> loadstok2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 18751 -k 25000 >> loadstok3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 25001 -k 31250 >> loadstok4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 31251 -k 37500 >> loadstok5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 37501 -k 43750 >> loadstok6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 43751 -k 50000 >> loadstok7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 50001 -k 56250 >> loadstok8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 56251 -k 62500 >> loadstok9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 62501 -k 68750 >> loadstok10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 68751 -k 75000 >> loadstok11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 75001 -k 81250 >> loadstok12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 81251 -k 87500 >> loadstok13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 87501 -k 93750 >> loadstok14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 20000 -S  -j 93751 -k 100000 >> loadstok15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

11.2. SQL Creation Scripts

createdb.sql

```
/* created automatically by  /home/oracle/tpcc_20k/scripts/buildcreatedb.sh Fri Mar 2
10:57:47 CST 2007 */
spool createdb.log

set echo on

shutdown abort

startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxinstances 1
  datafile
   '/home/oracle/tpcc_20k/dev/system_1' size 400M reuse
  logfile '/home/oracle/tpcc_20k/dev/log_1_1' size 36621M reuse,
          '/home/oracle/tpcc_20k/dev/log_1_2' size 36621M reuse
  sysaux datafile '/home/oracle/tpcc_20k/dev/tpccaux' size 120M reuse ;


create undo tablespace undo_1 datafile
  '/home/oracle/tpcc_20k/dev/roll1' size 8096M reuse blocksize 8K;

set echo off
exit sql.sqlcode
```

createuser.sql

```
spool createusertpcc.log;

set echo on;

create user tpcc identified by tpcc;

grant dba to tpcc;

set echo off;
spool off;

exit ;
```

assigntemp.sql

```
spool assigntemp.log;

set echo on;

alter user tpcc temporary tablespace temp_0;

set echo off;
spool off;

exit ;
```

createtable_ware.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreatetable.sh Fri Mar 2
10:57:48 CST 2007 */
set timing on
    set sqlblanklines on
    spool createtable_ware.log
    set echo on
      drop cluster warecluster including tables ;

create cluster warecluster (
  w_id number
  )
  single table
  hashkeys 20000
  hash is ( (w_id - 1) )
  size 1448
    initrans 2
  storage ( buffer_pool default )
  tablespace ware_0;

create table ware (
  w_id number
, w_ytd number
, w_tax number
, 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)
)
cluster warecluster (
  w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_cust.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreatetable.sh Fri Mar 2
10:57:49 CST 2007 */
set timing on
    set sqlblanklines on
    spool createtable_cust.log
    set echo on
      drop cluster custcluster including tables ;

create cluster custcluster (
  c_id number
, c_d_id number
, c_w_id number
  )
  single table
  hashkeys 600000000
  hash is ( (c_id * ( 20000 * 10 ) + c_w_id * 10 + c_d_id) )
  size 180
  pctfree 0  initrans 3
  storage ( buffer_pool recycle ) parallel ( degree 8 )
  tablespace cust_0;

create table cust (
  c_id number
, c_d_id number
, c_w_id number
, c_discount number
, c_credit char(2)
, c_last varchar2(16)
, c_first varchar2(16)
, c_credit_lim number
, c_balance number
, c_ytd_payment number
, c_payment_cnt number
, c_delivery_cnt number
, 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_middle char(2)
, c_data char(500)
)
cluster custcluster (
  c_id
, c_d_id
, c_w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_dist.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreatetable.sh Fri Mar 2
10:57:50 CST 2007 */
set timing on
    set sqlblanklines on
    spool createtable_dist.log
    set echo on
      drop cluster distcluster including tables ;

create cluster distcluster (
  d_id number
, d_w_id number
  )
  single table
  hashkeys 200000
  hash is ( ((d_w_id * 10) + d_id) )
  size 1448
    initrans 4
  storage ( buffer_pool default )
  tablespace dist_0;

create table dist (
  d_id number
, d_w_id number
, d_ytd number
, d_next_o_id number
, d_tax 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)
)
cluster distcluster (
  d_id
, d_w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_hist.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreatetable.sh Fri Mar 2
10:57:51 CST 2007 */
set timing on
    set sqlblanklines on
    spool createtable_hist.log
    set echo on
      drop table hist ;

create table hist (
  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
, h_data varchar2(24)
)
  pctfree 5  initrans 4
  storage ( buffer_pool recycle )
  tablespace hist_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_stok.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreatetable.sh Fri Mar 2
10:57:51 CST 2007 */
set timing on
    set sqlblanklines on
    spool createtable_stok.log
    set echo on
      drop cluster stokcluster including tables ;

create cluster stokcluster (
  s_i_id number
, s_w_id number
  )
  single table
  hashkeys 2000000000
  hash is ( (s_i_id * 20000 + s_w_id) )
  size 256
  pctfree 0  initrans 2 maxtrans 2
  storage ( buffer_pool keep ) parallel ( degree 8 )
  tablespace stok_0;

create table stok (
  s_i_id number
, s_w_id number
, s_quantity number
, s_ytd number
, s_order_cnt number
, s_remote_cnt number
, s_data varchar2(50)
, s_dist_01 char(24)
, s_dist_02 char(24)
, s_dist_03 char(24)
, s_dist_04 char(24)
, s_dist_05 char(24)
, s_dist_06 char(24)
, s_dist_07 char(24)
, s_dist_08 char(24)
, s_dist_09 char(24)
, s_dist_10 char(24)
)
cluster stokcluster (
  s_i_id
, s_w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_item.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreatetable.sh Fri Mar 2
10:57:52 CST 2007 */
set timing on
    set sqlblanklines on
    spool createtable_item.log
    set echo on
      drop cluster itemcluster including tables ;

create cluster itemcluster (
  i_id number(6,0)
  )
  single table
  hashkeys 100000
  hash is ( (i_id) )
  size 120
  pctfree 0  initrans 3
  storage ( buffer_pool keep )
  tablespace item_0;

create table item (
  i_id number(6,0)
, i_name varchar2(24)
, i_price number
, i_data varchar2(50)
, i_im_id number
)
cluster itemcluster (
  i_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_ordl.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreatetable.sh Fri Mar 2
10:57:54 CST 2007 */
set timing on
    set sqlblanklines on
    spool createtable_ordl.log
    set echo on
        create table ordl (
    ol_w_id number
, ol_d_id number
, ol_o_id number sort
, ol_number number sort
, ol_i_id number
, ol_delivery_d date
, ol_amount number
, ol_supply_w_id number
, ol_quantity number
, ol_dist_info char(24)
  , constraint ordl_uk primary key (ol_w_id, ol_d_id, ol_o_id, ol_number  )) CLUSTER
ordrcluster_queue(ol_w_id, ol_d_id, ol_o_id, ol_number) ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_nord.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreatetable.sh Fri Mar 2
10:57:54 CST 2007 */
set timing on
    set sqlblanklines on
    spool createtable_nord.log
    set echo on
       drop cluster nordcluster_queue including tables ;

  create cluster nordcluster_queue (
    no_w_id number
, no_d_id number
, no_o_id number SORT
    )

    hashkeys 200000
    hash is ( (no_w_id - 1) * 10 + no_d_id - 1 )
    size 190
    tablespace nord_0;

  create table nord (
    no_w_id number
, no_d_id number
, no_o_id number sort
    , constraint nord_uk primary key ( no_w_id
, no_d_id
, no_o_id )
  )
  cluster nordcluster_queue (
    no_w_id
, no_d_id
, no_o_id
  );
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_iware.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreateindex.sh Fri Mar 2
10:57:56 CST 2007 */
set timing on
    set sqlblanklines on
    spool createindex_iware.log ;
    set echo on ;
    drop index iware ;
      create unique index iware on ware ( w_id )
  pctfree 1  initrans 3
  storage ( buffer_pool default )
  parallel 1
  compute statistics
  tablespace iware_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_icust1.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreateindex.sh Fri Mar 2
10:57:56 CST 2007 */
set timing on
    set sqlblanklines on
    spool createindex_icust1.log ;
    set echo on ;
    drop index icust1 ;
      create unique index icust1 on cust ( c_w_id
, c_d_id
, c_id )
  pctfree 1  initrans 3
  storage ( buffer_pool default )
  parallel 32
  compute statistics
  tablespace icust1_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_icust2.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreateindex.sh Fri Mar 2
10:57:56 CST 2007 */
set timing on
    set sqlblanklines on
    spool createindex_icust2.log ;
    set echo on ;
    drop index icust2 ;
      create unique index icust2 on cust ( c_last
, c_w_id
, c_d_id
, c_first
, c_id )
  pctfree 1  initrans 3
  storage ( buffer_pool default )
  compute statistics
  tablespace icust2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_idist.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreateindex.sh Fri Mar 2
10:57:57 CST 2007 */
set timing on
    set sqlblanklines on
    spool createindex_idist.log ;
    set echo on ;
    drop index idist ;
      create unique index idist on dist ( d_w_id
, d_id )
  pctfree 5  initrans 3
  storage ( buffer_pool default )
  parallel 1
  compute statistics
  tablespace idist_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_istok.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreateindex.sh Fri Mar 2
10:57:57 CST 2007 */
set timing on
    set sqlblanklines on
    spool createindex_istok.log ;
    set echo on ;
    drop index istok ;
      create unique index istok on stok ( s_i_id
, s_w_id )
  pctfree 1  initrans 3
  storage ( buffer_pool default )
  parallel 32
  compute statistics
  tablespace istok_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_iitem.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreateindex.sh Fri Mar 2
10:57:58 CST 2007 */
set timing on
    set sqlblanklines on
    spool createindex_iitem.log ;
    set echo on ;
    drop index iitem ;
      create unique index iitem on item ( i_id )
  pctfree 5  initrans 4
  storage ( buffer_pool default )

  compute statistics
  tablespace iitem_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_iordr2.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreateindex.sh Fri Mar 2
10:57:58 CST 2007 */
set timing on
    set sqlblanklines on
    spool createindex_iordr2.log ;
    set echo on ;
    drop index iordr2 ;
      create unique index iordr2 on ordr ( o_c_id
, o_d_id
, o_w_id
, o_id )
  pctfree 25  initrans 4
  storage ( buffer_pool default )
  parallel 32
  compute statistics
  tablespace iordr2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_ordr.sql

```
/* created automatically by /home/oracle/tpcc_20k/scripts/buildcreatetable.sh Fri Mar 2
10:57:53 CST 2007 */
set timing on
    set sqlblanklines on
    spool createtable_ordr.log
    set echo on
       drop cluster ordrcluster_queue including tables ;

  create cluster ordrcluster_queue (
```

```
    o_w_id number
, o_d_id number
, o_id number SORT
, o_number number SORT
    )

    hashkeys 200000
    hash is ( (o_w_id - 1) * 10 + o_d_id - 1 )
    size 1490
    tablespace ordr_0;

  create table ordr (
    o_id number sort
, o_w_id number
, o_d_id number
, o_c_id number
, o_carrier_id number
, o_ol_cnt number
, o_all_local number
, o_entry_d date
     , constraint ordr_uk primary key ( o_w_id
, o_d_id
, o_id )
  )
  cluster ordrcluster_queue (
    o_w_id
, o_d_id
, o_id
  );
    set echo off
    spool off
    exit sql.sqlcode;
```

analyze.sql

```
spool analyze.log;
set echo on;

connect tpcc/tpcc

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                TABNAME=>'STOK', -
                                PARTNAME=>NULL, -
                                ESTIMATE_PERCENT=>1, -
                                BLOCK_SAMPLE=>TRUE, -
                                METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                DEGREE=>10, -
                                GRANULARITY=>'DEFAULT', -
                                CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                TABNAME=>'CUST', -
                                PARTNAME=>NULL, -
                                ESTIMATE_PERCENT=>1, -
                                BLOCK_SAMPLE=>TRUE, -
                                METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                DEGREE=>10, -
                                GRANULARITY=>'DEFAULT', -
                                CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                TABNAME=>'ORDR', -
                                PARTNAME=>NULL, -
                                ESTIMATE_PERCENT=>1, -
                                BLOCK_SAMPLE=>TRUE, -
                                METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                DEGREE=>10, -
                                GRANULARITY=>'DEFAULT', -
                                CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                TABNAME=>'ORDL', -
                                PARTNAME=>NULL, -
                                ESTIMATE_PERCENT=>1, -
                                BLOCK_SAMPLE=>TRUE, -
                                METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                DEGREE=>10, -
                                GRANULARITY=>'DEFAULT', -
                                CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS (OWNNAME=>'TPCC', -
                                TABNAME=>'NORD', -
                                PARTNAME=>NULL, -
                                ESTIMATE_PERCENT=>1, -
                                BLOCK_SAMPLE=>TRUE, -
                                METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                DEGREE=>10, -
                                GRANULARITY=>'DEFAULT', -
                                CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
                                TABNAME=>'HIST', -
                                PARTNAME=>NULL, -
                                ESTIMATE_PERCENT=>1, -
                                BLOCK_SAMPLE=>TRUE, -
                                METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                DEGREE=>10, -
                                GRANULARITY=>'DEFAULT', -
                                CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
                                TABNAME=>'DIST', -
                                PARTNAME=>NULL, -
                                ESTIMATE_PERCENT=>1, -
                                BLOCK_SAMPLE=>TRUE, -
                                METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                DEGREE=>10, -
                                GRANULARITY=>'DEFAULT', -
                                CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
                                TABNAME=>'ITEM', -
                                PARTNAME=>NULL, -
                                ESTIMATE_PERCENT=>10, -
                                BLOCK_SAMPLE=>TRUE, -
                                METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                DEGREE=>1, -
                                GRANULARITY=>'DEFAULT', -
                                CASCADE=>TRUE);

execute dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
                                TABNAME=>'WARE', -
                                PARTNAME=>NULL, -
                                ESTIMATE_PERCENT=>10, -
                                BLOCK_SAMPLE=>TRUE, -
                                METHOD_OPT=>'FOR ALL COLUMNS SIZE 1', -
                                DEGREE=>10, -
                                GRANULARITY=>'DEFAULT', -
                                CASCADE=>TRUE);



set echo off;
spool off;

exit sql.sqlcode;
```

createspacestats.sql

```
@space_init
@space_get 235063 20000
@space_rpt
spool off
exit sql.sqlcode;
```

createstoredprocs.sql

```
spool createstoreprocs.log
@tkvcinin.sql
spool off
exit sql.sqlcode;
```

createmisc.sh

```
#!/bin/sh

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool createmisc.log
set echo on;
alter user tpcc temporary tablespace system;
grant execute on dbms_lock to public;
grant execute on dbms_pipe to public;
grant select on v_\$parameter to public;

REM
REM begin plsql_mon.sql
REM

connect tpcc/tpcc;
set echo on;
CREATE OR REPLACE PACKAGE plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  );
END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  )
  IS
    s           NUMBER;
  BEGIN
    dbms_pipe.pack_message (info);
    s := dbms_pipe.send_message ('plsql_mon');
    IF (s <> 0) THEN
      raise_application_error (-20000, 'Error:' || to_char(s) ||
                               ' sending on pipe');
    END IF;
  END;
END;
/
show errors;
```

```
set echo off;

REM
REM end plsql_mon.sql
REM

REM
REM  begin cre_tab.sql
REM

connect tpcc/tpcc;
set echo on;

drop table temp_o1;
drop table temp_no;
drop table temp_o2;
drop table temp_ol;
drop table tpcc_audit_tab;

create table temp_o1 (
   o_w_id  integer,
   o_d_id  integer,
   o_o_id  integer);

create table temp_no (
   no_w_id integer,
   no_d_id integer,
   no_o_id integer);

create table temp_o2 (
   o_w_id  integer,
   o_d_id  integer,
   o_count integer);

create table temp_ol (
   ol_w_id  integer,
   ol_d_id  integer,
   ol_count integer);

create table tpcc_audit_tab (starttime date);

delete from tpcc_audit_tab;

set echo off;

REM
REM  end cre_tab.sql
REM

REM
REM  begin views.sql
REM

connect tpcc/tpcc;
set echo on;

create or replace view wh_cust
(w_id, w_tax, c_id, c_d_id, c_w_id, c_discount, c_last, c_credit)
as select w.w_id, w.w_tax,
          c.c_id, c.c_d_id, c.c_w_id, c.c_discount, c.c_last, c.c_credit
   from cust c, ware w
   where w.w_id = c.c_w_id;

create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
   from dist d, ware w
   where w.w_id = d.d_w_id;

create  or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
 select i.i_id, s_w_id, i.i_price, i.i_name, i.i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
  from stok s, item i
 where i.i_id = s.s_i_id;

set echo off;

REM
REM  end views.sql
REM


REM
REM  begin dml.sql
REM
connect tpcc/tpcc;
set echo on;

   alter table ware disable table lock;
   alter table dist disable table lock;
   alter table cust disable table lock;
   alter table hist disable table lock;
   alter table item disable table lock;
   alter table stok disable table lock;
   alter table ordr disable table lock;
   alter table nord disable table lock;
   alter table ordl disable table lock;

set echo off;

REM
REM  end dml.sql
REM

REM
REM  begin extent.sql
REM

$SYS_CONNECTION_STRING

@$tpcc_sql_dir/extent

@$tpcc_sql_dir/freeext

exit sql.sqlcode;

!
```

extent.sql

```
REM============================================================
=+
REM           Copyright (c) 1994  Oracle Corp, Belmont, CA         |
REM                OPEN SYSTEMS PERFORMANCE GROUP                  |
REM                    All Rights Reserved                         |
REM============================================================
=+
REM FILENAME
REM       extent.sql
REM DESCRIPTION
REM       List all extents in all the TPCC tablespaces.
REM
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @extent
REM============================================================
=*/
     set space 2
     set pagesize 2000
     set echo off
     set termout off
     set verify off
     set feedback off
     spool extent.rpt
     select substr(e.tablespace_name,1,8) tspace,
       substr(segment_name,1,11) segment, substr(segment_type,1,15) type,
       substr(extent_id,1,5) eid, substr(file_id,1,5) fid, blocks,
       blocks * t.block_size / 1048576 size_MB
     from   dba_extents e, dba_tablespaces t
     where  owner = 'TPCC' AND ( segment_type = 'INDEX' OR
       segment_type = 'INDEX PARTITION' OR segment_type = 'CLUSTER'
       OR segment_type = 'TABLE' OR segment_type = 'TABLE PARTITION')
       AND e.tablespace_name <> 'SYSTEM'
       AND e.tablespace_name = t.tablespace_name
     order by e.tablespace_name, segment_name, extent_id, file_id;

     select substr(e.tablespace_name,1,8) tspace,
       substr(segment_name,1,11) segment,
       sum(blocks) tot_blk, sum(blocks) * t.block_size / 1048576 size_MB
     from   dba_extents e, dba_tablespaces t
     where  owner = 'TPCC' AND ( segment_type = 'INDEX' OR
       segment_type = 'INDEX PARTITION' OR segment_type = 'CLUSTER'
       OR segment_type = 'TABLE' OR segment_type = 'TABLE PARTITION')
       AND e.tablespace_name <> 'SYSTEM'
       AND e.tablespace_name = t.tablespace_name
     group by e.tablespace_name, segment_name, t.block_size
     order by e.tablespace_name, segment_name;
     spool off;
```

freeext.sql

```
REM============================================================
=+
REM           Copyright (c) 1994  Oracle Corp, Belmont, CA         |
REM                OPEN SYSTEMS PERFORMANCE GROUP                  |
REM                    All Rights Reserved                         |
REM============================================================
=+
REM FILENAME
REM       freeext.sql
REM DESCRIPTION
REM       List all free extents in all the TPCC tablespace
REM
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @freeext
REM============================================================
=*/
     set space 2
     set pagesize 2000
     set echo off
     set termout off
     set verify off
     set feedback off
     spool freeextent.rpt
     select substr(e.tablespace_name,1,8) tspace, file_id, block_id, blocks,
          blocks * t.block_size / 1048576 size_MB
     from dba_free_space e, dba_tablespaces t
     where e.tablespace_name = t.tablespace_name
     order by e.tablespace_name, file_id, block_id;

     select substr(e.tablespace_name,1,8) tspace, sum(blocks) tot_blk,
          sum(blocks) * t.block_size / 1048576 size_MB
     from dba_free_space e, dba_tablespaces t
     where e.tablespace_name = t.tablespace_name
     group by e.tablespace_name, t.block_size
```

```
    order by e.tablespace_name;
```

post_restore.sql

```
alter table ordr enable table lock;
alter index iordr2 noparallel;
alter table ordr disable table lock;

alter tablespace dist_0 add datafile '/home/oracle/tpcc_20k/dev/dist_0_1' size 428M reuse
autoextend on;
alter tablespace icust1_0  add datafile '/home/oracle/tpcc_20k/dev/icust1_0_1' size 340M
reuse autoextend on;
alter tablespace istok_0 add datafile '/home/oracle/tpcc_20k/dev/istok_0_1' size 668M
reuse autoextend on;

alter user tpcc temporary tablespace system;
alter user tpcc default tablespace system;

commit;

exit
```

11.3. 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 wares> [options]
 |            options: -A  load all tables
 |                     -w  load ware table
 |                     -d  load dist table
 |                     -c  load cust table (cluster around c_w_id)
 |                     -C  load cust table (cluster around c_id)
 |                     -i  load item table
 |                     -s  load stok table (cluster around s_w_id)
 |                     -S  load stok table (cluster around s_i_id)
 |                     -h  load hist table
 |                     -n  load new-order table
 |                     -o <oline file> load order and order-line table
 |                     -b <ware#>  beginning ware number
 |                     -e <ware#>  ending ware number
 |                     -j <item#>  beginning item number (with -S)
 |                     -k <item#>  ending item number (with -S)
 |                     -l <cid#>  beginning cid number (with -C)
 |                     -m <cid#>  ending cid number (with -C)
 |                     -g  generate rows to standard output
 +=================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include "tpcc.h"

#ifdef ORA_NT
#undef boolean
#include <process.h>
#include "dpbcore.h"
# define gettime dpbtimef
# define getcpu  dpbcpu
#define lrand48() ((long)rand() <<15 | rand())
#ifdef __STDC__
#  define PROTO(args)    args
#else
#  define PROTO(args)    ()
#endif
#endif

#define DISTARR 10     /* dist insert array size   */
#define CUSTARR 100      /* cust insert array size  */
#define STOCARR 100      /* stok insert array size  */
#define ITEMARR 100      /* item insert array size  */
#define HISTARR 100            /* hist insert array size    */
#define ORDEARR 100              /* order insert array size      */
#define NEWOARR 100              /* new order insert array size  */

#define DISTFAC 10      /* max. dist id    */
#define CUSTFAC 3000    /* max. cust id   */
#define STOCFAC 100000    /* max. stok 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 NOT_SERIALIZABLE  8177  /* ORA-08177: transaction not serializable */
#define SNAPSHOT_TOO_OLD  1555  /* ORA-01555: snapshot too old */
#define RECOVERR -10
#define IRRECERR -20

#define SQLTXTW "INSERT INTO ware (w_id, w_ytd, w_tax, w_name, w_street_1,
w_street_2, w_city, w_state, w_zip) VALUES (:w_id,
30000000, :w_tax, :w_name, :w_street_1, \
   :w_street_2, :w_city, :w_state, :w_zip)"

#define SQLTXTD "INSERT INTO dist (d_id, d_w_id, d_ytd, d_tax, d_next_o_id, d_name,
d_street_1, d_street_2, d_city, d_state, d_zip) 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 SQLTXTCQUERY "select /*+ HASH ( cust ) */ count(*) from cust where c_w_id
= :s_c_w_id and c_d_id = :s_c_d_id and c_id = :s_c_id"

#define SQLTXTC "INSERT INTO cust (C_ID, C_D_ID, C_W_ID, C_FIRST, C_MIDDLE,
C_LAST, 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_YTD_PAYMENT,
C_PAYMENT_CNT, C_DELIVERY_CNT, C_DATA) 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 hist (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
h_date, h_amount, h_data) VALUES (:h_c_id, :h_c_d_id, :h_c_w_id, \
   :h_d_id, :h_w_id, SYSDATE, 1000, :h_data)"

#define SQLTXTSQUERY "select /*+ HASH ( stok ) */ count(*) from stok where s_w_id
= :s_s_w_id and s_i_id = :s_s_i_id "

#define SQLTXTS "INSERT INTO stok (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,
s_ytd, s_order_cnt, s_remote_cnt, s_data) \
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 (I_ID,I_IM_ID,I_NAME,I_PRICE,I_DATA) VALUES
(:i_id, :i_im_id, :i_name, :i_price, \
   :i_data)"

#define SQLTXTO1 "INSERT INTO ordr (O_ID,
O_D_ID,O_W_ID,O_C_ID,O_ENTRY_D,O_CARRIER_ID,O_OL_CNT,O_ALL_LOCAL) \
   VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
   SYSDATE, :o_carrier_id, :o_ol_cnt, 1)"

#define SQLTXTO2 "INSERT INTO ordr (O_ID,
O_D_ID,O_W_ID,O_C_ID,O_ENTRY_D,O_CARRIER_ID,O_OL_CNT,O_ALL_LOCAL) \
   VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
   SYSDATE, 11, :o_ol_cnt, 1)"

#define SQLTXTOL1 "INSERT INTO ordl (OL_O_ID, OL_D_ID, OL_W_ID, OL_NUMBER,
OL_DELIVERY_D, OL_I_ID, OL_SUPPLY_W_ID, OL_QUANTITY, OL_AMOUNT,
OL_DIST_INFO) \
   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 ordl (OL_O_ID, OL_D_ID, OL_W_ID, OL_NUMBER,
OL_DELIVERY_D, OL_I_ID, OL_SUPPLY_W_ID, OL_QUANTITY, OL_AMOUNT,
OL_DIST_INFO) \
   VALUES (:ol_o_id, :ol_d_id, \
   :ol_w_id, :ol_number, to_date('01-Jan-1811'), :ol_i_id, :ol_supply_w_id, 5, :ol_amount, \
   :ol_dist_info)"

#define SQLTXTNO "INSERT INTO nord (no_o_id, no_d_id, no_w_id) VALUES
(:no_o_id, :no_d_id, :no_w_id)"

#define SQLTXTENHA "alter session set \"_enable_hash_overflow\"=true"
#define SQLTXTDIHA "alter session set \"_enable_hash_overflow\"=false"

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];

void initperm();
void randstr();
void randdatastr();
void randnum();
void randlastname (char*, int);
int NURand();
```

```
void sysdate();

OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;

OCIStmt *curw;
OCIStmt *curd;
OCIStmt *curc;
OCIStmt *curcs;
OCIStmt *curh;
OCIStmt *curs;
OCIStmt *curss;
OCIStmt *curi;
OCIStmt *curo1;
OCIStmt *curo2;
OCIStmt *curol1;
OCIStmt *curol2;
OCIStmt *curno;

OCIBind *w_id_bp = (OCIBind *) 0;
OCIBind *w_name_bp = (OCIBind *) 0;
OCIBind *w_street1_bp = (OCIBind *) 0;
OCIBind *w_street2_bp = (OCIBind *) 0;
OCIBind *w_city_bp = (OCIBind *) 0;
OCIBind *w_state_bp = (OCIBind *) 0;
OCIBind *w_zip_bp = (OCIBind *) 0;
OCIBind *w_tax_bp = (OCIBind *) 0;

OCIBind *d_id_bp = (OCIBind *) 0;
OCIBind *d_w_id_bp = (OCIBind *) 0;
OCIBind *d_name_bp = (OCIBind *) 0;
OCIBind *d_street1_bp = (OCIBind *) 0;
OCIBind *d_street2_bp = (OCIBind *) 0;
OCIBind *d_city_bp = (OCIBind *) 0;
OCIBind *d_state_bp = (OCIBind *) 0;
OCIBind *d_zip_bp = (OCIBind *) 0;
OCIBind *d_tax_bp = (OCIBind *) 0;

OCIDefine *s_c_ret_bp = (OCIDefine *) 0;
OCIBind *s_c_id_bp = (OCIBind *) 0;
OCIBind *s_c_d_id_bp = (OCIBind *) 0;
OCIBind *s_c_w_id_bp = (OCIBind *) 0;

OCIBind *c_id_bp = (OCIBind *) 0;
OCIBind *c_d_id_bp = (OCIBind *) 0;
OCIBind *c_w_id_bp = (OCIBind *) 0;
OCIBind *c_first_bp = (OCIBind *) 0;
OCIBind *c_last_bp = (OCIBind *) 0;
OCIBind *c_street1_bp = (OCIBind *) 0;
OCIBind *c_street2_bp = (OCIBind *) 0;
OCIBind *c_city_bp = (OCIBind *) 0;
OCIBind *c_state_bp = (OCIBind *) 0;
OCIBind *c_zip_bp = (OCIBind *) 0;
OCIBind *c_phone_bp = (OCIBind *) 0;
OCIBind *c_discount_bp = (OCIBind *) 0;
OCIBind *c_credit_bp = (OCIBind *) 0;
OCIBind *c_data_bp = (OCIBind *) 0;

OCIBind *i_id_bp = (OCIBind *) 0;
OCIBind *i_im_id_bp = (OCIBind *) 0;
OCIBind *i_name_bp = (OCIBind *) 0;
OCIBind *i_price_bp = (OCIBind *) 0;
OCIBind *i_data_bp = (OCIBind *) 0;

OCIDefine *s_s_ret_bp = (OCIDefine *) 0;
OCIBind *s_s_i_id_bp = (OCIBind *) 0;
OCIBind *s_s_w_id_bp = (OCIBind *) 0;

OCIBind *s_i_id_bp = (OCIBind *) 0;
OCIBind *s_w_id_bp = (OCIBind *) 0;
OCIBind *s_quantity_bp = (OCIBind *) 0;
OCIBind *s_dist_01_bp = (OCIBind *) 0;
OCIBind *s_dist_02_bp = (OCIBind *) 0;
OCIBind *s_dist_03_bp = (OCIBind *) 0;
OCIBind *s_dist_04_bp = (OCIBind *) 0;
OCIBind *s_dist_05_bp = (OCIBind *) 0;
OCIBind *s_dist_06_bp = (OCIBind *) 0;
OCIBind *s_dist_07_bp = (OCIBind *) 0;
OCIBind *s_dist_08_bp = (OCIBind *) 0;
OCIBind *s_dist_09_bp = (OCIBind *) 0;
OCIBind *s_dist_10_bp = (OCIBind *) 0;
OCIBind *s_data_bp = (OCIBind *) 0;

OCIBind *h_c_id_bp = (OCIBind *) 0;
OCIBind *h_c_d_id_bp = (OCIBind *) 0;
OCIBind *h_c_w_id_bp = (OCIBind *) 0;
OCIBind *h_d_id_bp = (OCIBind *) 0;
OCIBind *h_w_id_bp = (OCIBind *) 0;
OCIBind *h_data_bp = (OCIBind *) 0;

OCIBind *ol_o_id_bp = (OCIBind *) 0;
OCIBind *ol_d_id_bp = (OCIBind *) 0;
OCIBind *ol_w_id_bp = (OCIBind *) 0;
OCIBind *ol_i_id_bp = (OCIBind *) 0;
OCIBind *ol_number_bp = (OCIBind *) 0;
OCIBind *ol_supply_w_id_bp = (OCIBind *) 0;
OCIBind *ol_dist_info_bp = (OCIBind *) 0;
OCIBind *ol_amount_bp = (OCIBind *) 0;

OCIBind *o_id_bp = (OCIBind *) 0;
OCIBind *o_d_id_bp = (OCIBind *) 0;
OCIBind *o_w_id_bp = (OCIBind *) 0;
OCIBind *o_c_id_bp = (OCIBind *) 0;
OCIBind *o_carrier_id_bp = (OCIBind *) 0;
OCIBind *o_ol_cnt_bp  = (OCIBind *) 0;
OCIBind *o_ocnt_bp = (OCIBind *) 0;
OCIBind *o_olcnt_bp  = (OCIBind *) 0;


OCIBind *no_o_id_bp  = (OCIBind *) 0;
OCIBind *no_d_id_bp  = (OCIBind *) 0;
OCIBind *no_w_id_bp  = (OCIBind *) 0;

void 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 ware table\n");
   fprintf (stderr, "\t-d :\tload dist table\n");
   fprintf (stderr, "\t-c :\tload cust table (cluster around c_w_id\n");
   fprintf (stderr, "\t-C :\tload cust table (cluster around c_id\n");
   fprintf (stderr, "\t-i :\tload item table\n");
   fprintf (stderr, "\t-s :\tload stok table (cluster around s_w_id)\n");
   fprintf (stderr, "\t-S :\tload stok table (cluster around s_i_id)\n");
   fprintf (stderr, "\t-h :\tload hist 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 ware number\n");
   fprintf (stderr, "\t-e <ware#> :\tending ware 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-l <cid#> :\tbeginning cid number (with -C)\n");
   fprintf (stderr, "\t-m <cid#> :\tending cid number (with -C)\n");
   fprintf (stderr, "\t-g :\tgenerate rows to standard output\n");
   fprintf (stderr,"\t   $tpcc_bench must be set to the location of the kit\n");
   fprintf (stderr, "\n");
   exit(1);
}

int sqlfile(fnam,linebuf)
char   *fnam;
text   *linebuf;
{
    FILE *fd;
    int nulpt = 0;
    char realfile[512];

    sprintf(realfile,"%s",fnam);
    fd = fopen(realfile,"r");
    if (!fd)
    {
      return (0);
    }
    while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE, fd))
    {
       nulpt = strlen((char *)linebuf);
    }
    return(nulpt);
}

void quit()
{
   OCIERROR(errhp,OCISessionEnd ( tpcsvc,errhp, tpcusr, OCI_DEFAULT));
   OCIERROR(errhp,OCIServerDetach ( tpcsrv, errhp, OCI_DEFAULT));
   OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
   OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
   OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
   OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
   OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);
}

void main (argc, argv)
int argc;
char *argv[];
{
   char *uid="tpcc";
   char *pwd="tpcc";
   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];
   float 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];
float d_tax[10];

int s_c_id;
int s_c_d_id;
int s_c_w_id;
int s_c_count;

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];
float 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_s_count;
int s_s_i_id;
int s_s_w_id;

int s_i_id[100];
int s_w_id[100];
int s_quantity[100];
char s_dist_01[100][25];
char s_dist_02[100][25];
char s_dist_03[100][25];
char s_dist_04[100][25];
char s_dist_05[100][25];
char s_dist_06[100][25];
char s_dist_07[100][25];
char s_dist_08[100][25];
char s_dist_09[100][25];
char s_dist_10[100][25];
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[1500];
int ol_d_id[1500];
int ol_w_id[1500];
int ol_number[1500];
int ol_i_id[1500];
int ol_supply_w_id[1500];
int ol_amount[1500];
char ol_dist_info[1500][24];
```

```
  int o_cnt;
  int ol_cnt;

  ub2 ol_o_id_len[1500];
  ub2 ol_d_id_len[1500];
  ub2 ol_w_id_len[1500];
  ub2 ol_number_len[1500];
  ub2 ol_i_id_len[1500];
  ub2 ol_supply_w_id_len[1500];
  ub2 ol_dist_info_len[1500];
  ub2 ol_amount_len[1500];

  ub4 ol_o_id_clen;
  ub4 ol_d_id_clen;
  ub4 ol_w_id_clen;
  ub4 ol_number_clen;
  ub4 ol_i_id_clen;
  ub4 ol_supply_w_id_clen;
  ub4 ol_dist_info_clen;
  ub4 ol_amount_clen;

  ub2 o_id_len[100];
  ub2 o_d_id_len[100];
  ub2 o_w_id_len[100];
  ub2 o_c_id_len[100];
  ub2 o_carrier_id_len[100];
  ub2 o_ol_cnt_len[100];

  ub4 o_id_clen;
  ub4 o_d_id_clen;
  ub4 o_w_id_clen;
  ub4 o_c_id_clen;
  ub4 o_carrier_id_clen;
  ub4 o_ol_cnt_clen;

  text stmbuf[16*1024];

  int no_o_id[100];
  int no_d_id[100];
  int no_w_id[100];

  char sdate[30];

#ifdef ORA_NT
  clock_t begin_time, end_time;
  clock_t begin_cpu, end_cpu;

  char *arg_ptr,  **end_args;
#else
  double begin_time, end_time;
  double begin_cpu, end_cpu;
  double gettime(), getcpu();

  extern int getopt();
  extern char *optarg;
  extern int optind, opterr;
  int opt;
#endif

  char   *argstr="M:AwdcCisShno:b:e:j:k:l:m:g";
  int do_A=0;
  int do_w=0;
  int do_d=0;
  int do_i=0;
  int do_c=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;
  int bcid=1;
  int ecid=0;

  FILE *olfp=NULL;
  char olfname[100];
  char* basename;
  int status;
#ifdef ORA_NT
  char fname[100];
  FILE *logfile;
#endif /* ORA_NT */

/*---------------------------------------------------------------+
 | Parse command line -- look for scale factor.          |
 +---------------------------------------------------------------*/

  if (argc == 1) {
    myusage ();
  }

#ifdef ORA_NT
  end_args = argv + argc;
  for (++argv; argv < end_args; )
  {
    arg_ptr = *argv++;

      if (*arg_ptr != '-')
      {
        myusage ();
      } else
      {
      switch (arg_ptr[1]) {
      case '?': myusage ();
                break;
      case 'M': scale = atoi (*argv++);
                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 '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, *argv++);
                break;
      case 'b': bware = atoi (*argv++);
                break;
      case 'e': eware = atoi (*argv++);
                break;
      case 'j': bitem = atoi (*argv++);
```

```
            break;
        case 'k': eitem = atoi (*argv++);
            break;
        case 'l': bcid = atoi (*argv++);
            break;
        case 'm': ecid = atoi (*argv++);
            break;
        case 'g': gen = 1;
            strcpy (fname, *argv++);
            break;
        case 'l': logfile=fopen(*argv++,"w");
            break;
        default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
            fprintf (stderr, "(reached default case in getopt ())\n");
            myusage ();
      }
       }
  }

#else

  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 '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 'l': bcid = atoi (optarg);
            break;
      case 'm': ecid = 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 ();
    }
  }

# endif /* ORA_NT */

/*----------------------------------------------------------------*|
|   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_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_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 (do_C && (do_A || do_c)) {
    fprintf (stderr, "Cluster cust table around c_w_id or c_id?\n");
    myusage ();
  }

  if (eware <= 0)
    eware = scale;
  if (ecid <= 0)
    ecid = CUSTFAC;
  if (eitem <= 0)
    eitem = STOCFAC;

  if (do_C) {
    if ((bcid  < 1) || (bcid > CUSTFAC)) {
      fprintf (stderr, "Invalid beginning cid number: '%d'\n", bcid);
      myusage ();
    }

    if ((ecid  < bcid) || (ecid > CUSTFAC)) {
      fprintf (stderr, "Invalid ending cid number: '%d'\n", ecid);
      myusage ();
    }
  }
  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 (do_o) {
    if ((basename = getenv ("tpcc_bench")) == NULL)
    {
      fprintf (stderr, "$tpcc_bench is not set");
      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 */

    OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0,0);
    OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid
**)0);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 , (dvoid
**)0);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid
**)0);
    OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
    OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
            (ub4)0,OCI_ATTR_SERVER, errhp);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid
**)0);
    OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
            (ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
    OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
          OCI_ATTR_PASSWORD, errhp);
    OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));

    OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_SESSION, errhp);

    fprintf (stderr, "\nConnected to Oracle userid '%s/%s'.\n", uid, pwd);

    /* open cursors and parse statement */
    if (do_A || do_w)  {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curw), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curw, errhp, (text *)SQLTXTW,
              strlen((char *)SQLTXTW), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_d) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curd), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curd, errhp, (text *)SQLTXTD,
              strlen((char *)SQLTXTD), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_c || do_C) {
```

```
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curc), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curc, errhp, (text *)SQLTXTC,
            strlen((char *)SQLTXTC), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curcs), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curcs, errhp, (text *)SQLTXTCQUERY,
            strlen((char *)SQLTXTCQUERY), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
    }

    if (do_A || do_h) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curh), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curh, errhp, (text *)SQLTXTH,
            strlen((char *)SQLTXTH), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_s || do_S) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curs), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curs, errhp, (text *)SQLTXTS,
            strlen((char *)SQLTXTS), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curss), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curss, errhp, (text *)SQLTXTSQUERY,
            strlen((char *)SQLTXTSQUERY), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
    }

    if (do_A || do_i) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curi), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curi, errhp, (text *)SQLTXTI,
            strlen((char *)SQLTXTI), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_o) {
      int stat;
      char fname[160];
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curo1), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      DISCARD strcpy(fname,basename);
      DISCARD strcat(fname, "/");
      DISCARD strcat(fname, "benchrun/blocks/load_ordordl.sql");
      stat = sqlfile(fname, stmbuf);
      if (!stat)
      {
        fprintf (stderr, "unable to open %s \n",fname);
        quit();
        exit(1);
      }
      OCIERROR(errhp,OCIStmtPrepare(curo1, errhp, stmbuf,
            strlen((char *)stmbuf), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    if (do_A || do_n) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curno), OCI_HTYPE_STMT, 0,
(dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curno, errhp, (text *)SQLTXTNO,
            strlen((char *)SQLTXTNO), (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
    }

    /* bind variables */

    /* warehouse */

    if (do_A || do_w) {
      OCIERROR(errhp, OCIBindByName(curw, &w_id_bp, errhp, (text *)(":w_id"),
strlen((":w_id")),
            (ub1 *)&(w_id), sizeof(w_id), SQLT_INT, (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_name_bp, errhp,(text *)":w_name",
strlen(":w_name"),
            (ub1 *)w_name, 11, SQLT_STR, (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_street1_bp, errhp, (text
*)":w_street_1",
          strlen(":w_street_1"), (ub1 *)w_street_1, 21, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_street2_bp, errhp, (text
*)":w_street_2",
          strlen(":w_street_2"), (ub1 *)w_street_2, 21, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_city_bp, errhp, (text *)":w_city",
          strlen(":w_city"), (ub1 *)w_city, 21, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_state_bp, errhp, (text *)":w_state",
          strlen(":w_state"), (ub1 *)w_state, 2, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_zip_bp, errhp, (text *)":w_zip",
          strlen(":w_zip"), (ub1 *)w_zip, 9, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_tax_bp, errhp, (text *)":w_tax",
          strlen(":w_tax"), (ub1 *) & w_tax, sizeof(w_tax), SQLT_FLT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* district */

    if (do_A || do_d) {
      OCIERROR(errhp, OCIBindByName(curd, &d_id_bp, errhp, (text *)":d_id",
          strlen(":d_id"), (ub1 *)d_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_w_id_bp, errhp, (text *)":d_w_id",
          strlen(":d_w_id"), (ub1 *)d_w_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_name_bp, errhp, (text *)":d_name",
          strlen(":d_name"), (ub1 *)d_name, 11, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_street1_bp, errhp, (text *)":d_street_1",
          strlen(":d_street_1"), (ub1 *)d_street_1, 21, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_street2_bp, errhp, (text *)":d_street_2",
          strlen(":d_street_2"), (ub1 *)d_street_2, 21, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_city_bp, errhp, (text *)":d_city",
          strlen(":d_city"), (ub1 *)d_city, 21, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_state_bp, errhp, (text *)":d_state",
          strlen(":d_state"), (ub1 *)d_state, 2, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_zip_bp, errhp, (text *)":d_zip",
          strlen(":d_zip"), (ub1 *)d_zip, 9, SQLT_CHR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_tax_bp, errhp, (text *)":d_tax",
          strlen(":d_tax"), (ub1 *)d_tax, sizeof(float), SQLT_FLT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* customer */

    if (do_A || do_c || do_C) {
      OCIERROR(errhp, OCIBindByName(curcs, &s_c_id_bp, errhp, (text *)":s_c_id",
          strlen(":s_c_id"), (ub1 *)&s_c_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curcs, &s_c_w_id_bp, errhp, (text *)":s_c_w_id",
          strlen(":s_c_w_id"), (ub1 *)&s_c_w_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curcs, &s_c_d_id_bp, errhp, (text *)":s_c_d_id",
          strlen(":s_c_d_id"), (ub1 *)&s_c_d_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIDefineByPos(curcs,&s_c_ret_bp,errhp,1,&s_c_count,sizeof(int),SQLT_INT,\
          0,0,0,OCI_DEFAULT);

      OCIERROR(errhp, OCIBindByName(curc, &c_id_bp, errhp, (text *)":c_id",
          strlen(":c_id"), (ub1 *)c_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_d_id_bp, errhp, (text *)":c_d_id",
          strlen(":c_d_id"), (ub1 *)c_d_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_w_id_bp, errhp, (text *)":c_w_id",
          strlen(":c_w_id"), (ub1 *)c_w_id, sizeof(int), SQLT_INT,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_first_bp, errhp, (text *)":c_first",
          strlen(":c_first"), (ub1 *)c_first, 17, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_last_bp, errhp, (text *)":c_last",
          strlen(":c_last"), (ub1 *)c_last, 17, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curc, &c_street1_bp, errhp, (text *)":c_street_1",
          strlen(":c_street_1"), (ub1 *)c_street_1, 21, SQLT_STR,
          (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
```

```
    OCIERROR(errhp, OCIBindByName(curc, &c_street2_bp, errhp, (text *)":c_street_2",
            strlen(":c_street_2"), (ub1 *)c_street_2, 21, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_city_bp, errhp, (text *)":c_city",
            strlen(":c_city"), (ub1 *)c_city, 21, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_state_bp, errhp, (text *)":c_state",
            strlen(":c_state"), (ub1 *)c_state, 2, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_zip_bp, errhp, (text *)":c_zip",
            strlen(":c_zip"), (ub1 *)c_zip, 9, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_phone_bp, errhp, (text *)":c_phone",
            strlen(":c_phone"), (ub1 *)c_phone, 16, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_credit_bp, errhp, (text *)":c_credit",
            strlen(":c_credit"), (ub1 *)c_credit, 2, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_discount_bp, errhp, (text
*)":c_discount",
            strlen(":c_discount"), (ub1 *)c_discount, sizeof(float), SQLT_FLT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_data_bp, errhp, (text *)":c_data",
            strlen(":c_data"), (ub1 *)c_data, 501, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
  }

  /* item */

  if (do_A || do_i) {
    OCIERROR(errhp, OCIBindByName(curi, &i_id_bp, errhp, (text *)":i_id",
            strlen(":i_id"), (ub1 *)i_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curi, &i_im_id_bp, errhp, (text *)":i_im_id",
            strlen(":i_im_id"), (ub1 *)i_im_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curi, &i_name_bp, errhp, (text *)":i_name",
            strlen(":i_name"), (ub1 *)i_name, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curi, &i_price_bp, errhp, (text *)":i_price",
            strlen(":i_price"), (ub1 *)i_price, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curi, &i_data_bp, errhp, (text *)":i_data",
            strlen(":i_data"), (ub1 *)i_data, 51, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
  }

  /* stock */

  if (do_A || do_s || do_S) {
    OCIERROR(errhp, OCIBindByName(curss, &s_s_i_id_bp, errhp, (text *)":s_s_i_id",
            strlen(":s_s_i_id"), (ub1 *)&s_s_i_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curss, &s_s_w_id_bp, errhp, (text *)":s_s_w_id",
            strlen(":s_s_w_id"), (ub1 *)&s_s_w_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    OCIDefineByPos(curss,&s_s_ret_bp,errhp,1,&s_s_count,sizeof(int),SQLT_INT,\
            0,0,0,OCI_DEFAULT);

    OCIERROR(errhp, OCIBindByName(curs, &s_i_id_bp, errhp, (text *)":s_i_id",
            strlen(":s_i_id"), (ub1 *)s_i_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_w_id_bp, errhp, (text *)":s_w_id",
            strlen(":s_w_id"), (ub1 *)s_w_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_quantity_bp, errhp, (text *)":s_quantity",
            strlen(":s_quantity"), (ub1 *)s_quantity, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_01_bp, errhp, (text *)":s_dist_01",
            strlen(":s_dist_01"), (ub1 *)s_dist_01, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    OCIERROR(errhp, OCIBindByName(curs, &s_dist_02_bp, errhp, (text *)":s_dist_02",
            strlen(":s_dist_02"), (ub1 *)s_dist_02, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    OCIERROR(errhp, OCIBindByName(curs, &s_dist_03_bp, errhp, (text *)":s_dist_03",
            strlen(":s_dist_03"), (ub1 *)s_dist_03, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    OCIERROR(errhp, OCIBindByName(curs, &s_dist_04_bp, errhp, (text *)":s_dist_04",
            strlen(":s_dist_04"), (ub1 *)s_dist_04, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    OCIERROR(errhp, OCIBindByName(curs, &s_dist_05_bp, errhp, (text *)":s_dist_05",
            strlen(":s_dist_05"), (ub1 *)s_dist_05, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    OCIERROR(errhp, OCIBindByName(curs, &s_dist_06_bp, errhp, (text *)":s_dist_06",
            strlen(":s_dist_06"), (ub1 *)s_dist_06, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    OCIERROR(errhp, OCIBindByName(curs, &s_dist_07_bp, errhp, (text *)":s_dist_07",
            strlen(":s_dist_07"), (ub1 *)s_dist_07, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    OCIERROR(errhp, OCIBindByName(curs, &s_dist_08_bp, errhp, (text *)":s_dist_08",
            strlen(":s_dist_08"), (ub1 *)s_dist_08, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    OCIERROR(errhp, OCIBindByName(curs, &s_dist_09_bp, errhp, (text *)":s_dist_09",
            strlen(":s_dist_09"), (ub1 *)s_dist_09, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    OCIERROR(errhp, OCIBindByName(curs, &s_dist_10_bp, errhp, (text *)":s_dist_10",
            strlen(":s_dist_10"), (ub1 *)s_dist_10, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_data_bp, errhp, (text *)":s_data",
            strlen(":s_data"), (ub1 *)s_data, 51, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
  }

  /* history */

  if (do_A || do_h) {
    OCIERROR(errhp, OCIBindByName(curh, &h_c_id_bp, errhp, (text *)":h_c_id",
            strlen(":h_c_id"), (ub1 *)h_c_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_c_d_id_bp, errhp, (text *)":h_c_d_id",
            strlen(":h_c_d_id"), (ub1 *)h_d_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_c_w_id_bp, errhp, (text *)":h_c_w_id",
            strlen(":h_c_w_id"), (ub1 *)h_w_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_d_id_bp, errhp, (text *)":h_d_id",
            strlen(":h_d_id"), (ub1 *)h_d_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_w_id_bp, errhp, (text *)":h_w_id",
            strlen(":h_w_id"), (ub1 *)h_w_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_data_bp, errhp, (text *)":h_data",
            strlen(":h_data"), (ub1 *)h_data, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  }

  /* order and order_line (delivered) */

  if (do_A || do_o) {

    for (i = 0; i < ORDEARR; i++ ) {
      o_id_len[i] = sizeof(int);
      o_d_id_len[i] = sizeof(int);
      o_w_id_len[i] = sizeof(int);
```

```
    o_c_id_len[i] = sizeof(int);
    o_carrier_id_len[i] = sizeof(int);
    o_ol_cnt_len[i] = sizeof(int);
  }

  OCIERROR(errhp, OCIBindByName(curo1, &ol_o_id_bp, errhp, (text *)":ol_o_id",
           strlen(":ol_o_id"), (ub1 *)ol_o_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)ol_o_id_len, (ub2 *)0,
           (ub4) 15*ORDEARR, (ub4 *)&ol_o_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &ol_d_id_bp, errhp, (text *)":ol_d_id",
           strlen(":ol_d_id"), (ub1 *)ol_d_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)ol_d_id_len, (ub2 *)0,
           (ub4) 15*ORDEARR, (ub4 *) &ol_d_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &ol_w_id_bp, errhp, (text *)":ol_w_id",
           strlen(":ol_w_id"), (ub1 *)ol_w_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)ol_w_id_len, (ub2 *)0,
           (ub4) 15*ORDEARR, (ub4 *) &ol_w_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &ol_number_bp, errhp, (text
*)":ol_number",
           strlen(":ol_number"), (ub1 *)ol_number, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)ol_number_len, (ub2 *)0,
           (ub4) 15*ORDEARR, (ub4 *) &ol_number_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &ol_i_id_bp, errhp, (text *)":ol_i_id",
           strlen(":ol_i_id"), (ub1 *)ol_i_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)ol_i_id_len, (ub2 *)0,
           (ub4) 15*ORDEARR, (ub4 *) &ol_i_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &ol_supply_w_id_bp, errhp, (text
*)":ol_supply_w_id",
           strlen(":ol_supply_w_id"), (ub1 *)ol_supply_w_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)ol_supply_w_id_len, (ub2 *)0,
           (ub4) 15*ORDEARR, (ub4 *) &ol_supply_w_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &ol_dist_info_bp, errhp, (text
*)":ol_dist_info",
           strlen(":ol_dist_info"), (ub1 *)ol_dist_info, 24, SQLT_CHR,
           (dvoid *) 0, (ub2 *)ol_dist_info_len, (ub2 *)0,
           (ub4) 15*ORDEARR, (ub4 *) &ol_dist_info_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &ol_amount_bp, errhp, (text
*)":ol_amount",
           strlen(":ol_amount"), (ub1 *)ol_amount, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)ol_amount_len, (ub2 *)0,
           (ub4) 15*ORDEARR, (ub4 *) &ol_amount_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &o_id_bp, errhp, (text *)":o_id",
           strlen(":o_id"), (ub1 *)o_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)o_id_len, (ub2 *)0,
           (ub4) ORDEARR, (ub4 *) &o_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &o_d_id_bp, errhp, (text *)":o_d_id",
           strlen(":o_d_id"), (ub1 *)o_d_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)o_d_id_len, (ub2 *)0,
           (ub4) ORDEARR, (ub4 *) &o_d_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &o_w_id_bp, errhp, (text *)":o_w_id",
           strlen(":o_w_id"), (ub1 *)o_w_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)o_w_id_len, (ub2 *)0,
           (ub4) ORDEARR, (ub4 *) &o_w_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &o_c_id_bp, errhp, (text *)":o_c_id",
           strlen(":o_c_id"), (ub1 *)o_c_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)o_c_id_len, (ub2 *)0,
           (ub4) ORDEARR, (ub4 *) &o_c_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &o_carrier_id_bp, errhp, (text
*)":o_carrier_id",
           strlen(":o_carrier_id"), (ub1 *)o_carrier_id, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)o_carrier_id_len, (ub2 *)0,
           (ub4) ORDEARR, (ub4 *) &o_carrier_id_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &o_ol_cnt_bp, errhp, (text *)":o_ol_cnt",
           strlen(":o_ol_cnt"), (ub1 *)o_ol_cnt, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)o_ol_cnt_len, (ub2 *)0,
           (ub4) ORDEARR, (ub4 *) &o_ol_cnt_clen, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &o_ocnt_bp, errhp, (text *)":order_rows",
           strlen(":order_rows"), (ub1 *)&o_cnt, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)0, (ub2 *)0,
           (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

  OCIERROR(errhp, OCIBindByName(curo1, &o_olcnt_bp, errhp, (text *)":ordl_rows",
           strlen(":ordl_rows"), (ub1 *)&ol_cnt, sizeof(int), SQLT_INT,
           (dvoid *) 0, (ub2 *)0, (ub2 *)0,
           (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
  }

  /* new order */

  if (do_A || do_n) {
    OCIERROR(errhp, OCIBindByName(curno, &no_o_id_bp, errhp, (text *)":no_o_id",
             strlen(":no_o_id"), (ub1 *)no_o_id, sizeof(int), SQLT_INT,
             (dvoid *) 0, (ub2 *)0, (ub2 *)0,
             (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curno, &no_d_id_bp, errhp, (text *)":no_d_id",
             strlen(":no_d_id"), (ub1 *)no_d_id, sizeof(int), SQLT_INT,
             (dvoid *) 0, (ub2 *)0, (ub2 *)0,
             (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curno, &no_w_id_bp, errhp, (text *)":no_w_id",
             strlen(":no_w_id"), (ub1 *)no_w_id, sizeof(int), SQLT_INT,
             (dvoid *) 0, (ub2 *)0, (ub2 *)0,
             (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }
  }

/*---------------------------------------------------------------+
 | Initialize random number generator                            |
 +---------------------------------------------------------------*/

  srand (SEED);
#ifndef ORA_NT
  srand48 (SEED);
#endif
  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 = (float) ((lrand48 () % 2001) * 0.0001);
      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 %6.4f %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);

      status = OCIStmtExecute(tpcsvc, curw, errhp, (ub4) 1, (ub4) 0,
                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                   (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
        fprintf (stderr, "Error at ware %d\n", loop);
        OCIERROR(errhp, status);
        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] = (float) ((lrand48 () % 2001) * 0.0001);
        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 3000000 %6.4f 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 {
          status = OCIStmtExecute(tpcsvc, curd, errhp, (ub4) DISTARR, (ub4) 0,
                   (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                   (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
          if (status != OCI_SUCCESS) {
           fprintf (stderr, "Aborted at ware %d, dist 1\n", dwid);
                OCIERROR(errhp, status);
                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);

      if (getenv("tpcc_hash_overflow")) {
        fprintf(stderr,"Hash overflow is enabled\n");
        OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
        sprintf ((char *) stmbuf, SQLTXTENHA);
        OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                  OCI_NTV_SYNTAX, OCI_DEFAULT);
        OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
        OCIHandleFree(curi, OCI_HTYPE_STMT);
        fprintf (stderr,"Customer loaded for horizontal partitioning\n");
      }
      else
      {
        fprintf (stderr,"Customer not loaded for horizontal partitioning\n");
      }
      begin_time = gettime ();
      begin_cpu = getcpu ();

      s_c_id = 1;
      s_c_d_id = 1;
      s_c_w_id = bware;

      while (s_c_w_id <= eware) {
        status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
                 (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                 (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
          if (status != OCI_SUCCESS) {
                OCIERROR(errhp, status);
                quit ();
                exit (1);
          }
          if (s_c_count == 0) {
            s_c_w_id--;
            break;
          }
          else s_c_w_id++;
        }

        if (s_c_w_id < bware ) s_c_w_id = bware;
        else {
          if (s_c_w_id > eware ) s_c_w_id = eware;
          while (s_c_d_id <= DISTFAC) {
            status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
                     (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                     (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
              if (status != OCI_SUCCESS) {
               fprintf (stderr, "Select failed\n");
                     OCIERROR(errhp, status);
                     quit ();
                     exit (1);
              }
              if (s_c_count == 0) {
                s_c_d_id--;
                break;
              }
              else s_c_d_id++;
          }
          if (s_c_d_id > DISTFAC) s_c_d_id = DISTFAC;

          while (s_c_id <= CUSTFAC) {
            status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
                     (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                     (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
              if (status != OCI_SUCCESS) {
                     OCIERROR(errhp, status);
                     quit ();
                     exit (1);
              }
              if (s_c_count == 0) break;
              else s_c_id++;
          }
          }
          if (s_c_id > CUSTFAC) {
            if (s_c_d_id == DISTFAC) {
              s_c_d_id=1;
              s_c_w_id++;
            } else {
            s_c_d_id++;
             }
             s_c_id=1;
          }

          fprintf (stderr, "start at wid: %d, did: %d, cid: %d\n   ",s_c_w_id, s_c_d_id, s_c_id);
          cid = s_c_id - 1;
          cdid = s_c_d_id;
          cwid = s_c_w_id;
          nrows = (eware - s_c_w_id + 1) * DISTFAC * CUSTFAC - (s_c_d_id - 1) * CUSTFAC -
s_c_id + 1;
          fprintf (stderr, "remaining rows: %d\n   ", nrows);
          loopcount = 0;

          for (row = 0; row < nrows; ) {
            for (i = 0; i < CUSTARR && row < nrows; i++, row++) {
              cid++;
              if (cid > CUSTFAC) {       /* cycle cust id */
                cid = 1;                 /* cheap mod */
                cdid++;                  /* shift dist cycle */
                if (cdid > DISTFAC) {
                  cdid = 1;
                  cwid++;                /* shift ware 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 (lrand48 () % 10)
                c_credit[i][0] = 'G';
              else
                c_credit[i][0] = 'B';
              c_discount[i] = (float)((lrand48 () % 5001) * 0.0001);
              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 %6.4f -
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 {
              status = OCIStmtExecute(tpcsvc, curc, errhp, (ub4) i, (ub4) 0,
                       (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                       (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);

              if (status != OCI_SUCCESS) {
               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]);
                    OCIERROR(errhp, status);
                    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 < 0 ? 0 : nrows, end_time - begin_time, end_cpu - begin_cpu);
          if (getenv("tpcc_hash_overflow")) {
            fprintf(stderr,"Hash overflow is disabled\n");
            OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
            sprintf ((char *) stmbuf, SQLTXTDIHA);
            OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
```

```
                    OCI_NTV_SYNTAX, OCI_DEFAULT);
      OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
      OCIHandleFree(curi, OCI_HTYPE_STMT);
    }
  }


/*-------------------------------------------------------------+
 | Load the CUSTOMER table (cluster around c_id)     |
 +-------------------------------------------------------------*/

  if (do_C) {

    srand (bcid);
#ifndef ORA_NT
    srand48 (bcid);
#endif

    nrows = (ecid - bcid + 1) * (eware - bware +1) * DISTFAC;

    fprintf (stderr, "Loading/generating customer: c%d - c%d, w%d - w%d (%d rows)\n   ",
          bcid, ecid, bware, eware, nrows);

    if (getenv("tpcc_hash_overflow")) {
      fprintf(stderr,"Hash overflow is enabled\n");
      OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
      sprintf ((char *) stmbuf, SQLTXTENHA);
      OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                  OCI_NTV_SYNTAX, OCI_DEFAULT);
      OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
      OCIHandleFree(curi, OCI_HTYPE_STMT);
      fprintf (stderr,"Customer loaded for horizontal partitioning\n");
    }
    else
    {
      fprintf (stderr,"Customer not loaded for horizontal partitioning\n");
    }
    begin_time = gettime ();
    begin_cpu = getcpu ();

    s_c_id = bcid;
    s_c_d_id = 1;
    s_c_w_id = bware;

    while (s_c_id <= ecid) {
      status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }

        if (s_c_count == 0) {
          s_c_id--;
          break;
        }
        else s_c_id++;
    }

    if (s_c_id < bcid ) s_c_id = bcid;
    else {
      if (s_c_id > ecid ) s_c_id = ecid;
      while (s_c_w_id <= eware) {
        status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
              (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
              (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
          if (status != OCI_SUCCESS) {
          fprintf (stderr, "Select failed\n");
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_c_count == 0) {
          s_c_w_id--;
          break;
        }
        else s_c_w_id++;
      }
      if (s_c_w_id > eware) s_c_w_id = eware;
      else if (s_c_w_id < bware) s_c_w_id = bware;

      while (s_c_d_id <= DISTFAC) {
        status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
              (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
              (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
          if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
          }
          if (s_c_count == 0) break;
          else s_c_d_id++;
      }
    }

    if (s_c_d_id > DISTFAC) {
      s_c_d_id=1;
      if (s_c_w_id==eware) {
        s_c_w_id=bware;
        s_c_id++;
      }
      else s_c_w_id++;
    }

    fprintf (stderr, "start at cid: %d, wid: %d, did: %d\n   ",s_c_id, s_c_w_id, s_c_d_id);
    cid = s_c_id;
    cdid = s_c_d_id-1;
    cwid = s_c_w_id;
    nrows = (ecid - s_c_id + 1) * (eware - bware + 1) * DISTFAC - (s_c_w_id - 1) *
DISTFAC - s_c_d_id + 1;
    fprintf (stderr, "remaining rows: %d\n   ", nrows);
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < CUSTARR && row < nrows; i++, row++) {
        cdid++;
        if (cdid > DISTFAC) {       /* cycle dist id */
          cdid = 1;                 /* cheap mod */
          cwid++;                   /* shift dist cycle */
          if (cwid > eware) {
            cwid = bware;           /* shift ware cycle */
            cid++;
          }
        }
        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 (lrand48 () % 10)
          c_credit[i][0] = 'G';
        else
          c_credit[i][0] = 'B';
        c_discount[i] = (float)((lrand48 () % 5001) * 0.0001);
        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 %6.4f -
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 {
        status = OCIStmtExecute(tpcsvc, curc, errhp, (ub4) i, (ub4) 0,
                  (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                  (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);

        if (status != OCI_SUCCESS) {
         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]);
              OCIERROR(errhp, status);
              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 < 0 ? 0 : nrows, end_time - begin_time, end_cpu - begin_cpu);
    if (getenv("tpcc_hash_overflow")) {
      fprintf(stderr,"Hash overflow is disabled\n");
      OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
      sprintf ((char *) stmbuf, SQLTXTDIHA);
      OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                  OCI_NTV_SYNTAX, OCI_DEFAULT);
      OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi, errhp,1,0,0,0,OCI_DEFAULT));
      OCIHandleFree(curi, OCI_HTYPE_STMT);
    }
  }


/*-------------------------------------------------------------+
 | Load the ITEM table.                        |
 +-------------------------------------------------------------*/

  if (do_A || do_i) {
    nrows = ITEMFAC;

    fprintf (stderr, "Loading/generating item: (%d rows)\n   ", nrows);
```

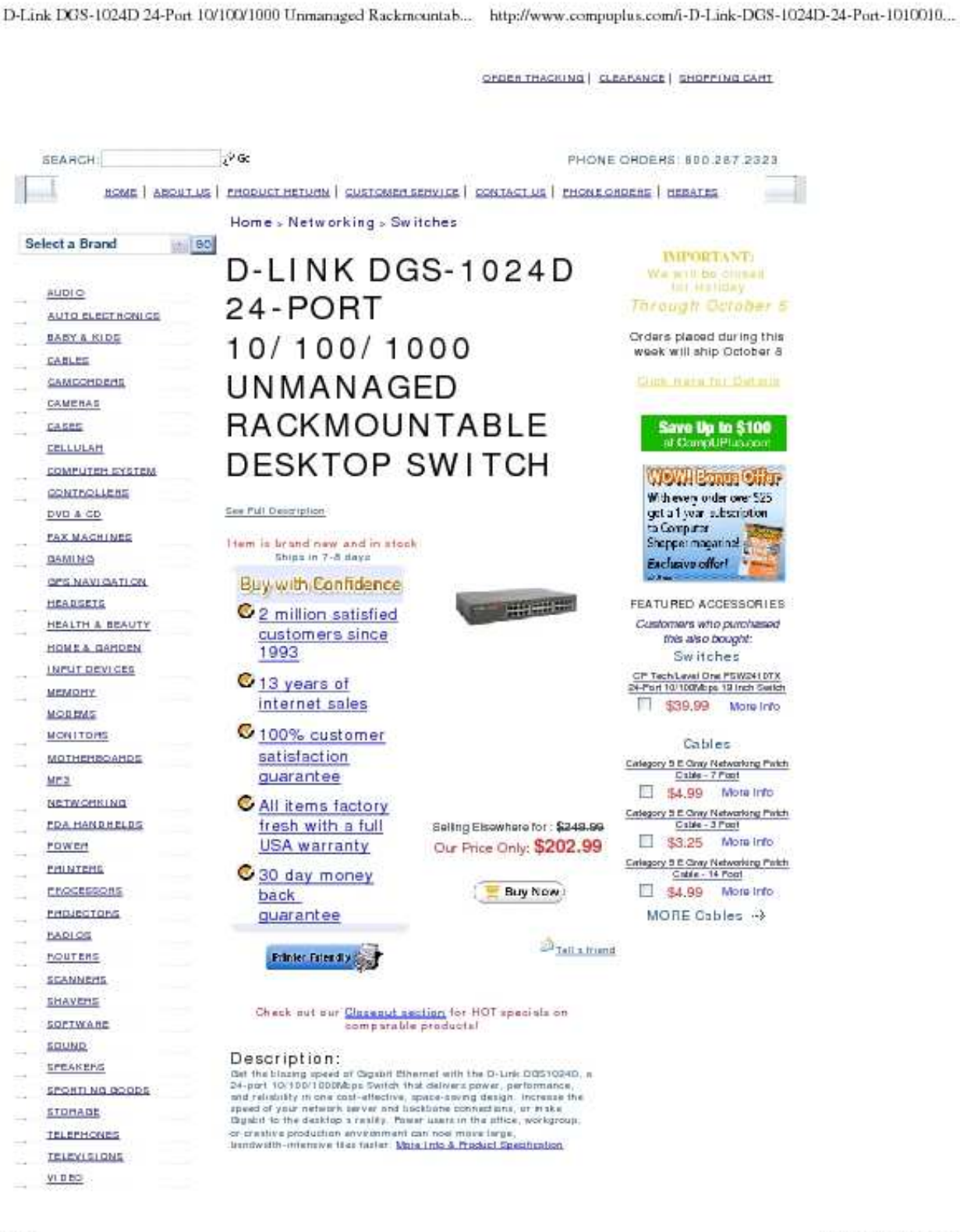
```
    begin_time = gettime ();
    begin_cpu = getcpu ();

    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < ITEMARR; i++, row++) {
        i_im_id[i] = (lrand48 () % 10000) + 1;
        i_price[i] = ((lrand48 () % 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 {
        status = OCIStmtExecute(tpcsvc, curi, errhp, (ub4) ITEMARR, (ub4) 0,
                  (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                  (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
         fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
              OCIERROR(errhp, status);
              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 ();

    s_s_i_id = 1;
    s_s_w_id = bware;

    while (s_s_w_id <= eware) {
      status = OCIStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4) 0,
              (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
              (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
        OCIERROR(errhp, status);
        quit ();
        exit (1);
      }
      if (s_s_count == 0) {
        s_s_w_id--;
        break;
      }
      else s_s_w_id++;
    }

    if (s_s_w_id < bware ) s_s_w_id = bware;
    else {
      if (s_s_w_id > eware) s_s_w_id = eware;
      while (s_s_i_id<=STOCFAC) {
        status = OCIStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
          OCIERROR(errhp, status);
          quit ();
          exit (1);
        }
        if (s_s_count == 0) {
          break;
        }
        else s_s_i_id++;
      }
    }
    if (s_s_i_id > STOCFAC) {
      s_s_i_id=1;
      s_s_w_id++;
    }

    fprintf(stderr,"start at s_i_id: %d, s_w_id: %d\n   ", s_s_i_id, s_s_w_id);

    sid = s_s_i_id - 1;
    swid = s_s_w_id;
    nrows = (eware - s_s_w_id + 1) * STOCFAC - ( s_s_i_id - 1);
    fprintf (stderr, "remaining rows: %d\n   ", nrows);
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      /* added row < nrows condition on next line - alex.ni */
    for (i = 0; (i < STOCARR) && (row < nrows); i++, row++) {
        if (++sid > STOCFAC) {        /* cheap mod */
          sid = 1;
          swid++;
        }
        s_quantity[i] = (lrand48 () % 91) + 10;
        randstr (s_dist_01[i], 24, 24);
        randstr (s_dist_02[i], 24, 24);
        randstr (s_dist_03[i], 24, 24);
        randstr (s_dist_04[i], 24, 24);
        randstr (s_dist_05[i], 24, 24);
        randstr (s_dist_06[i], 24, 24);
        randstr (s_dist_07[i], 24, 24);
        randstr (s_dist_08[i], 24, 24);
        randstr (s_dist_09[i], 24, 24);
        randstr (s_dist_10[i], 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], s_dist_01[i], s_dist_02[i],
                  s_dist_03[i], s_dist_04[i], s_dist_05[i], s_dist_06[i],
                  s_dist_07[i], s_dist_08[i], s_dist_09[i], s_dist_10[i],
                  s_data[i]);
        }
        else {
          s_i_id[i] = sid;
          s_w_id[i] = swid;
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
     /* Changed to STOCKARR to i - alex.ni */
        status = OCIStmtExecute(tpcsvc, curs, errhp, (ub4) i, (ub4) 0,
                  (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                  (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
          fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0], s_i_id[0]);
            OCIERROR(errhp, status);
            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 < 0 ? 0 : 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 ();

    s_s_i_id = bitem;
    s_s_w_id = bware;

    while (s_s_i_id <= eitem) {
      status = OCIStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4) 0,
              (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
              (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
          OCIERROR(errhp, status);
          quit ();
          exit (1);
      }
      if (s_s_count == 0) {
        s_s_i_id--;
        break;
      }
      else s_s_i_id++;
    }

    if (s_s_i_id < bitem ) s_s_i_id = bitem;
    else {
```

```
  if (s_s_i_id > eitem) s_s_i_id = eitem;
  while (s_s_w_id <= eware) {
    status = OCIStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
      }
      if (s_s_count == 0) {
        break;
      }
      else s_s_w_id++;
  }
}
if (s_s_w_id > eware) {
  s_s_w_id=bware;
  s_s_i_id++;
}

fprintf(stderr,"start at s_i_id: %d, s_w_id: %d\n   ", s_s_i_id, s_s_w_id);

sid = s_s_i_id;
swid = s_s_w_id - 1;
nrows = (eitem - s_s_i_id + 1) * (eware - bware + 1) - (s_s_w_id - bware);
fprintf (stderr, "remaining rows: %d\n   ", nrows);
loopcount = 0;

for (row = 0; row < nrows; ) {
  for (i = 0; i < STOCARR && row < nrows; i++, row++) {
    if (++swid > eware) {          /* cheap mod */
      swid = bware;
      sid++;
    }
    s_quantity[i] = (lrand48 () % 91) + 10;
    randstr (s_dist_01[i], 24, 24);
    randstr (s_dist_02[i], 24, 24);
    randstr (s_dist_03[i], 24, 24);
    randstr (s_dist_04[i], 24, 24);
    randstr (s_dist_05[i], 24, 24);
    randstr (s_dist_06[i], 24, 24);
    randstr (s_dist_07[i], 24, 24);
    randstr (s_dist_08[i], 24, 24);
    randstr (s_dist_09[i], 24, 24);
    randstr (s_dist_10[i], 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], s_dist_01[i], s_dist_02[i],
            s_dist_03[i], s_dist_04[i], s_dist_05[i], s_dist_06[i],
            s_dist_07[i], s_dist_08[i], s_dist_09[i], s_dist_10[i],
            s_data[i]);
    }
    else {
      s_i_id[i] = sid;
      s_w_id[i] = swid;
    }
  }

  if (gen) {
    fflush (stdout);
  }
  else {
    status = OCIStmtExecute(tpcsvc, curs, errhp, (ub4) i, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
    if (status != OCI_SUCCESS) {
          fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0], s_i_id[0]);
          OCIERROR(errhp, status);
          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 < 0 ? 0 : 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 {
      status = OCIStmtExecute(tpcsvc, curh, errhp, (ub4) HISTARR, (ub4) 0,
              (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
              (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
      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]);
          OCIERROR(errhp, status);
          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) {

  int batch_olcnt;

  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; ) {

    batch_olcnt = 0;

    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] = lrand48 () % 10 + 1;
      o_ol_cnt[i] = olcnt = lrand48 () % 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 */
          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];
    if (cid >= 2101 ) {
      o_carrier_id[i] = 11;
    }
  }

  for (j = 0; j < o_ol_cnt[i]; j++, batch_olcnt++ ) {
    ol_i_id[batch_olcnt] = sid = lrand48 () % 100000 + 1;
    if (cid < 2101)
      ol_amount[batch_olcnt] = 0;
    else
      ol_amount[batch_olcnt] = (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,
                cid, cwid, j + 1, sdate, ol_i_id[batch_olcnt], cwid,
                ol_amount[batch_olcnt], str24[j]);
      }
      else {
        /* Insert a default date instead of null date */
        fprintf (olfp, "%d %d %d %d 01-Jan-1811 %d %d 5 %ld %s\n", cid,
                cid, cwid, j + 1, ol_i_id[batch_olcnt], cwid,
                ol_amount[batch_olcnt], str24[j]);
      }
    }
    else {
      ol_o_id[batch_olcnt] = cid;
      ol_d_id[batch_olcnt] = cdid;
      ol_w_id[batch_olcnt] = cwid;
      ol_number[batch_olcnt] = j + 1;
      ol_supply_w_id[batch_olcnt] = cwid;
      strncpy (ol_dist_info[batch_olcnt], str24[j], 24);
    }
  }
  if (gen) {
    fflush (olfp);
  }
}

o_cnt =  ORDEARR;
ol_cnt =  batch_olcnt;

for (j = 0; j < batch_olcnt; j++) {
  ol_o_id_len[j] = sizeof(int);
  ol_d_id_len[j] = sizeof(int);
  ol_w_id_len[j] = sizeof(int);
  ol_number_len[j] = sizeof(int);
  ol_i_id_len[j] = sizeof(int);
  ol_supply_w_id_len[j] = sizeof(int);
  ol_dist_info_len[j] = 24;
  ol_amount_len[j] = sizeof(int);
}
for (j = batch_olcnt; j < 15*ORDEARR; j++) {
  ol_o_id_len[j] = 0;
  ol_d_id_len[j] = 0;
  ol_w_id_len[j] = 0;
  ol_number_len[j] = 0;
  ol_i_id_len[j] = 0;
  ol_supply_w_id_len[j] = 0;
  ol_dist_info_len[j] = 0;
  ol_amount_len[j] = 0;
}

o_id_clen = ORDEARR;
o_d_id_clen = ORDEARR;
o_w_id_clen = ORDEARR;
o_c_id_clen = ORDEARR;
o_carrier_id_clen = ORDEARR;
o_ol_cnt_clen = ORDEARR;

ol_o_id_clen = batch_olcnt;
ol_d_id_clen = batch_olcnt;
ol_w_id_clen = batch_olcnt;
ol_number_clen = batch_olcnt;
ol_i_id_clen = batch_olcnt;
ol_supply_w_id_clen = batch_olcnt;
ol_dist_info_clen = batch_olcnt;
ol_amount_clen = batch_olcnt;

OCIERROR(errhp, OCIStmtExecute(tpcsvc, curo1, errhp, (ub4) 1, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS ));

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 {
      status = OCIStmtExecute(tpcsvc, curno, errhp, (ub4) NEWOARR, (ub4) 0,
                      (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                      (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
    fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n", cwid, cdid, cid + 2100);
          OCIERROR(errhp, status);
          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);

}

void 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 = lrand48 () % i;
    temp = randperm3000[i - 1];
    randperm3000[i - 1] = randperm3000[pos];
    randperm3000[pos] = temp;
  }
}

void randstr (str, x, y)
char *str;
int x;
int y;
{
  int i, j;
  int len;

  len = (lrand48 () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = lrand48 () % 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';

}

void randdatastr (str, x, y)
char *str;
int x;
int y;
{
  int i, j;
  int len;
  int pos;

  len = (lrand48 () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = lrand48 () % 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 ((lrand48 () % 10) == 0) {
    pos = (lrand48 () % (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';
  }
}

void randnum (str, len)
char *str;
int len;
{
  int i;

  for (i = 0; i < len; i++)
    str[i] = (char) (lrand48 () % 10 + '0');
  str[len] = '\0';

}

void 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]);
}

int 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);
}

void sysdate (sdate)
char *sdate;
{
  time_t tp;
  struct tm *tmptr;

  time (&tp);
  tmptr = localtime (&tp);
  strftime (sdate, 29, "%d-%b-%Y", tmptr);
}

int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
 text errbuf[512];
 sb4 errcode;
 sb4 lstat;
 ub4 recno=2;

 switch (status) {
 case OCI_SUCCESS:
   break;
 case OCI_SUCCESS_WITH_INFO:
   fprintf(stderr,"Module %s Line %d\n", fname, lineno);
   fprintf(stderr,"Error - OCI_SUCCESS_WITH_INFO\n");
   lstat = OCIErrorGet (errhp, recno++, (text *) NULL, &errcode, errbuf,
                        (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
   fprintf(stderr,"Error - %s\n", errbuf);
   break;
 case OCI_NEED_DATA:
   fprintf(stderr,"Module %s Line %d\n", fname, lineno);
   fprintf(stderr,"Error - OCI_NEED_DATA\n");
   return (IRRECERR);
 case OCI_NO_DATA:
   fprintf(stderr,"Module %s Line %d\n", fname, lineno);
   fprintf(stderr,"Error - OCI_NO_DATA\n");
   return (IRRECERR);
 case OCI_ERROR:
   lstat = OCIErrorGet (errhp, (ub4) 1,
          (text *) NULL, &errcode, errbuf,
           (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
   if (errcode == NOT_SERIALIZABLE) return (errcode);
   if (errcode == SNAPSHOT_TOO_OLD) return (errcode);
   while (lstat != OCI_NO_DATA)
   {
     fprintf(stderr,"Module %s Line %d\n", fname, lineno);
     fprintf(stderr,"Error - %s\n", errbuf);
     lstat = OCIErrorGet (errhp, recno++, (text *) NULL, &errcode, errbuf,
                          (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
   }
   return (errcode);
 case OCI_INVALID_HANDLE:
   fprintf(stderr,"Module %s Line %d\n", fname, lineno);
   fprintf(stderr,"Error - OCI_INVALID_HANDLE\n");
   exit(-1);
 case OCI_STILL_EXECUTING:
   fprintf(stderr,"Module %s Line %d\n", fname, lineno);
   fprintf(stderr,"Error - OCI_STILL_EXECUTE\n");
   return (IRRECERR);
 case OCI_CONTINUE:
   fprintf(stderr,"Module %s Line %d\n", fname, lineno);
   fprintf(stderr,"Error - OCI_CONTINUE\n");
   return (IRRECERR);
 default:
   fprintf(stderr,"Module %s Line %d\n", fname, lineno);
   fprintf(stderr,"Status - %s\n", status);
   return (IRRECERR);
 }
 return (RECOVERR);
}
```

12 Appendix D: Pricing



Shopping Cart

Item	Quantity		Price	Line Total
New Subscription Contract July 16, 2008 - July 16, 2011				
Red Hat Enterprise Linux Advanced Platform for IBM POWER, Premium (unlimited sockets) - for 3 Years July 16, 2008 - July 16, 2011	1	Remove	$6,747.00	$6,747.00
Promotion Code:		Update Cart	Subtotal:	$6,747.00

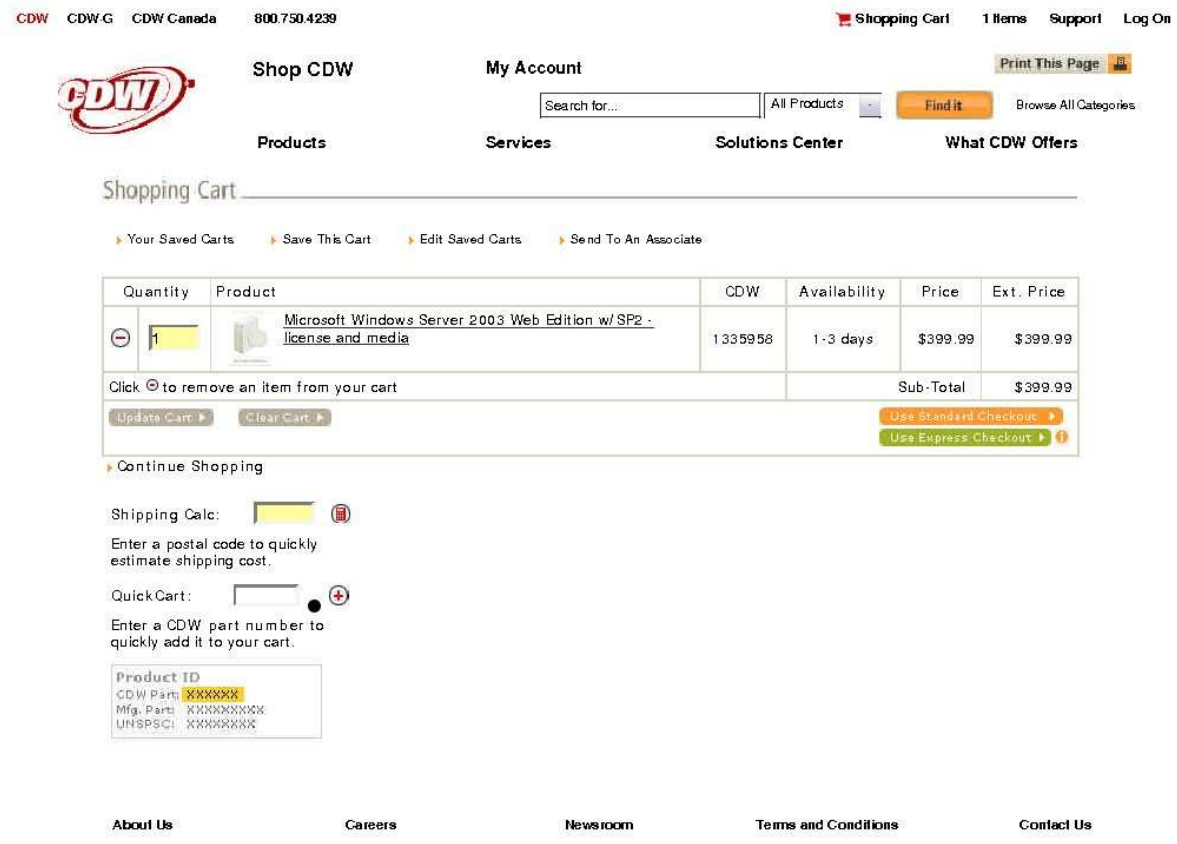
Optional Install Discs and Documentation

add to cart Red Hat Enterprise Linux 5 (for POWER) $25 Media Kit (DVD only)

Continue shopping

Continue to Checkout

ABOUT SSL CERTIFICATES



CDW CDW-G CDW Canada 800.750.4239 Shopping Cart 1 Items Support Log On

Shop CDW My Account Print This Page

Search for... All Products Find it Browse All Categories

Products Services Solutions Center What CDW Offers

Shopping Cart

Your Saved Carts | Save This Cart | Edit Saved Carts | Send To An Associate

Quantity	Product	CDW	Availability	Price	Ext. Price
1	Microsoft Windows Server 2003 Web Edition w/ SP2 - license and media	1335958	1-3 days	$399.99	$399.99
Click to remove an item from your cart				Sub-Total	$399.99

Update Cart | Clear Cart | Use Standard Checkout | Use Express Checkout

Continue Shopping

Shipping Calc:

Enter a postal code to quickly estimate shipping cost.

Quick Cart:

Enter a CDW part number to quickly add it to your cart.

Product ID
CDW Part: XXXXXX
Mfg. Part: XXXXXXXXX
UNSPSC: XXXXXXXX

About Us Careers Newsroom Terms and Conditions Contact Us

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

September 26, 2007

IBM Corporation
Andrea Davis
11501 Burnet Road
Austin, TX 78758

Here is the information you requested regarding pricing for several Microsoft products to be used in conjunction with your TPC-C benchmark testing.

All pricing shown is in US Dollars ($).

Part Number	Description	Unit Price	Quantity	Price
127-00012	**Visual Studio Standard 2005** *Full License* *No Discount Applied*	$250	1	$250
N/A	**Microsoft Problem Resolution Services** *Professional Support* *(1 Incident)*	$245	1	$245

All products are currently orderable through Microsoft's normal distribution channels. A list of Microsoft's resellers can be found at **http://www.microsoft.com/products/info/render.aspx?view=22&type=mnp&content=22/licensing**

Defect support is included in the purchase price. Additional support is available from Microsoft PSS on an incident by incident basis at $245 per call.

This quote is valid for the next 90 days.

If we can be of any further assistance, please contact Jamie Reding at (425) 703-0510 or jamiere@microsoft.com.

Reference ID: PCAnDa0709260000008633.
Please include this Reference ID in any correspondence regarding this price quote.

International Business Machines Corporation

11501 BURNETT RD
AUSTIN TX 78758

July 16, 2008

The requested quote for the IBM System pS 570 TPC-C benchmark using IBM System Storage DS4300.

Description	Part Number	Source	Unit Price	Qty	Ext Price	Maint.
Server Hardware						
IBM System p5 570	9117-570	1	3,867	1	3,867	36,333
2-Way 2.2GHz Processor Card, 0-way activation,	8338	1	6,850	4	27,400	
One way Processor Activation f	7618	1	13,700	4	54,800	
Op Panel	1846	1	199	1	199	
Processor Cable	1847	1	2,647	1	2,647	
SP Flex Cable	1857	1	1,324	1	1,324	
16GB (4x4GB) DIMMS, DDR2 SDRAM	4497	1	30,310	8	242,480	
73.4GB 15K RPM Ultra320 SCSI Disk Drive	3278	1	659	2	1,318	
4Gb Dual Port Fibre Channel	5759	1	3,308	5	16,540	
IDE Slimline DVD-ROM Drive	2640	1	274	1	274	
Processor Power Regulator	7768	1	675	6	4,050	
CEC Backplane	7865	1	1,588	2	3,176	
I/O Backplane	7866	1	5,426	2	10,852	
Midplane	7867	1	662	2	1,324	
DASD Backplane	7868	1	1,588	2	3,176	
Media Backplane	7869	1	185	1	185	
Power Midplane	7870	1	265	2	530	
System Port Riser Card	7878	1	132	2	264	
AC Power Supply, 240V, 1400W	7888	1	1,059	4	4,236	
FSP Service Processor Card	7997	1	860	1	860	
Power Cord, Drawer to IBM PDU 250V/10A	6671	1	19	4	76	
IBM Rack-mount Drawer Rail Kit	7164	1	222	2	444	
System Drawer Enclosure w/Bezel	7300	1	463	2	926	
Desktop Hardware Management Console	7310-C05	1	1,830	1	1,830	
IBM TS41H/L150P 15"TFT Display	3637	1	508	1	508	
IBM Full Width USB Keyboard	5951	1	104	1	104	
IBM 3-Button Optical Mouse - Black - USB	8841	1	78	1	78	
				Subtotal	383,468	36,333
Server Storage						
IBM System Storage EXP3000	1727-HC1	1	3,199	21	67,179	
IBM 3M SAS cable	39R6531	1	135	42	5,670	
IBM DS3000 (ESM)	39R6515	1	999	21	20,979	
IBM System Storage DS3400 DC Express	1726-42E	1	6702	8	53,616	
IBM Hot-Swap 3.5 inch 73.4GB 15K SAS HDD	40K1043	1	309	304	93,936	
IBM S2 42U Standard Rack	93074RX	1	1,489	2	2,978	
ServicePac for 3-Year 24x7x4 Support (DS3400)	44J8073	1	1,300	8	10,400	
ServicePac for 3-Year 24x7x4 Support(EXP3000)	41L2768	1	760	21	15,960	
ServicePac for 3-Year 24x7x4 Support (Rack)	41L2760	1	300	2	600	
				Subtotal	244,358	26,960
Client Hardware						
x3500 Dual-Core Intel Xeon 5160 3.0GHz	797792U	1	2,999	8	23,992	
73GB 15K SAS Hot-Swap Drive	40K1043	1	309	8	2,472	
ServicePac for 3-Year 24x7x4 Support (x3500)	96P2250	1	586	8		4,688
				Subtotal	26,464	4,688
				Total	691,596	67,981
				IBM Total System Discounts*		- $229,033
				Three-Year Cost of Ownership USD:		$501,054

For more information on:
All products: http://www.ibm.com/products
For Storage products: http://www-03.ibm.com/systems/storage/disk/index.html
For System p products: http://www-03.ibm.com/systems/p
For System x products: http://www-03.ibm.com/systems/x

For additional information, please contact me directly:
Dan Hebrank
IBM Sales & Distribution, STG Sales
1-314-252-4160