Cisco Systems, Inc.

TPC Benchmark™ C Full Disclosure Report

Cisco UCS C250 M2 Extended-Memory Server

using

Oracle Database 11g Standard Edition One

and

Oracle Linux with Unbreakable Enterprise Kernel Release 2

First Edition
December 2011

First Edition – December, 2011

Table of Contents

Abstract

This report documents the methodology and results of the TPC Benchmark © C test conducted on the Cisco UCS C250 M2Extended-Memory Server using Oracle Database 11g Standard Edition One and Oracle Linux with Unbreakable Enterprise Kernel Release 2.

Cisco UCS C250 M2 Extended-Memory Server C/S configuration with 5 Cisco UCS C200 M2 Servers

Company Name	System Name	Database Software	Opeating System
Cisco Systems, Inc	Cisco UCS C250 M2 Extended-Memory Server	Oracle Database 11g Standard Edition One	Oracle Linux with Unbreakable Enterprise Kernel Release 2

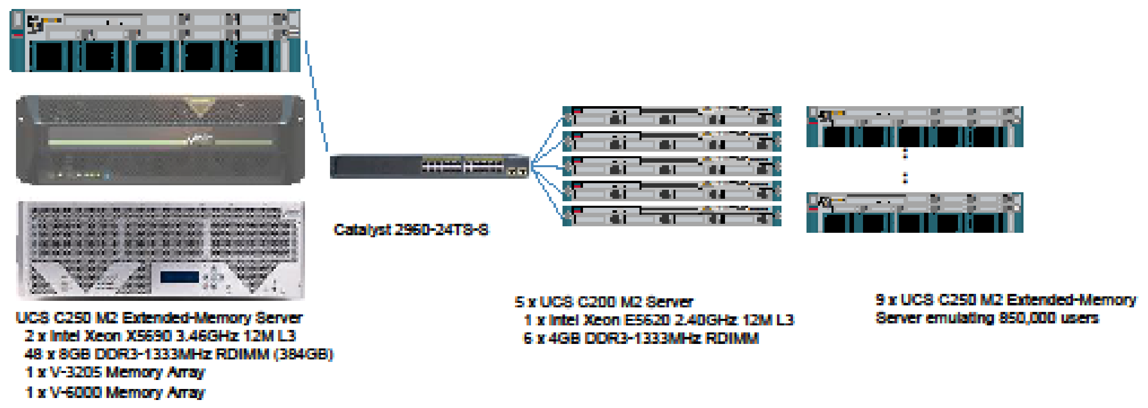
TPC Benchmark© C Metrics

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$602,316	1,053,100.32 tpmC	$0.58 USD/ tpmC	December 7, 2011

CISCO	Cisco UCS C250 M2 Extended-Memory Server	**TPC-C Rev. 5.11** **TPC-Pricing Rev. 1.60** **Report Date:** December 7, 2011

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$602,316 USD	1,053,100.32 tpmC	$0.58 USD/ tpmC	December 7, 2011

Processors	**Database Manager**	**Operating System**	**Other Software**	**Number of Users**
Intel Xeon X5690 3.46 GHz 12M L3 Cache	Oracle Database 11g Standard Edition One	Oracle Linux with Unbreakable Enterprise Kernel Release 2	Microsoft COM+	**850,000**


Catalyst 2960-24TS-S
UCS C250 M2 Extended-Memory Server
2 x Intel Xeon X5690 3.46GHz 12M L3
48 x 8GB DDR3-1333MHz RDIMM (384GB)
1 x V-3205 Memory Array
1 x V-6000 Memory Array
5 x UCS C200 M2 Server
1 x Intel Xeon E5620 2.40GHz 12M L3
6 x 4GB DDR3-1333MHz RDIMM
9 x UCS C250 M2 Extended-Memory Server emulating 850,000 users

	Server		**5 Clients** (each with)	
System Components	**Quantity**	**Description**	**Quantity**	**Description**
Processor/Cores/Threads	2/12/24	Intel Xeon X5690 3.46GHz 12M L3 Cache	1/4/8	Intel Xeon E5620 2.40GHz 12M L3 Cache
Memory	384 GB	48 x 8GB DDR3	24 GB	6 x 4GB DDR3
Storage Controllers	2	LSI MegaRAID SAS 9280-4i4e	1	LSI 1064E
Disk Drives (Internal)	2	300GB 6Gb SAS 10K RPM SFF HDD (OS)	1	500GB SATA 7.2K RPM HDD (OS)
Storage Arrays	1	V-3205 Memory Array (5.3 TB)		
	1	V-6000 Memory Array (16.3 TB)		
	2	VTrak J630sS JBOD with: 6 x 600GB 15K SAS HDD, 8 x 2TB 7.2K NL-SAS HDD		
Total Storage	62.6 TB			

TPC-C Executive Summary
© 2011 Cisco Systems, Inc. All rights reserved.

Cisco	Cisco UCS C250 M2 Extended-Memory Server	TPC-C Rev. 5.11 TPC-Pricing Rev. 1.6.0 Report Date: 6-Dec-2011

Description	Part Number	Brand	Source	Unit Price	Qty	Extended Price	3 Year Maint. Price
Server Hardware							
UCS C250 M2 Srvr w/1PSU,DVD w/o CPU, mem, HDD, or PCIe card	R250-2480805W	Cisco	1	$4,687	1	$4,687	
Intel Xeon X5690 3.46GHz/6c/130W/12M/DDR3 1333MHz	A01-X0115	Cisco	1	$4,686	2	$9,372	
8GB DDR3-1333MHz RDIMM/PC3-10600	N01-M308GB2-L	Cisco	1	$435	48	$20,880	
850W power supply for the UCS C250 M2	R250-PSU2-850W	Cisco	1	$560	1	$560	
LSI MegaRAID SAS 9280-4i4e, 4 internal and 4 external ports	UCSC-RAID-C-4I4E	Cisco	1	$1,592	2	$3,184	
300GB 6Gb SAS 10K RPM SFF HDD	A03-D300GA2	Cisco	1	$561	2	$1,122	
MS Comfort Curve DT 3000 USB-EN NA Keyboard & Mouse	7ZJ-00001	Provantage	6	$25	1	$25	
V173DJb LCD Monitor	ET-BV3RP-D03	Provantage	6	$90	1	$90	
Cisco R42610 expansion rack	RACK-UCS	Cisco	1	$2,857	1	$2,857	
C250 24x7x4Hr Onsite Support Service	CON-UCS7-R250W	Cisco	1	$566	3		$1,698
					Subtotal	$42,777	$1,698
Storage							
V-6000 Memory Array	V-6616-HA64-4xPCIe	Violin	4	$530,000	1	$530,000	Inc
V-3205Memory Array	V-3205	Violin	4	$150,000	1	$150,000	Inc
Violin Bronze Maintenance 3Y 24x7 Hr	VS-BRONZE	Violin	4	$93,085	1		$68,952
VTrak J630sS JBOD storage enclosure	VTJ630sS	Datalink	5	$4,189	2	$8,378	
600GB 15K SAS Drive (inc. 10% spare)	X30DVEA1	Datalink	5	$641	14	$8,974	
2TB 7.2K NL-SAS Drive (inc. 10% spare)	X30DVSA1	Datalink	5	$385	18	$6,930	
Promise ServicePlus Plan	SP3YEAR	Datalink	5	$883	2		$1,766
					Subtotal	$704,282	$70,718
Client Hardware							
UCS C200 M2 Srvr w/1PSU, DVD w/o CPU, mem, HDD or PCIe card	R200-1120402W	Cisco	1	$2,343	5	$11,715	
2.26GHz Xeon E5520 80W CPU/8MB cache/DDR3 1066MHz	N20-X00003	Cisco	1	$1,124	5	$5,620	
4GB DDR3-1333-MHz RDIMM/PC3-10600	UCS-MR-1X041RX-A	Cisco	1	$319	30	$9,570	
LSI 1064E (4-port SAS 3.0G RAID 0, 1, 1E) Mezz Card	R2X0-ML002	Cisco	1	$281	5	$1,405	
Cisco UCS P81E Virtual Interface Card	N2XX-ACPCI01	Cisco	1	$1,594	5	$7,970	
Gen 2 500GB SATA 7.2K RPM 3.5in HDD	R200-D500GCSATA03	Cisco	1	$448	5	$2,240	
C200 24x7x4Hr Onsite Support Service	CON-UCS7-R200W	Cisco	1	$405	15		$6,075
					Subtotal	$38,520	$6,075
Connectivity							
Catalyst 2960S 24 GigE	WS-C3560CG-8TC-S	Cisco	1	$1,995	1	$1,995	
Onsite 24x7x4 Cat 2960S 24GigE	CON-OSP-2960S2SS	Cisco	1	$280	3		$840
					Subtotal	$1,995	$840
Large Purchase Discount [i]	57.0%	Cisco	1			-$46,339	
Large Purchase Discount	35.0%	Violin	1			-$238,000	
					Hardware Subtotal	$503,234	$79,331
Server Software							
Oracle Linux		Oracle	2	$0			
Oracle Linux Basic Limited (3 year support)		Oracle	2	$1,497	1		$1,497
Oracle Database 11g Release 2 Standard Edition One, Per Processor for 3Years		Oracle	2	$2,900	2	$5,800	
Incident Server Support		Oracle	2	$2,300	3		$6,900
					Subtotal	$5,800	$8,397
Client Software							
Microsoft Windows Server 2008 Standard Edition	P73-03883	Microsoft	3	$999	5	$4,995	
Microsoft Problem Resolution Services	NA	Microsoft	3	$259	1		$259
Microsoft Visual Studio Standard 2008	127-00166	Microsoft	3	$299	1	$299	
					Subtotal	$5,294	$259
					Total	$514,328	$87,987

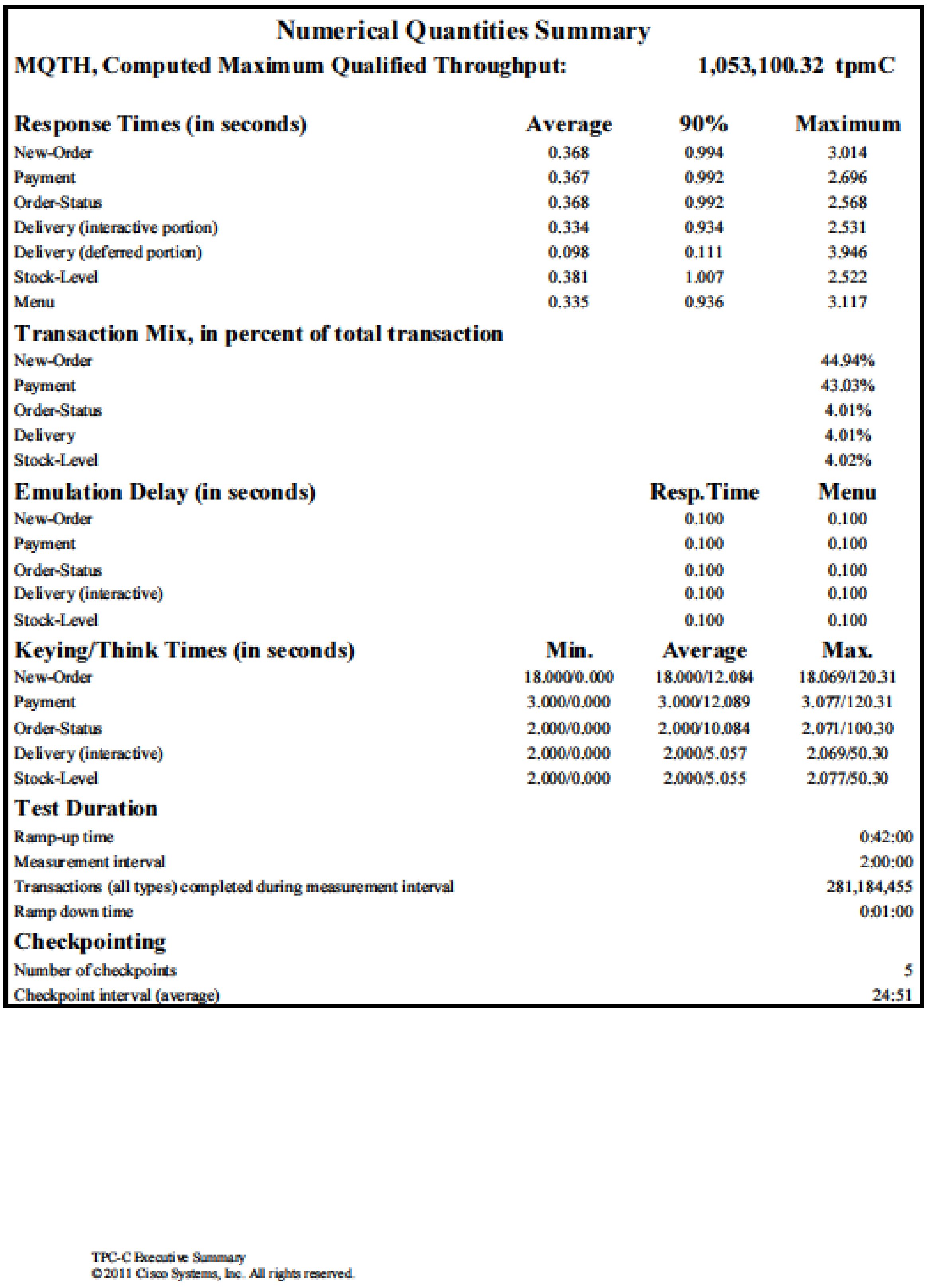
Pricing: 1=Cisco 2=Oracle 3=Microsoft 4=Viollin 5=Datalink 6=Provantage; i-Excludes Support Services and Connectivity

Audited by Francois Raab of InfoSizing,Inc. www.sizing.com

All discounts are based on US list prices and for similar quantities and configurations. The discounts are based on the overall specific components pricing from respective vendors in this single quotation. Discounts for similarly sized configurations will be similar to those quoted here, but may vary based on the components in the configuration.

Three-Year Cost of Ownership:	**$602,316 USD**
tpmC:	**1,053,100.32**
$ / tpmC:	**$0.58 USD**

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



Numerical Quantities Summary

MQTH, Computed Maximum Qualified Throughput: **1,053,100.32 tpmC**

Response Times (in seconds)	**Average**	**90%**	**Maximum**
New-Order	0.368	0.994	3.014
Payment	0.367	0.992	2.696
Order-Status	0.368	0.992	2.568
Delivery (interactive portion)	0.334	0.934	2.531
Delivery (deferred portion)	0.098	0.111	3.946
Stock-Level	0.381	1.007	2.522
Menu	0.335	0.936	3.117

Transaction Mix, in percent of total transaction	
New-Order	44.94%
Payment	43.03%
Order-Status	4.01%
Delivery	4.01%
Stock-Level	4.02%

Emulation Delay (in seconds)	**Resp.Time**	**Menu**
New-Order	0.100	0.100
Payment	0.100	0.100
Order-Status	0.100	0.100
Delivery (interactive)	0.100	0.100
Stock-Level	0.100	0.100

Keying/Think Times (in seconds)	**Min.**	**Average**	**Max.**
New-Order	18.000/0.000	18.000/12.084	18.069/120.31
Payment	3.000/0.000	3.000/12.089	3.077/120.31
Order-Status	2.000/0.000	2.000/10.084	2.071/100.30
Delivery (interactive)	2.000/0.000	2.000/5.057	2.069/50.30
Stock-Level	2.000/0.000	2.000/5.055	2.077/50.30

Test Duration	
Ramp-up time	0:42:00
Measurement interval	2:00:00
Transactions (all types) completed during measurement interval	281,184,455
Ramp down time	0:01:00

Checkpointing	
Number of checkpoints	5
Checkpoint interval (average)	24:51

TPC-C Executive Summary
© 2011 Cisco Systems, Inc. All rights reserved.

Preface

The Processing Performance Council (TPC) is a non-profit corporation founded to define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry. The TPC Benchmark© C is an on-line transaction processing benchmark (OLTP) developed by the TPC.

TPC Benchmark© C Overview

TPC Benchmark© C (TPC-C) simulates a complete computing environment where a population of users executes transactions against a database. The benchmark is centered around the principal activities (transactions) of an order-entry environment. These transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. While the benchmark portrays the activity of a wholesale supplier, TPC-C is not limited to the activity of any particular business segment, but, rather represents any industry that must manage, sell, or distribute a product or service.

TPC-C consists of a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP application environments. It does so by exercising a breadth of system components associated with such environments, which are characterized by:

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

The performance metric reported by TPC-C is a “business throughput” measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of processing an order, and each transaction is subject to a response time constraint. The performance metric for this benchmark is expressed in transactions-per-minute-C (tpmC). To be 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.

TPC-C uses terminology and metrics that are similar to other benchmarks, originated by the TPC or others. Such similarity in terminology does not in any way imply that TPC-C results are comparable to other benchmarks. The only benchmark results comparable to TPC-C are other TPC-C results conformant with the same revision.

Despite the fact that this benchmark offers a rich environment that emulates many OLTP applications, this benchmark does not reflect the entire range of OLTP requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to other environments are not recommended.

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 benchmark when critical capacity planning and/or product evaluation decisions are contemplated.

Further information is available at www.tpc.org

0 General Items

0.1 Application Code and Definition Statements

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

Appendix A contains the application source code for the transactions.

0.2 Benchmark Sponsor

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

This benchmark was sponsored by Cisco Systems, Inc. and developed and engineered in partnership with Oracle 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 by not limited to:

- *Database options*
- *Recover/commit options*
- *Consistency locking options*
- *Operating system and application configuration parameters*

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

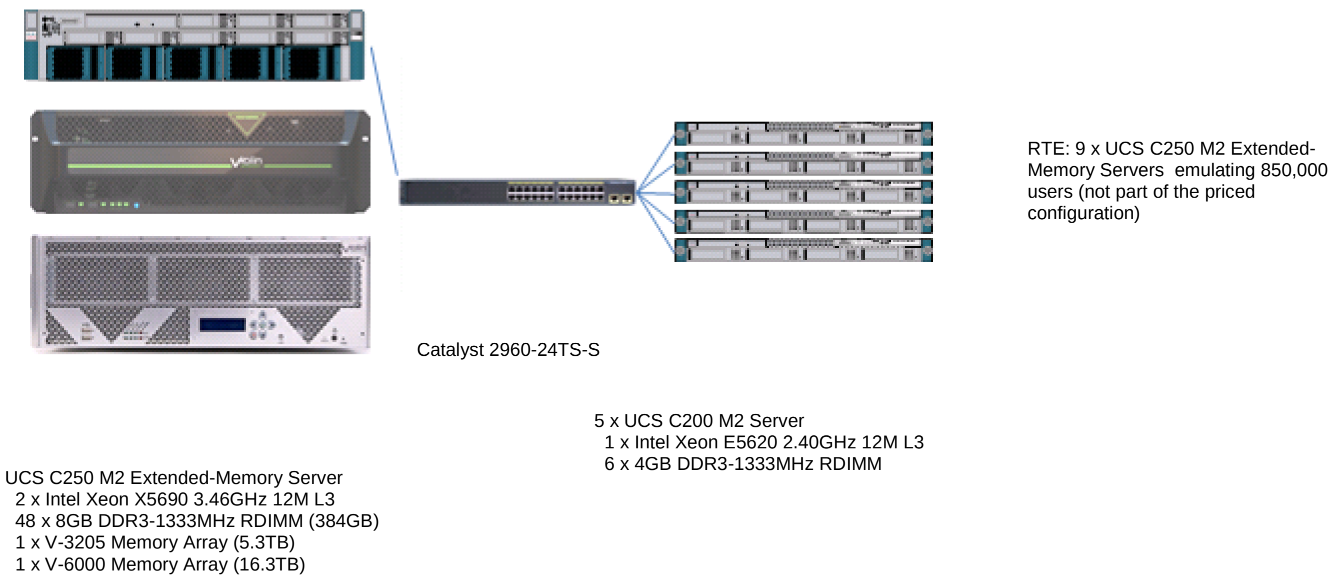
Appendix C contains the tunable parameters for the database, the operating system, and the transaction monitor.

0.4 Configuration Diagrams

Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences.

The configuration diagram for both the tested and priced system is depicted in Figure.

Figure 0.4 Benchmarked and Priced Configuration


RTE: 9 x UCS C250 M2 Extended-Memory Servers emulating 850,000 users (not part of the priced configuration)
Catalyst 2960-24TS-S
5 x UCS C200 M2 Server
1 x Intel Xeon E5620 2.40GHz 12M L3
6 x 4GB DDR3-1333MHz RDIMM
UCS C250 M2 Extended-Memory Server
2 x Intel Xeon X5690 3.46GHz 12M L3
48 x 8GB DDR3-1333MHz RDIMM (384GB)
1 x V-3205 Memory Array (5.3TB)
1 x V-6000 Memory Array (16.3TB)

1 Clause 1: Logical Database Design

1.1 Table Definitions

Listing must be provided for all table definition statements and all other statements used to set up the database.
Appendix B contains the code used to define and load the database tables.

1.2 Physical Organization of Database

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

The physical organization of the database is shown in the table below.

Table 1.2: Physical Organization of the Database

Controller	Array	Drives	RAID	Details
LSI MegaRAID SAS 9280-4i4e	Internal	2 x 300GB 10K SAS	RAID 1+0	OS
	VTrak J630sS	6 x 600GB 15K SAS	RAID 0	Redo Log
		8 x 2TB 7.2K NL SAS	RAID 0	60 Day Space
LSI MegaRAID SAS 9280-4i4e	VTrak J630sS	6 x 600GB 15K SAS	RAID 0	Redo Log
		8 x 2TB 7.2K NL SAS	RAID 0	60 Day Space
PCIe Host Interface Card	V-3205 Memory Array	5.25TB Flash	vRAID	Database Files
PCIe Host Interface Card	V-6000 Memory Array	16.0 TB Flash	vRAID	Database Files

1.3 Insert and Delete Operations

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

All insert and delete functions were verified to be fully operational during the entire benchmark.

1.4 Horizontal or Vertical Partitioning

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

No horizontal or vertical partitioning was used.

1.5 Replication or Duplication

Replication of tables, if used, must be disclosed. Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.

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

2 Clause 2: Transaction and Terminal Profiles

2.1 Random Number Generation

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

Random numbers were generated using the drand48() and lrand48() UNIX calls. These functions generate pseudo random numbers using the linear congruential algorithm and 48-bit integer arithmetic. The random number generators are initially seeded using the srand48() call.

2.2 Input/Output Screens

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

All screen layouts followed the specifications exactly.

2.3 Priced Terminal Feature

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 terminal attributes were verified by the auditor manually exercising each specification on a representative system.

2.4 Presentation Managers

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

No presentation manager software or intelligent terminal features were used. All screen processing was handled by the client system. All data manipulation was handled by the database server. The source code for the forms applications are included in Appendix A.

2.5 Transaction Statistics

Table 2.1 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 require.

Table 2. 1 Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines	99.00%
	Remote warehouse order lines	1.00%
	Rolled back transactions	1.00%
	Average items per order	10.00
Payment	Home warehouse	85.00%
	Remote warehouse	15.00%
	Accessed by last name	60.00%
Order Status	Accessed by last name	60.00%
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.94%
	Payment	43.03%
	Order status	4.01%
	Delivery	4.01%
	Stock level	4.02%

2.6 Queuing Mechanism

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

The queuing mechanism was implemented using Microsoft COM+. Delivery transactions were submitted asynchronously to Microsoft COM+ with control being returned immediately.

3 Clause 3: Transaction and System Properties

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

All ACID property tests conducted according to the specification.

3.1 Atomicity

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

3.1.1 Atomicity of Completed Transactions

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.

A row was randomly selected from the warehouse, district and customer tables, and the balances noted. A payment transaction was started with the same warehouse, district and customer identifiers and a known amount. The payment transaction was committed and the rows were verified to contain correctly updated balances.

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.

A row was randomly selected from the warehouse, district and customer tables, and the balances noted. A payment transaction was started with the same warehouse, district and customer identifiers and a known amount. The payment transaction was rolled back and the rows were verified to contain the original balances.

3.2 Consistency

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 Consistency conditions 1 through 4 demonstrated as follows:

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

An RTE run was executed against a freshly loaded database. After the run the consistency conditions 1 through 4 were tested using a script to issue queries to the database. All queries showed that the database was still in a consistent state. Consistency conditions were executed and verified

3.3 Isolation

Sufficient conditions must be enabled at either the system or application level to ensure the required isolation defined above (clause 3.4.1) is obtained.

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. Each included timestamps to demonstrate the concurrency of operations. The results of the queries were captured to files. The captured files were verified to demonstrate the required isolation had been met.

Isolation test 7 followed Case D, where T3 does not stall and no transaction is rolled back. T4 query of item price verifies to the changed prices of T3.

3.4 Durability

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)*

3.4.1 Durable Media Failure

3.4.1.1 Loss of Log Media and Data Media

These tests were conducted on a 12% scaled database. To demonstrate recovery from a permanent failure of durable medium containing TPC-C Log Media and Data Media, the following steps were executed:

1. The total number of New Orders was determined by the sum of d_next_o_id of all rows in the District table giving the beginning count. Consistency check 3 was verified.
2. The RTE was started with 12% load and allowed to run for 5 minutes at steady state.
3. The cable connecting one of the disk arrays containing the transaction log was disconnected.
4. The system continued running due to the fact that the transactions log files are mirrored across two different disk array using Oracle.
5. The run was allowed to continue for 5 minutes.
6. One flash module from each Violin Arrays containing the database files was disabled.
7. The system continued running due to the fact that the flash modules are protected by vRAID and active hot swap modules (4 per Violin Array)
8. The RTE run was stopped.
9. The total number of New Orders was determined by the sum of d_next_o_id of all rows in the District table giving the end count. Consistency conditions were executed and verified.
10. A RTE report was generated for the entire run time giving the number of New Orders successfully returned to the RTE.
11. It was verified that the number of New Orders from step 10 was equal to the difference between the counts from step 1 and step 9.
12. Samples were taken from the RTE files and used to query the database to demonstrate successful transactions had corresponding rows in the Orders table.

3.4.2 Instantaneous Interruption, Loss of Memory

As the loss of power erases the contents of memory, the instantaneous interruption and the loss of memory tests were combined into a single test. This test was executed on a fully scaled database. The following steps were executed:

1. The total number of New Orders was determined by the sum of d_next_o_id of all rows in the District table giving the beginning count. Consistency check 3 was verified.
2. The RTE was started at a full load with 850,000 users.
3. The test was allowed to run for a minimum of 5 minutes at steady state.
4. A system crash and loss of memory were induced by pulling the power plugs out of the database server.
5. The RTE was shutdown. Power was restored and the system restarted.
6. Oracle Database was restarted and performed an automatic recovery.
7. The total number of New Orders was determined by the sum of d_next_o_id of all rows in the District table giving the end count. Consistency conditions were executed and verified.
8. A RTE report was generated for the entire run time giving the number of New Orders successfully returned to the RTE.
9. It was verified that the number of New Orders from step 7 was equal to the difference between the counts from step 1 and step 9.
10. Samples were taken from the RTE files and used to query the database to demonstrate successful transactions had corresponding rows in the Orders table.

4 Clause 4: Scaling and Database Population

4. 1 Cardinality of Tables

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

Table 4.1 shows that number of rows for each table as they were built initially as per the TPC-C Specification.

Table 4.1 Number of Rows for Server

Table	Occurrences
Warehouse	86,000
District	860,000
Customer	2,580,000,000
History	2,580,000,000
Order	2,580,000,000
New Order	774,000,000
Order Line	25,801,336,176
Stock	8,600,000,000
Item	100,000
Unused Warehouse	1,000

4.2 Database Implementation

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

Oracle Database 11g Edition One is a relational DBMS.

Anonymous block PL/SQL and stored procedures were accessed through the Oracle Call Interface (OCI). Application code and stored procedures are included in Appendix A.

4.3 Distribution of Database Files

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

The physical organization of the database files and redo log files are shown in the table 5.2. One V-3200 Memory Array and one V-6000 Memory Array are used to host the database files. All the components in the Violin Memory Array are redundant and contain four hot-swappable flash modules. Data is protected by Violin vRAID technology. Each Violin Memory Array presented one physical volume to the operating system. Linux Logical Volume Manager (LVM) was used to create logical volumes on which database files created.

The VTrak J630sS arrays connected to the server using two separate LSI MegaRAID SAS 9280-4i4e controllers. Each VTrak J630sS array consisted of one RAID0 volume of eight disk drives. Linux LVM was used to create logical volumes. Battery Back Write Cache (BBWC) was enabled on the MegaRAID controller. Redo Log files were mirrored across the disk arrays at the database level to ensure no single point of disk drive, enclosure, and controller and write cache failures.

The database build scripts are included in Appendix B.

4.4 60 Day Space

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.

Table 4.4.1 60 Day Space

tpmC	**NWARE**					
1053100.32	86000					
SEGMENT	**TYPE**	**BLOCKS**	**BLOCK_SIZE**	**FIVE_PCT**	**DAILY_GROW**	**TOTAL**
CUSTCLUSTER	CLUSTER	603841229	4096	30192061	0	634033290
DB_STAT	SYS	524288	4096	0	0	524288
DISTCLUSTER	CLUSTER	870390	4096	43520	0	913910
HIST	TABLE	38150672	4096	0	7474695	45625367
ICUST1	INDEX	3854340	16384	192717	0	4047057
ICUST2	INDEX	7822953	16384	391148	0	8214101
IDIST	INDEX	107756	4096	5388	0	113144
IITEM	INDEX	2816	4096	141	0	2957
IORDR2	INDEX	5607420	16384	280371	0	5887791
ISTOK	INDEX	11006330	16384	550317	0	11556647
ITEMCLUSTER	CLUSTER	4223	4096	211	0	4434
IWARE	INDEX	27131	4096	1357	0	28488
NORDCLUSTER_QUEUE	CLUSTER	5229900	4096	261495	0	5491395
ORDRCLUSTER_QUEUE	CLUSTER	132550784	16384	0	25970097	158520881
STOKCLUSTER	CLUSTER	615009168	4096	30750458	0	645759626

SYSAUX	SYS	30720	4096	0	0	30720
SYSTEM	SYS	102400	4096	0	0	102400
SYS_IQ0000013285$$	INDEX	581760	16384	29088	0	610848
SYS_IQ0000013289$$	INDEX	70200	4096	3510	0	73710
WARECLUSTER	CLUSTER	92276	4096	4614	0	96890
SEGMENT	**BLOCKS**	**BLOCK_SIZE**	**REQUIRED**	**STATIC**	**DYNAMIC**	**OVERSIZE**
CUSTCLUSTER	672163840	4096	634033290	634033290	0	38130550
DB_STAT	524288	4096	524288	524288	0	0
DISTCLUSTER	983040	4096	913910	913910	0	69130
HIST	55541760	4096	45625367	0	38150672	9916393
ICUST1	4147200	16384	4047057	4047057	0	100143
ICUST2	8652800	16384	8214101	8214101	0	438699
IDIST	128000	4096	113144	113144	0	14856
IITEM	5120	4096	2957	2957	0	2163
IORDR2	8499200	16384	5887791	5887791	0	2611409
ISTOK	11673600	16384	11556647	11556647	0	116953
ITEMCLUSTER	5120	4096	4434	4434	0	686
IWARE	51200	4096	28488	28488	0	22712
NORDCLUSTER_QUEUE	7536640	4096	5491395	5491395	0	2045245
ORDRCLUSTER_QUEUE	187650560	16384	158520881	0	132550784	29129679
STOKCLUSTER	659046400	4096	645759626	645759626	0	13286774
SYSAUX	30720	4096	30720	30720	0	0
SYSTEM	102400	4096	102400	102400	0	0
SYS_IQ0000013285$$	187650560	16384	610848	610848	0	187039712
SYS_IQ0000013289$$	7536640	4096	73710	73710	0	7462930
WARECLUSTER	128000	4096	96890	96890	0	31110
STATIC	**DYNAMIC**	**OVERSIZE**	**DAILY_GROW**	**DAILY_SPRE**	**SPACE60**	**8 Hr Log Space**
5633764112	2273415232	3794915716	445420332	0	32359000000	5816320

Table 4.4.2 Available Space

	Total Capacity (TB)*	RAID	Formatted Capacity (TB)
V-3205 Memory Array	5.2	vRAID	10.6
V-6000 Memory Array	16	vRAID	2.6
VTrak J630sS 1 Array 1	4.7	RAID0	4.23
VTrak J630sS 2 Array 2	16	RAID0	14.4
VTrak J630sS 2 Array 1	4.7	RAID0	4.23
VTrak J630sS 2 Array 2	16	RAID0	14.4
	*1 TB=1024 GB		Total available for Data= 39.06 Total available for Log: 8.46

5 Clause 5: Performance Metrics

5.1 TPC Benchmark C Metrics

TPC Benchmark C Metrics are reported in the executive summary section of this document.

5.2 Response Times

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

See Table 5.2

Table 5.1: Response Times

Type	Average	Maximum	90th %
New-Order	0.368	3.014	0.994
Payment	0.367	2.696	0.992
Order-Status	0.368	2.568	0.992
Interactive Delivery	0.334	2.532	0.934
Deferred Delivery	0.098	3.946	0.111
Stock-Level	0.381	2.522	1.007
Menu	0.335	3.117	0.936

5.3 Keying and Think Times

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

See Table 5.3

Table 5.3: Keying Times/Think Times

Type	Minimum	Average	Maximum
New-Order	18.000/0.000	18.000/12.084	18.069/120.31
Payment	3.000/0.000	3.000/12.089	3.077/120.31
Order-Status	2.000/0.000	2.000/10.084	2.071/100.30
Interactive Delivery	2.000/0.000	2.000/5.057	2.069/50.30
Stock-Level	2.000/0.000	2.000/5.055	2.077/50.30

5.4 Response Time Frequency Distribution and Performance Curves

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

See Figure 5.4.1, 5.4.2, 5.4.3, 5.4.4 and 5.4.5

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

See Figure 5.4.6

Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type.

See Figure 5.4.7

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

See Figure 5.4.8

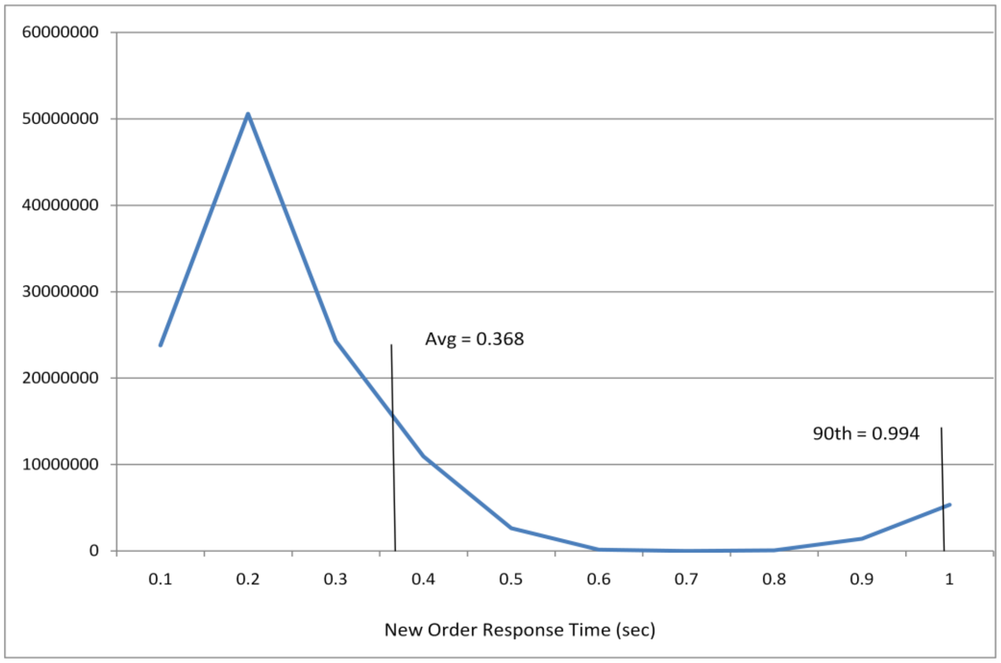


Figure 5.4.1: Response Times Frequency Distribution for New Order Transactions

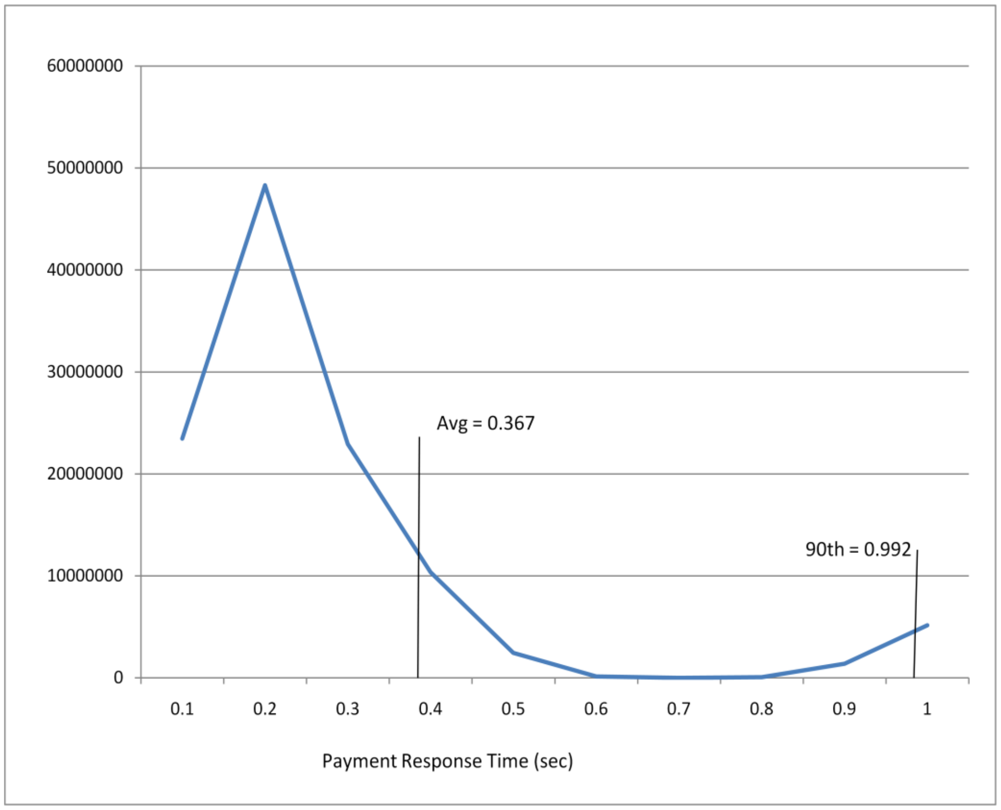


Figure 5.4.2: Response Times Frequency Distribution for Payment Transactions

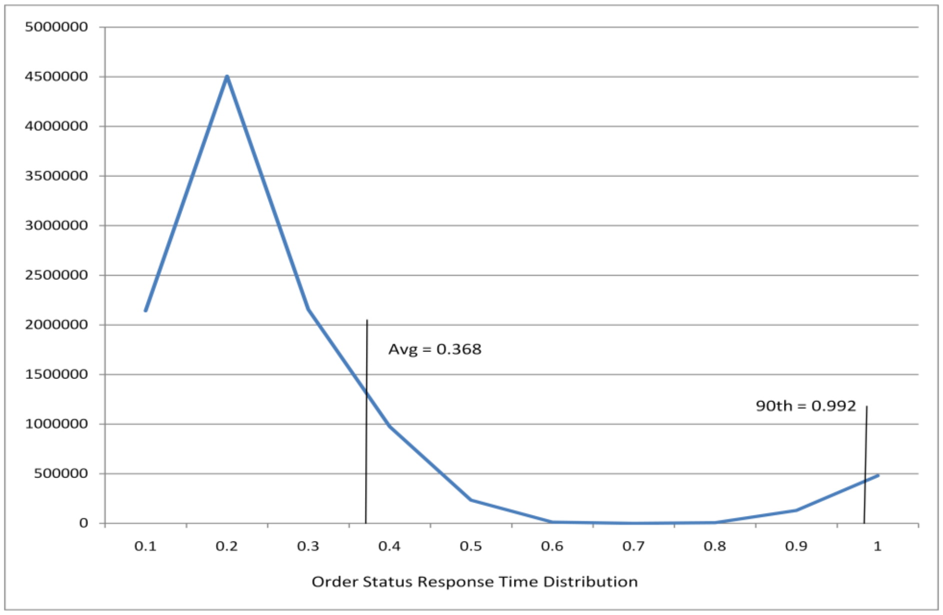


Figure 5.4.3: Response Times Frequency Distribution for Order Status Transactions

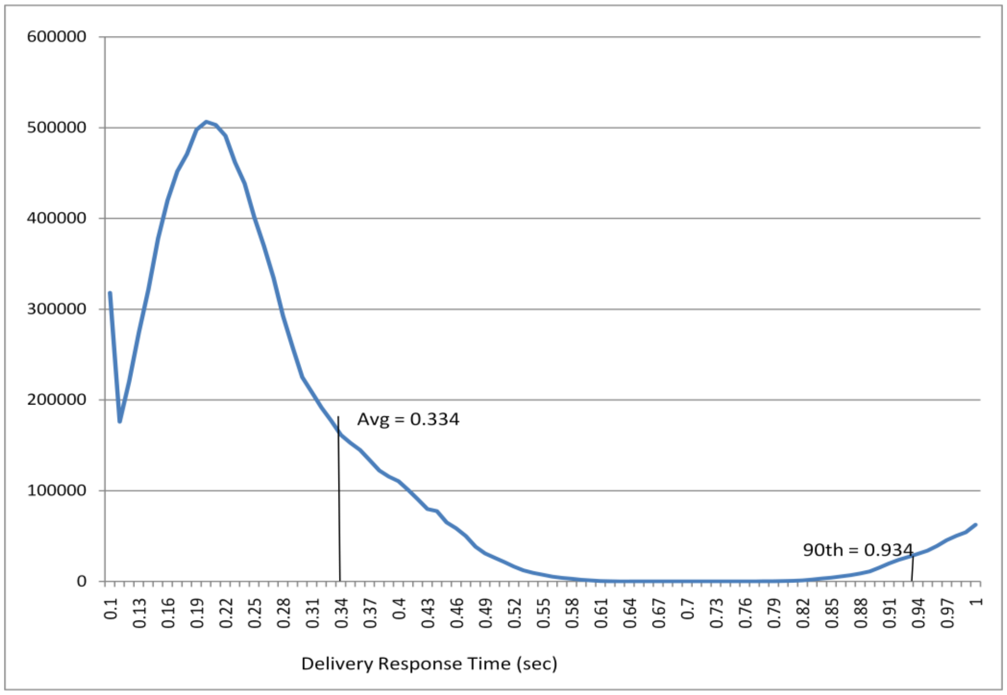


Figure 5.4.4: Response Times Frequency Distribution for Delivery Transactions

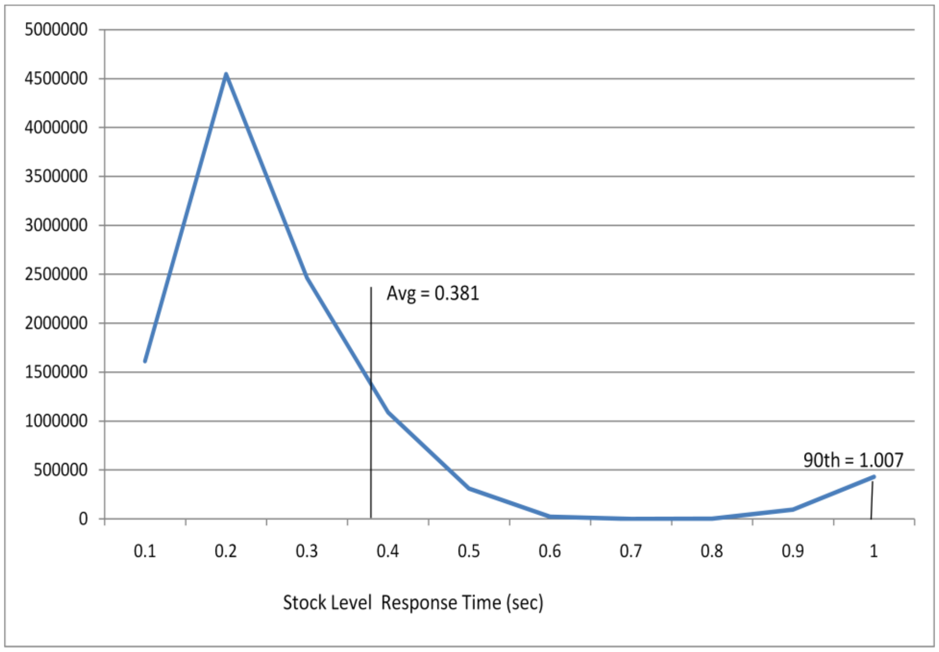


Figure 5.4.5: Response Times Frequency Distribution for Stock Level Transactions

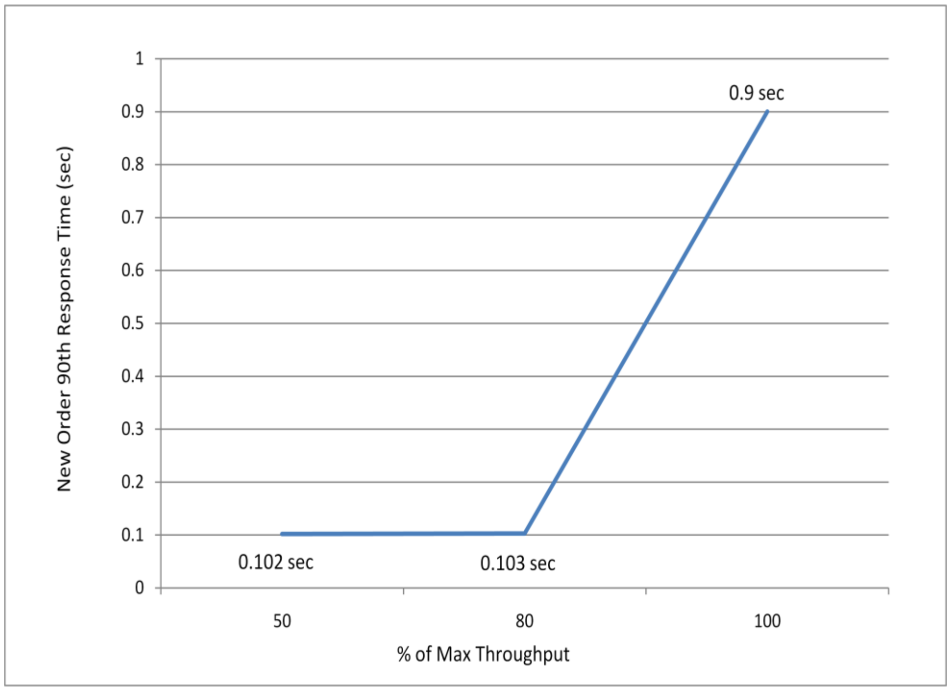


Figure 5.4.6: Response Time versus Throughput

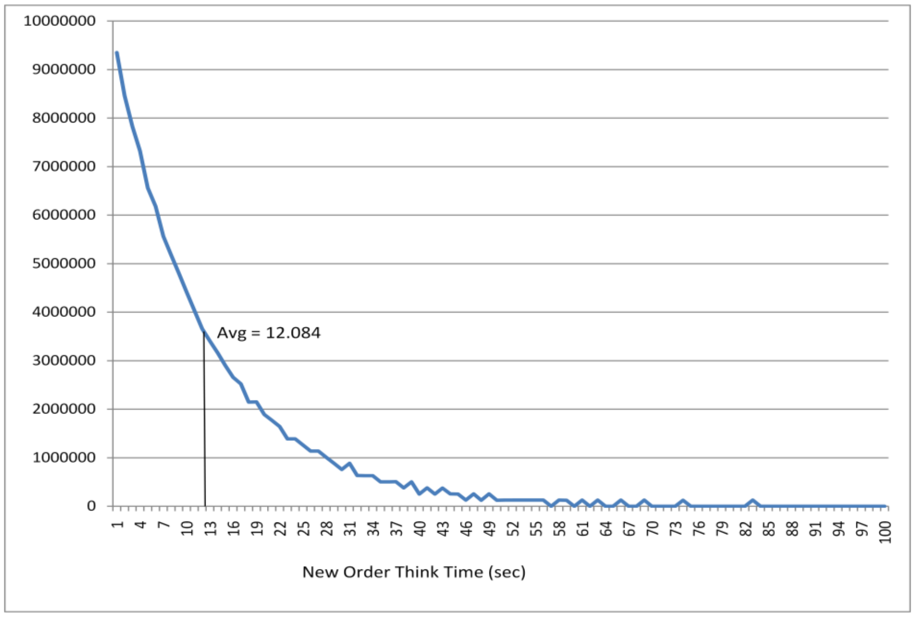


Figure 5.4.7: Think Times distribution for New Order Transactions

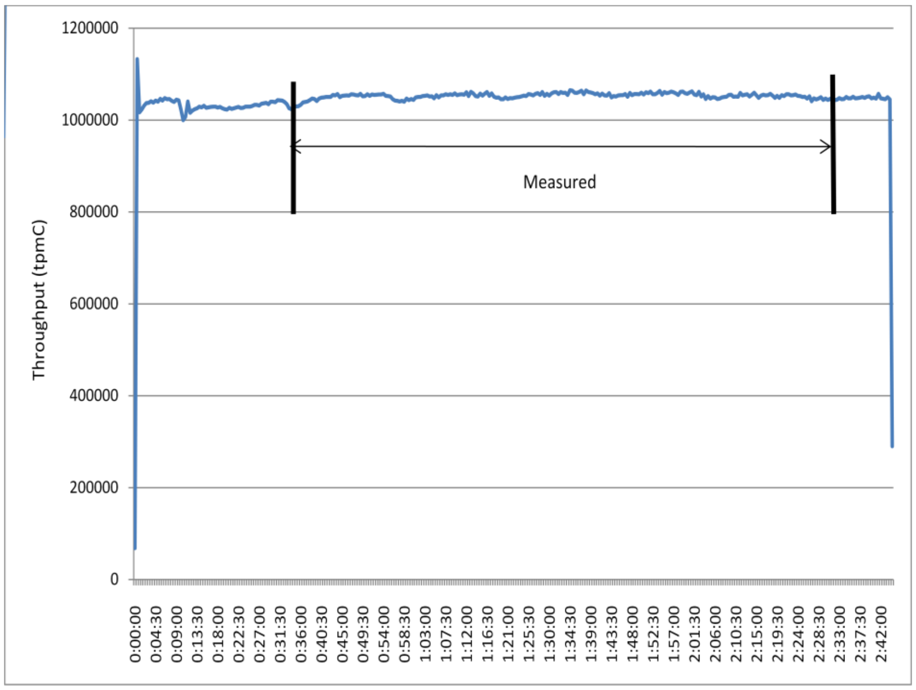


Figure 5.4.8: Throughput versus Time

5.5 Steady State Determination

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

Steady state was determined using real time monitor utilities from the RTE. Steady state was further confirmed by the throughput data collected during the run and graphed in section 5.4.

5.6 Work Performed During Steady State

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

A two hour measurement interval was used to demonstrate the work normally performed during a sustained test.

TPC-C transactions are implemented in accordance to the TPC-C Benchmark Specification. During the performance run, transactions are submitted by the RTE in accordance to the described mix, keying times and think times of the TPC-C benchmark Specification. Transactions are submitted by emulated user via HTTP. After each transaction the emulated user waits for a randomly generated think time before selecting the next transaction. All timings are recorded by the RTE. The response time is measured from the start of the transaction until the last byte is received by the RTE.

Oracle Database records transactions in the database tables and the redo log files. Writes to the database may be cached in the the main memory of the server before being written to durable media. Checkpoints are initiated once the log file filled and rolled over to the next log file.

5.7 Measurement Period Duration

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

The reported measured interval was 7200 seconds.

5.8 Transaction Statistics

The percentage of the total mix for each transaction type must be disclosed. The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed. The average number of order-lines entered per New-Order transaction must be disclosed. The percentage of remote order lines per New-Order transaction must be disclosed. The percentage of remote Payment transactions must be disclosed. The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed. The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.

Table 5.8: Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines	99.00%
	Remote warehouse order lines	1.00%
	Rolled back transactions	1.00%
	Average items per order	10.00
Payment	Home warehouse	85.00%
	Remote warehouse	15.00%
	Accessed by last name	60.00%
Order Status	Accessed by last name	60.00%
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.94%
	Payment	43.03%
	Order status	4.01%
	Delivery	4.01%
	Stock level	4.02%

5.9 Checkpoints

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

Five complete checkpoints occurred during the measurement period.

6 Clause 6: SUT, Driver and Communication

6.1 Remote Terminal Emulator (RTE)

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

The RTE software used was developed internally. The RTE simulates terminal users , generated transactions accordance to the described mix , keying times and think times, and records response times.

Nine UCS C250 M2 Extended-Memory Servers are used to emulating 850,000 users.

6.2 Emulated Components

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to the priced system. The results of the test described in Clause 6.6.3.4 must be disclosed.

No components were emulated.

6.3 Functional Diagrams

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

The diagram in Section 0.4 shows the tested and priced benchmark configurations.

6.4 Networks

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

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

Section 1 of this report contains detailed diagrams of both the benchmark configuration and the priced configuration. The network between the clients and the database server was 1GigE.

6.5 Operator Intervention

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

No operator intervention is required to sustain eight hours of the reported throughput.

7 Clause 7: Pricing

7. 1 Hardware and Software Pricing

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

The details of the hardware and software are reported in the executive summary section of this document.

7.2 Three Year Price

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 are reported in the executive summary section of this document. .

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 are available on the date of publication.

8 Auditor' Information and Attestation Letter

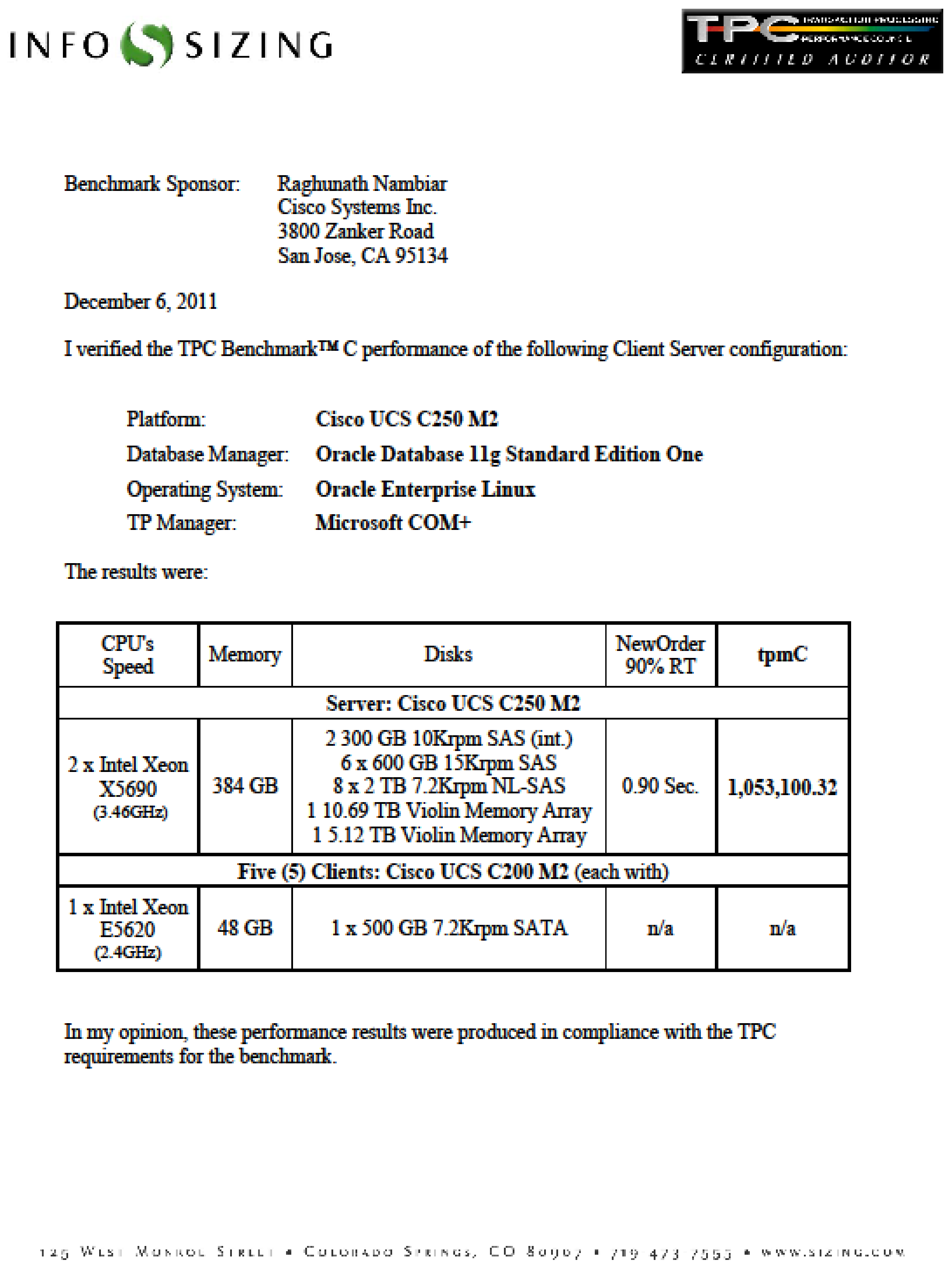
8.1 Auditor's Report

The auditor's agency name, address, phone number, and Attestation letter with a brief audit summary report indicating compliance must be included in the full disclosure report. A statement should be included specifying who to contact in order to obtain further information regarding the audit process.

The auditor's letter is included in the following section.

This benchmark was audited by:

Francois Raab
InfoSizing, Inc.
531 Crystal Hills Blvd
Manitou Springs, CO 80829
Phone: 719-473-7555.



INFO SIZING

TPC Transaction Processing Performance Council CERTIFIED AUDITOR

Benchmark Sponsor: Raghunath Nambiar
Cisco Systems Inc.
3800 Zanker Road
San Jose, CA 95134

December 6, 2011

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

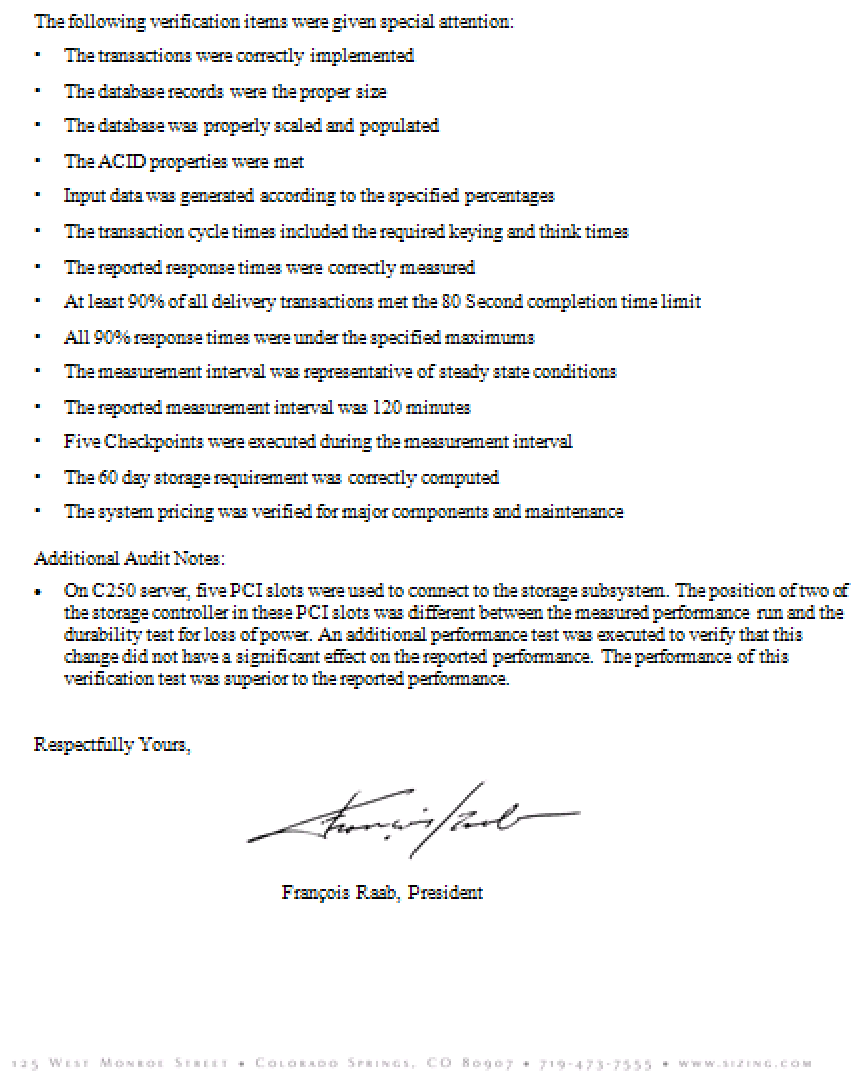
Platform:	**Cisco UCS C250 M2**
Database Manager:	**Oracle Database 11g Standard Edition One**
Operating System:	**Oracle Enterprise Linux**
TP Manager:	**Microsoft COM+**

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% RT	tpmC
Server: Cisco UCS C250 M2				
2 x Intel Xeon X5690 (3.46GHz)	384 GB	2 300 GB 10Krpm SAS (int.) 6 x 600 GB 15Krpm SAS 8 x 2 TB 7.2Krpm NL-SAS 1 10.69 TB Violin Memory Array 1 5.12 TB Violin Memory Array	0.90 Sec.	**1,053,100.32**
Five (5) Clients: Cisco UCS C200 M2 (each with)				
1 x Intel Xeon E5620 (2.4GHz)	48 GB	1 x 500 GB 7.2Krpm SATA	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for 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 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 120 minutes
- Five Checkpoints were executed 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:

- On C250 server, five PCI slots were used to connect to the storage subsystem. The position of two of the storage controller in these PCI slots was different between the measured performance run and the durability test for loss of power. An additional performance test was executed to verify that this change did not have a significant effect on the reported performance. The performance of this verification test was superior to the reported performance.

Respectfully Yours,

[illegible]

François Raab, President

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

9 Appendix A: Source Code

- - - - - - - - - - - - - - - - - -
DBConnection.cpp
- - - - - - - - - - - - - - - - - -

```
// DBConnection.cpp : Defines the entry point for the DLL application.
//

#include "stdafx.h"
#include "DBConnection.h"

//#define USE_CRITICAL_SECTION
//#define OPS_LOGIN
//#define CONNECTION_MUTEX
//#define DEBUG
//#define DEBUG_DETAIL
//#define LOOPBACK

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                                                                          )
{
  char string[MAXLEN];

  if (ul_reason_for_call == DLL_PROCESS_ATTACH) {
        int i;

                DisableThreadLibraryCalls((HMODULE)hModule);

                GetModuleFileName((HMODULE)hModule, DllPath,
MAXLEN-1);
                if (DllPath[0]=='\\' && DllPath[1]=='\\' && DllPath[2]=='?' &&
DllPath[3]=='\\')
                                strcpy(DllPath, DllPath+4);
        for (i=strlen(DllPath); DllPath[i]!='\\' && i; i--);
                DllPath[i]='\0';
                sprintf(LogFile, "%s\\%s", DllPath, LogName);
                sprintf(InitFile, "%s\\%s", DllPath, InitName);
    sprintf(DelLogFile, "%s\\%s", DllPath, DelLogName);

    if (!SetCurrentDirectory(DllPath)) {
      userlog("Cannot change current directory to %s, Error: %n", DllPath,
GetLastError());
      return FALSE;
    }

    if ((TlsPtr = TlsAlloc()) == 0xFFFFFFFF) {
      userlog("Error during TlsAlloc\n");
      return FALSE;
    }


    readInit(string, "DBConnections", Default_DBConnections);
                DBConnections = atoi(string);
                userlog("number of DBConnections is %d\n", DBConnections);

                TotalLoop=DBConnections*2;



                DBExecution_lock=(HANDLE*)malloc(sizeof(HANDLE)*DB
Connections);
#ifdef USE_CRITICAL_SECTION
                DBExecution_CS=(CRITICAL_SECTION*)malloc(sizeof(CRI
TICAL_SECTION)*DBConnections);
#endif
                for (i=0; i<DBConnections; i++) {
#ifdef USE_CRITICAL_SECTION
                                InitializeCriticalSection(&DBExecution_CS[i]);
#endif
                                if ((DBExecution_lock[i]=CreateMutex(NULL,
FALSE, NULL))==NULL) {
                                                userlog("Cannot create mutex :
DBExecution_lock[%d]\n", i);
                                                return FALSE;
                                }
                }

                if (initializeDBExecutionPool() != TRUE) {
                                userlog("initializeDBExecutionPool failed\n");
                                return FALSE;
                }

                if ((waitIdle = CreateEvent(NULL, FALSE, FALSE, "Wait Idle
Event")) == NULL) {
                                userlog("Cannot create event : waitIdle\n");
                                return FALSE;
                }

                ready=1;

  }
  else if (ul_reason_for_call == DLL_PROCESS_DETACH) {
                 int i;

    if ((TlsFree(TlsPtr)) == NULL) {
      userlog("Error during TlsFree\n");
      return FALSE;
    }
                for (i=0; i<DBConnections; i++) {
                                ((DBExecution *)(DBExecution_pool[i].pointer))-
>TPCexit();
                                free(DBExecution_pool[i].pointer);
                }
                free (DBExecution_pool);
                CloseHandle(waitIdle);

                for (i=0; i<DBConnections; i++) {
                                CloseHandle(DBExecution_lock[i]);
#ifdef USE_CRITICAL_SECTION
                                DeleteCriticalSection(&DBExecution_CS[i]);
#endif
                }

  }

  return TRUE;
}

void initDelLog(int DelThreads)
{
                char filename[MAXLEN];

                DelFiles=(FILE **)malloc(sizeof(FILE *)*DelThreads);
                for (int i=0; i<DelThreads; i++) {
                  sprintf(filename, "%s%d", DelLogFile, i);

                  if ((DelFiles[i]=fopen(filename, "a"))==(FILE *) NULL) {
                                userlog("Can't open file : %s\n", filename);
                                exit(-1);
                  }
                  setvbuf(DelFiles[i], NULL, _IOFBF, 102400);
		}
}

void endDelLog(int DelThreads)
{
		for (int i=0; i<DelThreads; i++) {
				fclose(DelFiles[i]);
		}
		free(DelFiles);
}

/**************************************************************
**********************************
*  Execute transactions                                          *
**************************************************************
**********************************/

#ifndef LOOPBACK

int mod_tpcc_neworder(T_neworder_data *output)
{
		int i;
#ifdef CONNECTION_MUTEX
		HANDLE *mutexptr=NULL;
#endif
   DBExecution_pool_info* ptr;

		DBExecution *dbexec;
		struct newstruct input;

		input.newin.w_id = output->w_id;
		input.newin.d_id = output->d_id;
		input.newin.c_id = output->c_id;

		for (i=0; i<output->o_ol_cnt; i++) {
		   input.newin.ol_i_id[i] = output->o_orderline[i].ol_i_id;
				input.newin.ol_supply_w_id[i] = output-
>o_orderline[i].ol_supply_w_id;
		   input.newin.ol_quantity[i] = output-
>o_orderline[i].ol_quantity;
		}

		for (; i<15; i++) {
		   input.newin.ol_i_id[i] = 0;
				input.newin.ol_supply_w_id[i] = 0;
		   input.newin.ol_quantity[i] = 0;
		}

#ifdef CONNECTION_MUTEX
		ptr=findIdleDBExecution(mutexptr);
#else
		ptr=findIdleDBExecution();
#endif
		dbexec=(DBExecution *)(ptr->pointer);
//		ptr->neworder_count++;


		if (dbexec->TPCnew(&input) == -1) {
				convert_status(output->txn_status, dbexec-
>execstatus);
#ifdef CONNECTION_MUTEX
				freeDBExecution(ptr, mutexptr);
#else
				freeDBExecution(ptr);
#endif
				userlog("TPCnew returns -1\n");
				return SUCCESS;
		} else {
		   output->txn_status = DB_RETURN_OCI_SUCCESS;
		}
		output->status = dbexec->status;

#ifdef CONNECTION_MUTEX
		freeDBExecution(ptr, mutexptr);
#else
		freeDBExecution(ptr);
#endif

		output->o_id = input.newout.o_id;
		output->o_ol_cnt = input.newout.o_ol_cnt;
		output->c_discount = input.newout.c_discount;
		output->w_tax = input.newout.w_tax;
		output->d_tax = input.newout.d_tax;
		output->total_amount = input.newout.total_amount;
		strncpy(output->o_entry_d.DateString,
input.newout.o_entry_d,20);
		strncpy(output->c_last, input.newout.c_last,17);
		strncpy(output->c_credit, input.newout.c_credit,3);
		for (i=0; i<output->o_ol_cnt; i++) {
				output->o_orderline[i].ol_amount =
input.newout.ol_amount[i];
				output->o_orderline[i].i_price =
input.newout.i_price[i];
				output->o_orderline[i].s_quantity =
input.newout.s_quantity[i];
				output->o_orderline[i].b_g[0] =
input.newout.brand_generic[i];
				strncpy(output->o_orderline[i].i_name,
input.newout.i_name[i], 25);
		}

		return SUCCESS;
}


int mod_tpcc_payment(T_payment_data *output)
{
#ifdef CONNECTION_MUTEX
		HANDLE *mutexptr=NULL;
#endif
   DBExecution_pool_info* ptr;
		DBExecution *dbexec;
		struct paystruct input;

		input.payin.w_id = output->w_id;
		input.payin.d_id = output->d_id;
		input.payin.c_w_id = output->c_w_id;
		input.payin.c_d_id = output->c_d_id;
		input.payin.bylastname = output->by_last_name;
		input.payin.h_amount = (int)(output->h_amount * 100);

		if (input.payin.bylastname) {
				input.payin.c_id = 0;
				strncpy(input.payin.c_last, output->c_last, 17);
				input.payin.c_last[16]='\0';
		} else {
				input.payin.c_id = output->c_id;
				input.payin.c_last[0]='\0';
		}

#ifdef CONNECTION_MUTEX
		ptr=findIdleDBExecution(mutexptr);
#else
		ptr=findIdleDBExecution();
#endif
		dbexec=(DBExecution *)(ptr->pointer);
//		ptr->payment_count++;

		if (dbexec->TPCpay(&input) == -1) {
				convert_status(output->txn_status, dbexec-
>execstatus);
#ifdef CONNECTION_MUTEX
		freeDBExecution(ptr, mutexptr);
#else
		freeDBExecution(ptr);
#endif
				userlog("TPCpay returns -1\n");
				return SUCCESS;
		} else {
				output->txn_status =
DB_RETURN_OCI_SUCCESS;
		}

#ifdef CONNECTION_MUTEX
		freeDBExecution(ptr, mutexptr);
#else
		freeDBExecution(ptr);
#endif

		strncpy(output->w_street_1, input.payout.w_street_1, 21);
		strncpy(output->w_street_2, input.payout.w_street_2, 21);
		strncpy(output->w_city, input.payout.w_city, 21);
		strncpy(output->w_state, input.payout.w_state, 3);
		strncpy(output->w_zip, input.payout.w_zip, 10);
		strncpy(output->d_street_1, input.payout.d_street_1, 21);
		strncpy(output->d_street_2, input.payout.d_street_2, 21);
		strncpy(output->d_city, input.payout.d_city, 21);
		strncpy(output->d_state, input.payout.d_state, 3);
		strncpy(output->d_zip, input.payout.d_zip, 10);
		output->c_id = input.payout.c_id;
		strncpy(output->c_first, input.payout.c_first, 17);
		strncpy(output->c_middle, input.payout.c_middle, 3);
		strncpy(output->c_last, input.payout.c_last, 17);
		strncpy(output->c_street_1, input.payout.c_street_1, 21);
		strncpy(output->c_street_2, input.payout.c_street_2, 21);
		strncpy(output->c_city, input.payout.c_city, 21);
		strncpy(output->c_state, input.payout.c_state, 3);
		strncpy(output->c_zip, input.payout.c_zip, 10);
		strncpy(output->c_phone, input.payout.c_phone, 17);
		strncpy(output->c_credit, input.payout.c_credit, 3);
		strncpy(output->c_since.DateString, input.payout.c_since, 11);
		strncpy(output->h_date.DateString, input.payout.h_date, 20);
		strncpy(output->c_data, input.payout.c_data, 200);
		output->c_credit_lim = input.payout.c_credit_lim;
		output->c_discount = input.payout.c_discount;
		output->c_balance = input.payout.c_balance;

		return SUCCESS;
}



int mod_tpcc_delivery(T_delivery_data *output, int id)
{
#ifdef CONNECTION_MUTEX
		HANDLE *mutexptr=NULL;
#endif
   DBExecution_pool_info *ptr;
		DBExecution *dbexec;;
		struct delstruct input;

		input.delin.w_id = output->w_id;
		input.delin.plsqlflag = 1;
		input.delin.o_carrier_id = output->o_carrier_id;

#ifdef CONNECTION_MUTEX
		ptr=findIdleDBExecution(mutexptr);
#else
		ptr=findIdleDBExecution();
#endif
		dbexec=(DBExecution *)(ptr->pointer);
//		ptr->delivery_count++;

		if (dbexec->TPCdel(&input) == -1) {
				convert_status(output->txn_status, dbexec-
>execstatus);
#ifdef CONNECTION_MUTEX
				freeDBExecution(ptr, mutexptr);
#else
				freeDBExecution(ptr);
#endif
				userlog("TPCdel returns -1\n");
				return SUCCESS;
		} else {
				output->txn_status =
DB_RETURN_OCI_SUCCESS;
		}

		output->complete_time = GetTickCount();
		for (int i=0; i<10; i++)
				output->o_id[i]=dbexec->del_o_id[i];

#ifdef CONNECTION_MUTEX
		freeDBExecution(ptr, mutexptr);
#else
		freeDBExecution(ptr);
#endif

		write_delivery_log(output, id);

		return SUCCESS;
}


int mod_tpcc_orderstatus(T_orderstatus_data *output)
{
#ifdef CONNECTION_MUTEX
		HANDLE *mutexptr=NULL;
#endif
   DBExecution_pool_info* ptr;
		DBExecution *dbexec;
		struct ordstruct input;

		input.ordin.w_id = output->w_id;
		input.ordin.d_id = output->d_id;
		input.ordin.bylastname = output->by_last_name;
		if (input.ordin.bylastname) {
				input.ordin.c_id = 0;
				strncpy(input.ordin.c_last, output->c_last, 17);
				input.ordin.c_last[16]='\0';
		}
		else {
				input.ordin.c_id = output->c_id;
				input.ordin.c_last[0]='\0';
		}

#ifdef CONNECTION_MUTEX
		ptr=findIdleDBExecution(mutexptr);
#else
		ptr=findIdleDBExecution();
#endif
		dbexec=(DBExecution *)(ptr->pointer);
//		ptr->orderstatus_count++;


		if (dbexec->TPCord(&input) == -1) {
				convert_status(output->txn_status, dbexec-
>execstatus);
#ifdef CONNECTION_MUTEX
```

```
				freeDBExecution(ptr, mutexptr);
#else
				freeDBExecution(ptr);
#endif
				userlog("TPCord returns -1\n");
				return SUCCESS;
		} else {
				output->txn_status =
DB_RETURN_OCI_SUCCESS;
		}

#ifdef CONNECTION_MUTEX
		freeDBExecution(ptr, mutexptr);
#else
		freeDBExecution(ptr);
#endif

		output->c_id = input.ordout.c_id;
		strncpy(output->c_last, input.ordout.c_last, 17);
		strncpy(output->c_first, input.ordout.c_first, 17);
		strncpy(output->c_middle, input.ordout.c_middle, 3);
		strncpy(output->o_entry_d.DateString, input.ordout.o_entry_d,
20);
		output->c_balance = input.ordout.c_balance;
		output->o_id = input.ordout.o_id;
		output->o_carrier_id = input.ordout.o_carrier_id;
		output->o_ol_cnt = input.ordout.o_ol_cnt;
		for (int i=0; i<output->o_ol_cnt; i++) {
				output->o_orderline[i].ol_supply_w_id =
input.ordout.ol_supply_w_id[i];
				output->o_orderline[i].ol_i_id =
input.ordout.ol_i_id[i];
				output->o_orderline[i].ol_quantity =
input.ordout.ol_quantity[i];
				output->o_orderline[i].ol_amount =
input.ordout.ol_amount[i];
				strncpy(output-
>o_orderline[i].ol_delivery_d.DateString, input.ordout.ol_delivery_d[i],
11);
		}

		return SUCCESS;
}


int mod_tpcc_stocklevel(T_stocklevel_data *output)
{
#ifdef CONNECTION_MUTEX
		HANDLE *mutexptr=NULL;
#endif
   DBExecution_pool_info* ptr;
		DBExecution *dbexec;
		struct stostruct input;

		input.stoout.low_stock=-123;
		input.stoin.w_id = output->w_id;
		input.stoin.d_id = output->ld_id;
		input.stoin.threshold = output->threshold;

#ifdef CONNECTION_MUTEX
		ptr=findIdleDBExecution(mutexptr);
#else
		ptr=findIdleDBExecution();
#endif
		dbexec=(DBExecution *)(ptr->pointer);
//		ptr->stocklevel_count++;

		if (dbexec->TPCsto(&input) == -1) {
				convert_status(output->txn_status, dbexec-
>execstatus);
#ifdef CONNECTION_MUTEX
				freeDBExecution(ptr, mutexptr);
#else
				freeDBExecution(ptr);
#endif
				userlog("TPCsto returns -1\n");
				return SUCCESS;
   } else {
				output->txn_status =
DB_RETURN_OCI_SUCCESS;
		}

#ifdef CONNECTION_MUTEX
		freeDBExecution(ptr, mutexptr);
#else
		freeDBExecution(ptr);
#endif

		output->low_stock = input.stoout.low_stock;

		return SUCCESS;
}

#endif



void write_delivery_log(T_delivery_data *pdata, int threadId)
{
		fprintf(DelFiles[threadId],
				"%d/%d/%d %d:%d:%d.%d %ld %ld %ld %d %d
%d %d %d %d %d %d %d %d %d %d %d\n",
				pdata->enqueue_date_time.wMonth, pdata-
>enqueue_date_time.wDay,
				pdata->enqueue_date_time.wYear, pdata-
>enqueue_date_time.wHour,
				pdata->enqueue_date_time.wMinute, pdata-
>enqueue_date_time.wSecond,
				pdata->enqueue_date_time.wMilliseconds, pdata-
>enqueue_time,
				pdata->complete_time, pdata->complete_time-pdata-
>enqueue_time, pdata->w_id,
				pdata->ld_id, pdata->o_carrier_id, pdata->o_id[0],
pdata->o_id[1],
				pdata->o_id[2], pdata->o_id[3], pdata->o_id[4],
pdata->o_id[5],
				pdata->o_id[6], pdata->o_id[7], pdata->o_id[8],
pdata->o_id[9]);
}


#ifdef CONNECTION_MUTEX
int freeDBExecution(DBExecution_pool_info *ptr, HANDLE *mutexptr)
#else
int freeDBExecution(DBExecution_pool_info *ptr)
#endif
{
		ptr->current_status = IDLE;

#ifdef DEBUG_DETAIL
		userlog("Thread %d release connection\n",
GetCurrentThreadId());
#endif


#ifdef CONNECTION_MUTEX
		if (mutexptr==NULL)
				userlog("Thread %d has mutexptr=NULL\n",
GetCurrentThreadId());
		ReleaseMutex((*mutexptr));
```

```
#endif
	if (!SetEvent(waitIdle)) {
		userlog("Error on SetEvent, in function: free
DBExection\n");
		return FALSE;
	}

	return TRUE;
}


#ifdef CONNECTION_MUTEX
DBExecution_pool_info* findIdleDBExecution(HANDLE *mutexptr)
#else
DBExecution_pool_info* findIdleDBExecution()
#endif
{
	int current=GetCurrentThreadId() % DBConnections;

#ifdef DEBUG
	findDBExecutionCall++;
#endif

	while (1) {

		for (int count=0; count<TotalLoop; count++) {

			if
(DBExecution_pool[current].current_status == IDLE) {
#ifdef USE_CRITICAL_SECTION
				EnterCriticalSection(
&DBExecution_CS[current] );
#else

	switch(WaitForSingleObject(DBExecution_lock[current], 0)) {

					case
WAIT_ABANDONED:
#ifdef DEBUG

	userlog("connection mutex returns WAIT_ABANDONED\n");
#endif


					case
WAIT_OBJECT_0:

#ifdef DEBUG_DETAIL

	userlog("Thread %d get connection: %d\n",
GetCurrentThreadId(), current);
#endif
#endif
						if
(DBExecution_pool[current].current_status == IDLE) {

	DBExecution_pool[current].current_status = IN_USE;
#ifdef USE_CRITICAL_SECTION

	LeaveCriticalSection( &DBExecution_CS[current] );
#else
#ifndef CONNECTION_MUTEX

	ReleaseMutex(DBExecution_lock[current]);
#else

	mutexptr=&(DBExecution_lock[current]);
#endif
#endif

	TlsSetValue(TlsPtr, (void *)
DBExecution_pool[current].pointer);


	return &(DBExecution_pool[current]);
						}
						else {
#ifdef USE_CRITICAL_SECTION

	LeaveCriticalSection( &DBExecution_CS[current] );
#else

	ReleaseMutex(DBExecution_lock[current]);
#endif
#ifdef DEBUG

	userlog("get connection mutex, but current_status is not
IDLE\n");
#endif
						}
#ifndef USE_CRITICAL_SECTION
						break;

					case
WAIT_TIMEOUT:
						break;


					default:

	userlog("Error on WaitForSingleObject, DBExecution\n");
						return
NULL;
				}
#endif


			}

			current++;
			if (current==DBConnections) current=0;
		}

#ifdef DEBUG
		findDBExecutionWait++;
		if (findDBExecutionWait !=0 &&
findDBExecutionWait % 100000 == 0)
			userlog("wait: %d,  total call: %d\n",
findDBExecutionWait, findDBExecutionCall);
#endif

		if ((WaitForSingleObject(waitIdle, INFINITE)) !=
WAIT_OBJECT_0) {
			userlog("Error on WaitForSingleObject,
in function findIdleDBExecution\n");
			return NULL;
		}
	}

	return NULL;
}



void readInit(char *output, char *parameter, char *default_value)
{
	if (_access(InitFile, 0x00) != NULL) {
		userlog("Cannot access init file: %s\n", InitFile);
		strcpy(output, default_value);
	}
	else
```

```
                GetPrivateProfileString("TPCC", parameter, default_value,
output, MAXLEN, InitFile);
}



int initializeDBExecutionPool()
{
   DBExecution *ptr;

        userlog("execute initializeDBExecutionPool()\n");

   DBExecution_pool = (DBExecution_pool_info *) malloc
(sizeof(DBExecution_pool_info)*DBConnections);
        if (DBExecution_pool == 0) {
                userlog("malloc failed in
initializeDBExecutionPool\n");
                return FALSE;
        }
        memset((void*)DBExecution_pool, 0,
sizeof(DBExecution_pool_info)*DBConnections);

        for (int i=0; i<DBConnections; i++) {
                if ((ptr=new DBExecution) == NULL) {
                        userlog("Cannot create DBExecution
object\n");
                        return FALSE;
                }

                if ((TlsSetValue(TlsPtr, (void *) ptr)) == NULL) {
                        userlog("TlsSetValue failed\n");
                        return FALSE;
                }

                if (ptr->TPCinit(i, "tpcc", "tpcc")) {
                        userlog("TPCinit failed\n");
                        return FALSE;
                }

                DBExecution_pool[i].current_status = IDLE;
                DBExecution_pool[i].pointer = (void *) ptr;

                userlog ("DBExecution %d is initialized\n", i);
        }

        return TRUE;
}



void userlog (char * str, ...)
{
        HANDLE logMutex;
        FILE *file;
        time_t t;
        struct tm *currtime;
        va_list va;
        int threadId;

        logMutex = CreateMutex(NULL, FALSE, "TPCC_LOG");
        // Wait for initialization ended
        WaitForSingleObject(logMutex, INFINITE);

        threadId = GetCurrentThreadId();
   time (&t);
        currtime = localtime(&t);

        if ((file=fopen(LogFile, "a"))==(FILE *) NULL) {

                fprintf(stderr, "Can't open file : %s\n", LogFile);
                        exit(-1);
        }

        va_start(va, str);
        fprintf(file, "[Time %d:%d:%d  Thread: %d]  ", currtime-
>tm_hour, currtime->tm_min, currtime->tm_sec, threadId);
        vfprintf(file, str, va);
        fprintf(file, "\n");
        va_end(va);

        fclose(file);


        ReleaseMutex(logMutex);
        CloseHandle(logMutex);
}


sb4 no_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
       dvoid **bufpp, ub4 *alenp, ub1 *piecep,
       dvoid **indpp)
{
  *bufpp = (dvoid*)0;
  *alenp =0;
  *indpp = (dvoid*)0;
  *piecep =OCI_ONE_PIECE;
  return (OCI_CONTINUE);
}


sb4 TPC_oid_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
       dvoid **bufpp, ub4 **alenp, ub1 *piecep,
       dvoid **indpp, ub2 **rcodepp)
{

  DBExecution *dbc;

  dbc = (DBExecution*) TlsGetValue(TlsPtr);
  if (dbc == 0) {
        userlog("TlsGetValue failed in TPC_oid_data\n");
        exit(-1);
  }

  *bufpp = &dbc->dctx->del_o_id[iter];
  *indpp= &dbc->dctx->del_o_id_ind[iter];
  dbc->dctx->del_o_id_len[iter]=sizeof(dbc->dctx->del_o_id[0]);
  *alenp= &dbc->dctx->del_o_id_len[iter];
  *rcodepp = &dbc->dctx->del_o_id_rcode[iter];
  *piecep =OCI_ONE_PIECE;
  return (OCI_CONTINUE);
}


sb4 cid_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
       dvoid **bufpp, ub4 **alenp, ub1 *piecep,
       dvoid **indpp, ub2 **rcodepp)
{

  DBExecution *dbc;

  dbc = (DBExecution*) TlsGetValue(TlsPtr);
  if (dbc == 0) {
        userlog("TlsGetValue failed in cid_data\n");
        exit(-1);
  }

  *bufpp = &dbc->dctx->c_id[iter];
  *indpp= &dbc->dctx->c_id_ind[iter];
  dbc->dctx->c_id_len[iter]=sizeof(dbc->dctx->c_id[0]);
  *alenp= &dbc->dctx->c_id_len[iter];
  *rcodepp = &dbc->dctx->c_id_rcode[iter];
  *piecep =OCI_ONE_PIECE;
  return (OCI_CONTINUE);
}


sb4 amt_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
              dvoid **bufpp, ub4 **alenp, ub1 *piecep,
              dvoid **indpp, ub2 **rcodepp)
{
  amtctx *actx;
  actx =(amtctx*)ctxp;

  *bufpp = &actx->ol_amt[index];
  *indpp= &actx->ol_amt_ind[index];
  actx->ol_amt_len[index]=sizeof(actx->ol_amt[0]);
  *alenp= &actx->ol_amt_len[index];
  *rcodepp = &actx->ol_amt_rcode[index];
  *piecep =OCI_ONE_PIECE;
  return (OCI_CONTINUE);
}



/*************************************************************
**********************************
*  DBExecution member functions                              *
*************************************************************
**********************************/

DBExecution::DBExecution()
{
  tracelevel = 0;
  logon = 0;
  new_init = 0;
  pay_init = 0;
  ord_init = 0;
  del_init_oci = 0;
  del_init_plsql = 0;
  sto_init = 0;
}

DBExecution::~DBExecution()
{

}



#define SQLTXTNEW2 "BEGIN inittpcc.init_no(:idx1arr); END;"
#define SQLTXTDEL "BEGIN inittpcc.init_del ; END;"
#define SQLTXTDEL1 "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 SQLTXTDEL3 "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 SQLTXTDEL4 "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 SQLTXTDEL6 "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 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 "

#ifdef PLSQLSTO
#define SQLTXTSTO "BEGIN stocklevel.getstocklevel (:w_id, :d_id,
:threshold, \
  :low_stock); END;"
#else
#define SQLTXTSTO "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


#define SQLTXT_INIT "BEGIN inittpcc.init_pay; END;"



int DBExecution::sqlfile(char *fnam, text *linebuf)
{
  FILE *fd;
  int nulpt = 0;
  char realfile[512];

  sprintf(realfile,"%s",fnam);
  fd = fopen(realfile,"r");
  if (!fd){
     fprintf(stderr," fopen on %s failed %d\n",fnam,fd);
```

```
      exit(-1);
  }
  while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
    nulpt = strlen((char *)linebuf);

  fclose(fd);

  return(nulpt);
}



int DBExecution::ocierror(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:
    userlog("ocierror: Module %s Line %d\n", fname, lineno);
    userlog("ocierror: Error - OCI_SUCCESS_WITH_INFO\n");
    lstat = OCIErrorGet (errhp, recno++, (text *) NULL, &errcode, errbuf,
                    (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    userlog("ocierror: Error - %s\n", errbuf);
    break;
  case OCI_NEED_DATA:
    userlog("ocierror: Module %s Line %d\n", fname, lineno);
    userlog("ocierror: Error - OCI_NEED_DATA\n");
    return (IRRECERR);
  case OCI_NO_DATA:
    userlog("ocierror: Module %s Line %d\n", fname, lineno);
    userlog("ocierror: 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)
    {
      userlog("ocierror: Module %s Line %d\n", fname, lineno);
      userlog("ocierror: Error - %s\n", errbuf);
      lstat = OCIErrorGet (errhp, recno++, (text *) NULL, &errcode, errbuf,
                    (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    }
    return (errcode);
/* vmm313    TPCexit(1); */
/* vmm313    exit(1); */
  case OCI_INVALID_HANDLE:
    userlog("ocierror: Module %s Line %d\n", fname, lineno);
    userlog("ocierror: Error - OCI_INVALID_HANDLE\n");
    TPCexit();
    exit(-1);
  case OCI_STILL_EXECUTING:
    userlog("ocierror: Module %s Line %d\n", fname, lineno);
    userlog("ocierror: Error - OCI_STILL_EXECUTE\n");
    return (IRRECERR);
  case OCI_CONTINUE:
    userlog("ocierror: Module %s Line %d\n", fname, lineno);
    userlog("ocierror: Error - OCI_CONTINUE\n");
    return (IRRECERR);
  default:
    userlog("ocierror: Module %s Line %d\n", fname, lineno);
    userlog("ocierror: Status - %s\n", status);
    return (IRRECERR);
  }
  return (RECOVERR);
}


/************************************************************
*********************************
* TPCinit   TPCexit                                          *
************************************************************
*********************************/


int DBExecution::TPCinit (int id, char *uid, char *pwd)
{
//   OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0,0);
//   OCIEnvCreate(&tpcenv, OCI_DEFAULT|OCI_OBJECT,(dvoid
*)0,0,0,0,0, (dvoid **)0);

#ifndef LOOPBACK

   text stmbuf[100];
   int i;

#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 SQLTXTOPS "alter session set current_schema = tpcc"

   proc_no = id;
/*
   char *temp;

   if ((temp = getenv("LOCAL"))==NULL)
                        _putenv( "LOCAL=tpcc" );

   OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0,0); */
// OCIERROR(errhp,
OCIInitialize(OCI_THREADED|OCI_OBJECT,(dvoid *)0,0,0,0));
// OCIERROR(errhp, OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid
**)0));

   OCIEnvCreate(&tpcenv, OCI_DEFAULT|OCI_OBJECT,(dvoid
*)0,0,0,0,0, (dvoid **)0);

   OCIERROR(errhp, OCIHandleAlloc((dvoid *)tpcenv, (dvoid
**)&tpcsrv, OCI_HTYPE_SERVER, 0 , (dvoid **)0));
   OCIERROR(errhp, OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp,
OCI_HTYPE_ERROR, 0 , (dvoid **)0));
   OCIERROR(errhp, OCIHandleAlloc((dvoid *)tpcenv, (dvoid
**)&tpcsvc, OCI_HTYPE_SVCCTX, 0 , (dvoid **)0));

   for (i=0; i<100; i++) {

                        execstatus = OCIServerAttach(tpcsrv, errhp, (text
*)0,0,OCI_DEFAULT);
                        if (execstatus == OCI_SUCCESS || execstatus ==
OCI_SUCCESS_WITH_INFO)
                                  break;
                        OCIERROR(errhp, execstatus);
                        Sleep(10);
   }

   if (i==100) {
                userlog("Can't attach to Server after 100 tries\n");
                return -1;
   }
   OCIERROR(errhp, OCIAttrSet((dvoid *)tpcsvc,
OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv, (ub4)0,OCI_ATTR_SERVER,
errhp));
   OCIERROR(errhp, OCIHandleAlloc((dvoid *)tpcenv, (dvoid
**)&tpcusr, OCI_HTYPE_SESSION, 0 , (dvoid **)0));
#ifdef OPS_LOGIN
   OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr,
OCI_CRED_EXT, OCI_DEFAULT));
#else
   OCIERROR(errhp, OCIAttrSet((dvoid *)tpcusr,
OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp));
   OCIERROR(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));
#endif

   OCIERROR(errhp, OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr,
0, OCI_ATTR_SESSION, errhp));

   /* run all transaction in serializable mode */

   OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT);
   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);

#ifdef OPS_LOGIN
     OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid**)0);
     memset(stmbuf,0,100);
     sprintf ((char *) stmbuf, SQLTXTOPS);
     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((dvoid *)curi, OCI_HTYPE_STMT);
#endif


   if (tracelevel == 3) {
     OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid**)0);
     memset(stmbuf,0,100);
     sprintf ((char *) stmbuf, SQLTXTTIM);
     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((dvoid *)curi, OCI_HTYPE_STMT);
   }

   logon = 1;

   OCIERROR(errhp,OCIDateSysDate(errhp,&cr_date));

   if (tkvcninit ()) {  /* new order */
      TPCexit ();
      return (-1);
   }
   else
      new_init = 1;

   if (tkvcpinit ()) {   /* payment */
      TPCexit ();
      return (-1);
   }
   else
      pay_init = 1;


   if (tkvcoinit ()) {   /* order status */
      TPCexit ();
      return (-1);
   }
   else
      ord_init = 1;

   if (tkvcdinit (0)) {   /* delivery */
      TPCexit ();
      return (-1);
   }
   else
      del_init_oci = 1;

   if (tkvcdinit (1)) {   /* delivery */
      TPCexit ();
      return (-1);
   }
   else
      del_init_plsql = 1;

   if (tkvcsinit ()) {   /* stock level */
      TPCexit ();
      return (-1);
   }
   else
      sto_init = 1;

#endif

   return (0);
}



void DBExecution::TPCexit()
{

#ifndef LOOPBACK

   if (new_init) {
      tkvcndone();
      new_init = 0;
   }
   if (pay_init) {
      tkvcpdone();
      pay_init = 0;
   }
   if (ord_init) {
      tkvcodone();
      ord_init = 0;
   }
   if (del_init_oci) {
      tkvcddone(0);
      del_init_oci = 0;
   }
   if (del_init_plsql) {
      tkvcddone(1);
      del_init_plsql = 0;
   }
   if (sto_init) {
      tkvcsdone();
      sto_init = 0;
    }


    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);

#endif

}


/*************************************************************
**********************************
* tkvcninit  tkvcndone  tkvcpinit  tkvcpdone  tkvcdinit  tkvcddone
tkvcoinit  tkvcodone      *
* tkvcsinit  tkvcsdone                                              *
*************************************************************
**********************************/


int DBExecution::tkvcninit ()
{

  text stmbuf[32*1024];

  nctx = (newctx *) malloc (sizeof(newctx));
  DISCARD memset(nctx,(char)0,sizeof(newctx));
  nctx->w_id_len = sizeof(w_id);
  nctx->d_id_len = sizeof(d_id);
  nctx->c_id_len = sizeof(c_id);
  nctx->o_all_local_len = sizeof(o_all_local);
  nctx->o_ol_cnt_len = sizeof(o_ol_cnt);
  nctx->w_tax_len = 0;
  nctx->d_tax_len = 0;
  nctx->o_id_len = sizeof(o_id);
  nctx->c_discount_len = 0;
  nctx->c_credit_len = 0;
  nctx->c_last_len = 0;
  nctx->retries_len = sizeof(retries);
  nctx->cr_date_len = sizeof(cr_date);

  /* open first cursor */
  DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&nctx-
>curn1),
          OCI_HTYPE_STMT, 0, (dvoid**)0));

#if defined(ISO)
  sqlfile(".\\blocks\\tkvcpnew_iso.sql",stmbuf);
#else
#if defined(ISO7)
  sqlfile(".\\blocks\\tkvcpnew_iso7.sql",stmbuf);
#else
  sqlfile(".\\blocks\\tkvcpnew.sql",stmbuf);
#endif
#endif

  DISCARD OCIERROR(errhp,OCIStmtPrepare(nctx->curn1, errhp,
stmbuf,
          strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

  /* bind variables */

  OCIBNDPL(nctx->curn1, nctx->w_id_bp, errhp,
":w_id",ADR(w_id),SIZ(w_id),
        SQLT_INT, &nctx->w_id_len);
  OCIBNDPL(nctx->curn1, nctx->d_id_bp, errhp,
":d_id",ADR(d_id),SIZ(d_id),
        SQLT_INT, &nctx->d_id_len);
  OCIBNDPL(nctx->curn1, nctx->c_id_bp, errhp,
":c_id",ADR(c_id),SIZ(c_id),
        SQLT_INT, &nctx->c_id_len);
  OCIBNDPL(nctx->curn1, nctx->o_all_local_bp, errhp, ":o_all_local",
        ADR(o_all_local), SIZ(o_all_local),SQLT_INT, &nctx-
>o_all_local_len);
  OCIBNDPL(nctx->curn1, nctx->o_all_cnt_bp, errhp,
":o_ol_cnt",ADR(o_ol_cnt),
        SIZ(o_ol_cnt),SQLT_INT, &nctx->o_ol_cnt_len);
  OCIBNDPL(nctx->curn1, nctx->w_tax_bp, errhp,
":w_tax",ADR(w_tax),SIZ(w_tax),
        SQLT_FLT, &nctx->w_tax_len);
  OCIBNDPL(nctx->curn1, nctx->d_tax_bp, errhp,
":d_tax",ADR(d_tax),SIZ(d_tax),
        SQLT_FLT, &nctx->d_tax_len);
  OCIBNDPL(nctx->curn1, nctx->o_id_bp, errhp,
":o_id",ADR(o_id),SIZ(o_id),
        SQLT_INT, &nctx->o_id_len);
  OCIBNDPL(nctx->curn1, nctx->c_discount_bp, errhp, ":c_discount",
        ADR(c_discount), SIZ(c_discount),SQLT_FLT, &nctx-
>c_discount_len);
  OCIBNDPL(nctx->curn1, nctx->c_credit_bp, errhp, ":c_credit",c_credit,
        SIZ(c_credit),SQLT_CHR, &nctx->c_credit_len);
  OCIBNDPL(nctx->curn1, nctx->c_last_bp, errhp,
":c_last",c_last,SIZ(c_last),
        SQLT_STR, &nctx->c_last_len);
  OCIBNDPL(nctx->curn1, nctx->retries_bp, errhp, ":retry",ADR(retries),
        SIZ(retries),SQLT_INT, &nctx->retries_len);
  OCIBNDPL(nctx->curn1, nctx->cr_date_bp, errhp, ":cr_date",&cr_date,
        SIZ(OCIDate), SQLT_ODT, &nctx->cr_date_len);

  OCIBNDPLA(nctx->curn1, nctx->ol_i_id_bp,errhp,":ol_i_id",nol_i_id,
         SIZ(int), SQLT_INT, nctx->nol_i_id_len,NITEMS,&nctx-
>nol_i_count);
  OCIBNDPLA(nctx->curn1, nctx->ol_supply_w_id_bp, errhp,
":ol_supply_w_id",
         nol_supply_w_id,SIZ(int),SQLT_INT, nctx->nol_supply_w_id_len,
         NITEMS, &nctx->nol_s_count);

#ifdef USE_IEEE_NUMBER
  OCIBNDPLA(nctx->curn1, nctx->ol_quantity_bp,errhp,":ol_quantity",
         nol_quantity, SIZ(double),SQLT_Bdouble,nctx->nol_quantity_len,
         NITEMS,&nctx->nol_q_count);

  OCIBNDPLA(nctx->curn1, nctx-
>i_price_bp,errhp,":i_price",i_price,SIZ(double),
         SQLT_Bdouble, nctx->i_price_len, NITEMS, &nctx-
>nol_item_count);
#else
  OCIBNDPLA(nctx->curn1, nctx->ol_quantity_bp,errhp,":ol_quantity",
         nol_quantity, SIZ(int),SQLT_INT,nctx->nol_quantity_len,
         NITEMS,&nctx->nol_q_count);

  OCIBNDPLA(nctx->curn1, nctx-
>i_price_bp,errhp,":i_price",i_price,SIZ(int),
         SQLT_INT, nctx->i_price_len, NITEMS, &nctx->nol_item_count);
#endif /* USE_IEEE_NUMBER */
  OCIBNDPLA(nctx->curn1, nctx->i_name_bp,errhp,":i_name",i_name,
         SIZ(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,errhp,":s_quantity",s_quantity,
         SIZ(double), SQLT_Bdouble,nctx->s_quant_len,NITEMS,&nctx-
>nol_qty_count);
#else
```

```
  OCIBNDPLA(nctx->curn1, nctx-
>s_quantity_bp,errhp,":s_quantity",s_quantity,
         SIZ(int), SQLT_INT,nctx->s_quant_len,NITEMS,&nctx-
>nol_qty_count);
#endif /* USE_IEEE_NUMBER */

  OCIBNDPLA(nctx->curn1, nctx-
>s_bg_bp,errhp,":brand_generic",brand_generic,
         SIZ(char), SQLT_CHR,nctx->s_bg_len,NITEMS,&nctx-
>nol_bg_count);
#ifdef USE_IEEE_NUMBER
  OCIBNDPLA(nctx->curn1, nctx-
>ol_amount_bp,errhp,":ol_amount",nol_amount,
         SIZ(double),SQLT_Bdouble, nctx-
>nol_amount_len,NITEMS,&nctx->nol_am_count);

  OCIBNDPLA(nctx->curn1, nctx->s_remote_bp,errhp,":s_remote",nctx-
>s_remote,
         SIZ(double),SQLT_Bdouble, nctx->s_remote_len,NITEMS,&nctx-
>s_remote_count);
#else
  OCIBNDPLA(nctx->curn1, nctx-
>ol_amount_bp,errhp,":ol_amount",nol_amount,
         SIZ(int),SQLT_INT, nctx->nol_amount_len,NITEMS,&nctx-
>nol_am_count);

  OCIBNDPLA(nctx->curn1, nctx->s_remote_bp,errhp,":s_remote",nctx-
>s_remote,
         SIZ(int),SQLT_INT, nctx->s_remote_len,NITEMS,&nctx-
>s_remote_count);
#endif /* USE_IEEE_NUMBER */

  /* open second cursor */
  DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid
**)(&nctx->curn2),
              OCI_HTYPE_STMT, 0, (dvoid**)0));
  DISCARD sprintf ((char *) stmbuf, SQLTXTNEW2);
  DISCARD OCIERROR(errhp,OCIStmtPrepare(nctx->curn2, 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;
             o_ol_cnt = NITEMS;


    /* Bind array */
    OCIBNDPLA(nctx->curn2, idx1arr_bp,errhp,":idx1arr",idx1arr,
          SIZ(int), SQLT_INT, idx1arr_len, NITEMS,&idx1arr_count);

     execstatus = OCIStmtExecute(tpcsvc,nctx->curn2,errhp,1,0,
             NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);

              if(execstatus != OCI_SUCCESS) {
                        OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
       errcode = OCIERROR(errhp,execstatus);
       return -1;
     }
   }

  return (0);
}



void DBExecution::tkvcndone ()
{
   if (nctx)
   {
     DISCARD OCIHandleFree((dvoid *)nctx-
>curn1,OCI_HTYPE_STMT);
     DISCARD OCIHandleFree((dvoid *)nctx-
>curn2,OCI_HTYPE_STMT);
     free (nctx);
   }
}



int DBExecution::tkvcdinit (int plsqlflag)

{

   text stmbuf[SQL_BUF_SIZE];

   if (plsqlflag)
   {
     pldctx = (pldelctx *) malloc (sizeof(pldelctx));
     DISCARD memset(pldctx,(char)0,(ub4)sizeof(pldelctx));
     /* Initialize */
     DISCARD OCIHandleAlloc(tpcenv, (dvoid**)&pldctx->curp1,
OCI_HTYPE_STMT, 0,
                       (dvoid**)0);
     DISCARD sprintf ((char *) stmbuf, SQLTXTDEL);
     DISCARD OCIStmtPrepare(pldctx->curp1, errhp, stmbuf,
                       (ub4) strlen((char *)stmbuf),
                OCI_NTV_SYNTAX, OCI_DEFAULT);
     DISCARD OCIERROR(errhp,
       OCIStmtExecute(tpcsvc,pldctx-
>curp1,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)
     sqlfile(".\\blocks\\tkvcpdel_iso5.sql",stmbuf);
  #endif
  #if defined(ISO6)
     sqlfile(".\\blocks\\tkvcpdel_iso6.sql",stmbuf);
  #endif
  #if defined(ISO8)
     sqlfile(".\\blocks\\tkvcpdel_iso8.sql",stmbuf);
  #endif
#else
     sqlfile(".\\blocks\\tkvcpdel.sql",stmbuf);
#endif
     DISCARD OCIStmtPrepare(pldctx->curp2, errhp, stmbuf,
                 (ub4)strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT);
     OCIBNDPL(pldctx->curp2, pldctx->w_id_bp , errhp,":w_id",
            ADR(w_id), SIZ(int), SQLT_INT,&pldctx->w_id_len);
     OCIBNDPL(pldctx->curp2, pldctx->ordcnt_bp , errhp,":ordcnt",
           ADR(pldctx->ordcnt), SIZ(int), SQLT_INT,&pldctx-
>ordcnt_len);
    OCIBNDPL(pldctx->curp2, pldctx->del_date_bp,errhp,":now",
           ADR(pldctx->del_date), SIZ(OCIDate), SQLT_ODT,&pldctx-
>del_date_len);
    OCIBNDPL(pldctx->curp2, pldctx->carrier_id_bp , errhp,
           ":carrier_id", ADR(o_carrier_id), SIZ(int),
           SQLT_INT, &pldctx->carrier_id_len);

    OCIBNDPLA(pldctx->curp2, pldctx->d_id_bp, 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, 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, errhp,":sums",
            pldctx->sums,SIZ(double),SQLT_Bdouble, pldctx-
>sums_len,NDISTS,
            &pldctx->sums_rcnt);
#else
    OCIBNDPLA(pldctx->curp2, pldctx->sums_bp, errhp,":sums",
            pldctx->sums,SIZ(int),SQLT_INT, pldctx->sums_len,NDISTS,
            &pldctx->sums_rcnt);
#endif

    OCIBNDPLA(pldctx->curp2, pldctx->o_c_id_bp, 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 , errhp,":retry",
          ADR(pldctx->retry), SIZ(int),SQLT_INT);

  }
  else
  {

    dctx = (delctx *) malloc (sizeof(delctx));
    memset(dctx,(char)0,sizeof(delctx));
    dctx->norow = 0;
    actx = (amtctx *) malloc (sizeof(amtctx));
    memset(actx,(char)0,sizeof(amtctx));

    OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd1),
OCI_HTYPE_STMT, 0,
              (dvoid**)0);
    DISCARD sprintf ((char *) stmbuf, "%s",    SQLTXTDEL1);
    DISCARD OCIStmtPrepare(dctx->curd1, errhp, stmbuf,
                  strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT);

    OCIBND(dctx->curd1, dctx->w_id_bp,errhp,":w_id",dctx-
>w_id,SIZ(int),
        SQLT_INT);
    OCIBNDRA(dctx->curd1, dctx->d_id_bp,errhp,":d_id",dctx-
>d_id,SIZ(int),
        SQLT_INT,NULL,NULL,NULL);

    OCIBNDRAD(dctx->curd1, dctx->del_o_id_bp, errhp, ":o_id",
           SIZ(int),SQLT_INT,NULL,
           &dctx->oid_ctx,no_data,TPC_oid_data);

  /* open third cursor */

    DISCARD OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd3),
OCI_HTYPE_STMT,
                      0, (dvoid**)0);
    DISCARD sprintf ((char *) stmbuf, SQLTXTDEL3);
    DISCARD OCIStmtPrepare(dctx->curd3, errhp, stmbuf, strlen((char
*)stmbuf),
                       OCI_NTV_SYNTAX, OCI_DEFAULT);


  /* bind variables */

    OCIBNDRA(dctx->curd3, dctx->carrier_id_bp,errhp,":carrier_id",
          dctx->carrier_id, SIZ(dctx->carrier_id[0]),SQLT_INT,
         dctx->carrier_id_ind, dctx->carrier_id_len,dctx->carrier_id_rcode);

    OCIBNDRA(dctx->curd3, dctx->w_id_bp3, errhp, ":w_id", dctx-
>w_id,SIZ(int),
          SQLT_INT, NULL, NULL, NULL);
    OCIBNDRA(dctx->curd3, dctx->d_id_bp3, errhp, ":d_id", dctx-
>d_id,SIZ(int),
          SQLT_INT,NULL, NULL, NULL);
    OCIBNDRA(dctx->curd3, dctx->del_o_id_bp3, errhp, ":o_id", dctx-
>del_o_id,
          SIZ(int), SQLT_INT,NULL,NULL,NULL);
    OCIBNDRAD(dctx->curd3, dctx->c_id_bp3, errhp, ":o_c_id", SIZ(int),
          SQLT_INT,NULL,&dctx->cid_ctx,no_data, cid_data);

  /* open fourth cursor */

    DISCARD OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd4),
OCI_HTYPE_STMT, 0,
                (dvoid**)0);
    DISCARD sprintf ((char *) stmbuf, SQLTXTDEL4);
    DISCARD OCIStmtPrepare(dctx->curd4, errhp, stmbuf, strlen((char
*)stmbuf),
                       OCI_NTV_SYNTAX, OCI_DEFAULT);

  /* bind variables */

    OCIBND(dctx->curd4, dctx->w_id_bp4,errhp,":w_id",dctx->w_id,
        SIZ(int), SQLT_INT);
    OCIBND(dctx->curd4, dctx->d_id_bp4,errhp,":d_id",dctx->d_id,
        SIZ(int), SQLT_INT);
    OCIBND(dctx->curd4, dctx->o_id_bp,errhp,":o_id",dctx->del_o_id,
        SIZ(int),SQLT_INT);
    OCIBND(dctx->curd4, dctx->cr_date_bp,errhp,":cr_date", dctx-
>del_date,
        SIZ(OCIDate), SQLT_ODT);
    OCIBNDRAD(dctx->curd4, dctx->olamt_bp, errhp, ":ol_amount",
         SIZ(int), SQLT_INT,NULL, actx,no_data,amt_data);



  /* open sixth cursor */

    DISCARD OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd6),
OCI_HTYPE_STMT,
                    0, (dvoid**)0);
    DISCARD sprintf ((char *) stmbuf, SQLTXTDEL6);
    DISCARD OCIStmtPrepare(dctx->curd6, errhp, stmbuf, strlen((char
*)stmbuf),
                       OCI_NTV_SYNTAX, OCI_DEFAULT);

  /* bind variables */


    OCIBND(dctx->curd6,dctx->amt_bp,errhp,":amt",dctx->amt,SIZ(int),
       SQLT_INT);
    OCIBND(dctx->curd6,dctx->w_id_bp6,errhp,":w_id",dctx-
>w_id,SIZ(int),
        SQLT_INT);
    OCIBND(dctx->curd6,dctx->d_id_bp6,errhp,":d_id",dctx-
>d_id,SIZ(int),
        SQLT_INT);
    OCIBND(dctx->curd6,dctx->c_id_bp,errhp,":c_id",dctx->c_id,SIZ(int),
          SQLT_INT);
  }
  return (0);

}



void DBExecution::shiftdata(int from)
{
  int i;
  for (i=from;i<NDISTS-1; i++)
  {
    dctx->del_o_id_ind[i] = dctx->del_o_id_ind[i+1];
    dctx->del_o_id[i] = dctx->del_o_id[i+1];
    dctx->w_id[i] = dctx->w_id[i+1];
    dctx->d_id[i] = dctx->d_id[i+1];
    dctx->carrier_id[i] = dctx->carrier_id[i+1];
  }
}



void DBExecution::tkvcddone(int plsqlflag)
{
  if (plsqlflag)
  {
    if (pldctx)
    {
      DISCARD OCIHandleFree((dvoid *)dctx-
>curd0,OCI_HTYPE_STMT);
      DISCARD free(pldctx);
    }
  }
  else
  {
    if (dctx)
    {
      OCIHandleFree((dvoid *)dctx->curd1,OCI_HTYPE_STMT);
      OCIHandleFree((dvoid *)dctx->curd2,OCI_HTYPE_STMT);
      OCIHandleFree((dvoid *)dctx->curd3,OCI_HTYPE_STMT);
      OCIHandleFree((dvoid *)dctx->curd4,OCI_HTYPE_STMT);
      OCIHandleFree((dvoid *)dctx->curd5,OCI_HTYPE_STMT);
      OCIHandleFree((dvoid *)dctx->curd6,OCI_HTYPE_STMT);
      DISCARD free (dctx);
    }
  }
}



int DBExecution::tkvcoinit ()
{
  int i;
  text stmbuf[SQL_BUF_SIZE];

  octx = (ordctx *) malloc (sizeof(ordctx));
  DISCARD memset(octx,(char)0,sizeof(ordctx));
  octx->cs = 1;
  octx->norow = 0;
  octx->somerows = 10;
  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-
>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,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,errhp));
/* get order/customer info back based on rowid */
  DISCARD sprintf((char *) stmbuf, SQLCUR1);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo1,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,errhp));

/*  c_id == 0, use lastname to find customer */
  DISCARD sprintf((char *) stmbuf, SQLCUR2);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo2,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,errhp));

  DISCARD sprintf((char *) stmbuf, SQLCUR3);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo3,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,errhp));

  DISCARD sprintf((char *) stmbuf, SQLCUR4);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo4,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,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(ol_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],errhp,":w_id",ADR(w_id),
        SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->d_id_bp[0],errhp,":d_id",ADR(d_id),
        SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->c_last_bp[0],errhp,":c_last",c_last,
        SIZ(c_last), SQLT_STR);
  OCIDFNRA(octx->curo0,octx->c_rowid_dp,errhp,1,octx->c_rowid_ptr,
          SIZ(OCIRowid*), SQLT_RDD, NULL, octx->c_rowid_len,
NULL);

  OCIBND(octx->curo1,octx->c_rowid_bp,errhp,":cust_rowid", &octx-
>c_rowid_cust,
        sizeof( octx->c_rowid_ptr[0]),SQLT_RDD);
  OCIDEF(octx->curo1,octx-
>c_id_dp,errhp,1,ADR(c_id),SIZ(int),SQLT_INT);
#ifdef USE_IEEE_NUMBER
  OCIDEF(octx->curo1,octx->c_balance_dp[0],errhp,2,ADR(c_balance),
         SIZ(double),SQLT_BDOUBLE);
#else
  OCIDEF(octx->curo1,octx->c_balance_dp[0],errhp,2,ADR(c_balance),
         SIZ(double),SQLT_FLT);
#endif /* USE_IEEE_NUMBER */
  OCIDEF(octx->curo1,octx->c_first_dp[0],errhp,3,c_first,SIZ(c_first)-1,
         SQLT_CHR);
  OCIDEF(octx->curo1,octx->c_middle_dp[0],errhp,4,c_middle,
         SIZ(c_middle)-1,SQLT_AFC);
  OCIDEF(octx->curo1,octx->c_last_dp[0],errhp,5,c_last,SIZ(c_last)-1,
         SQLT_CHR);
  OCIDEF(octx->curo1,octx-
>o_id_dp[0],errhp,6,ADR(o_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_entry_d_dp[0],errhp,7,
         &o_entry_d_base,SIZ(OCIDate),SQLT_ODT);
  OCIDEF(octx->curo1,octx->o_cr_id_dp[0],errhp,8,ADR(o_carrier_id),
         SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_ol_cnt_dp[0],errhp,9,ADR(o_ol_cnt),
         SIZ(int),SQLT_INT);

/* Bind for third cursor , no-zero customer id */
  OCIBND(octx->curo2,octx->w_id_bp[1],errhp,":w_id",ADR(w_id),
        SIZ(int),SQLT_INT);
  OCIBND(octx->curo2,octx->d_id_bp[1],errhp,":d_id",ADR(d_id),
               SIZ(int),SQLT_INT);
  OCIBND(octx->curo2,octx->c_id_bp,errhp,":c_id",ADR(c_id),
               SIZ(int),SQLT_INT);
#ifdef USE_IEEE_NUMBER
  OCIDEF(octx->curo2,octx->c_balance_dp[1],errhp,1,ADR(c_balance),
         SIZ(double),SQLT_BDOUBLE);
#else
  OCIDEF(octx->curo2,octx->c_balance_dp[1],errhp,1,ADR(c_balance),
         SIZ(double),SQLT_FLT);
#endif /* USE_IEEE_NUMBER */
  OCIDEF(octx->curo2,octx->c_first_dp[1],errhp,2,c_first,SIZ(c_first)-1,
         SQLT_CHR);
  OCIDEF(octx->curo2,octx->c_middle_dp[1],errhp,3,c_middle,
         SIZ(c_middle)-1,SQLT_AFC);
  OCIDEF(octx->curo2,octx->c_last_dp[1],errhp,4,c_last,SIZ(c_last)-1,
        SQLT_CHR);
  OCIDEF(octx->curo2,octx-
>o_id_dp[1],errhp,5,ADR(o_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo2,octx->o_entry_d_dp[1],errhp,6, &o_entry_d_base,
        SIZ(OCIDate),SQLT_ODT);
  OCIDEF(octx->curo2, octx->o_cr_id_dp[1],errhp,7,ADR(o_carrier_id),
         SIZ(int), SQLT_INT);
  OCIDEF(octx->curo2,octx->o_ol_cnt_dp[1],errhp,8,ADR(o_ol_cnt),
         SIZ(int),SQLT_INT);

/* Bind for last cursor */

  OCIBND(octx->curo3,octx->w_id_bp[2],errhp,":w_id",ADR(w_id),
SIZ(int),SQLT_INT);
  OCIBND(octx->curo3,octx->d_id_bp[2],errhp,":d_id",ADR(d_id),
SIZ(int),SQLT_INT);
  OCIBND(octx->curo3,octx->o_id_bp,errhp,":o_id",ADR(o_id),
SIZ(int),SQLT_INT);
/*
  OCIBND(octx->curo3,octx->c_id_bp,errhp,":c_id",ADR(c_id),
SIZ(int),SQLT_INT);
 */

  OCIDFNRA(octx->curo3, octx->ol_i_id_dp, errhp, 1,
ol_i_id,SIZ(int),SQLT_INT,
                NULL,octx->ol_i_id_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_supply_w_id_dp,errhp,2,
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,errhp,3,
ol_quantity,SIZ(double),
          SQLT_Bdouble, NULL,octx->ol_quantity_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_amount_dp,errhp,4,ol_amount,
SIZ(double),
          SQLT_Bdouble,NULL, octx->ol_amount_len, NULL);
#else
  OCIDFNRA(octx->curo3, octx->ol_quantity_dp,errhp,3,
ol_quantity,SIZ(int),
          SQLT_INT, NULL,octx->ol_quantity_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_amount_dp,errhp,4,ol_amount,
SIZ(int),
          SQLT_INT,NULL, octx->ol_amount_len, NULL);
#endif /* USE_IEEE_NUMBER */
  OCIDFNRA(octx->curo3,octx-
>ol_d_base_dp,errhp,5,ol_d_base,SIZ(OCIDate),
          SQLT_ODT, NULL,octx->ol_delivery_d_len,NULL);

  OCIBND(octx->curo4,octx->w_id_bp[3],errhp,":w_id",ADR(w_id),
        SIZ(int),SQLT_INT);
  OCIBND(octx->curo4,octx->d_id_bp[3],errhp,":d_id",ADR(d_id),
        SIZ(int),SQLT_INT);
  OCIBND(octx->curo4,octx->c_last_bp[1],errhp,":c_last",c_last,
        SIZ(c_last), SQLT_STR);
  OCIDEF(octx->curo4,octx->c_count_dp,errhp,1,ADR(octx-
>rcount),SIZ(int),
        SQLT_INT);

  return (0);
}



void DBExecution::tkvcodone ()
{

  if (octx)
    free (octx);

}
```

```
int DBExecution::tkvcpinit (void)
{
   text stmbuf[SQL_BUF_SIZE];
   pctx = (payctx *)malloc(sizeof(payctx));
   memset(pctx,(char)0,sizeof(payctx));

/* 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, errhp,
stmbuf,
           strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
   DISCARD OCIERROR(errhp, OCIStmtExecute(tpcsvc,pctx-
>curpi,errhp,1,0,
                  NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT));

   /* customer id !=  0, go by last name */

   sqlfile(".\\blocks\\paynz.sql",stmbuf);
   DISCARD OCIERROR(errhp,OCIStmtPrepare(pctx->curp0, errhp,
stmbuf,
           strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

   /* customer id ==  0, go by last name */

   sqlfile(".\\blocks\\payz.sql",stmbuf); /* sqlfile opens
$O/bench/.../blocks/... */
   DISCARD OCIERROR(errhp,OCIStmtPrepare(pctx->curp1, errhp,
stmbuf,
           strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

  pctx->w_id_len = SIZ(w_id);
  pctx->d_id_len = SIZ(d_id);
  pctx->c_w_id_len = SIZ(c_w_id);
  pctx->c_d_id_len = SIZ(c_d_id);
  pctx->c_id_len = 0;
  pctx->h_amount_len = SIZ(h_amount);
  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 = 0;
  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(retries) ;
  pctx->cr_date_len = 7;



   /* bind variables */



   OCIBNDPL(pctx->curp0, pctx->w_id_bp[0],
errhp,":w_id",ADR(w_id),SIZ(int),
         SQLT_INT, NULL);
   OCIBNDPL(pctx->curp0, pctx->d_id_bp[0],
errhp,":d_id",ADR(d_id),SIZ(int),
         SQLT_INT, NULL);
   OCIBND(pctx->curp0, pctx->c_w_id_bp[0],
errhp,":c_w_id",ADR(c_w_id),SIZ(int),
         SQLT_INT);
   OCIBND(pctx->curp0, pctx->c_d_id_bp[0],
errhp,":c_d_id",ADR(c_d_id),SIZ(int),
         SQLT_INT);
   OCIBND(pctx->curp0, pctx->c_id_bp[0],
errhp,":c_id",ADR(c_id),SIZ(int),
         SQLT_INT);
#ifdef USE_IEEE_NUMBER
   OCIBNDPL(pctx->curp0, pctx->h_amount_bp[0],
errhp,":h_amount",ADR(h_amount),
         SIZ(double),SQLT_Bdouble,  &pctx->h_amount_len);
#else
   OCIBNDPL(pctx->curp0, pctx->h_amount_bp[0],
errhp,":h_amount",ADR(h_amount),
         SIZ(int),SQLT_INT,  &pctx->h_amount_len);
#endif /* USE_IEEE_NUMBER */
   OCIBNDPL(pctx->curp0, pctx->c_last_bp[0],
errhp,":c_last",c_last,SIZ(c_last),
         SQLT_STR, &pctx->c_last_len);
   OCIBNDPL(pctx->curp0, pctx->w_street_1_bp[0],
errhp,":w_street_1",w_street_1,
         SIZ(w_street_1),SQLT_STR, &pctx->w_street_1_len);
   OCIBNDPL(pctx->curp0, pctx->w_street_2_bp[0],
errhp,":w_street_2",w_street_2,
         SIZ(w_street_2),SQLT_STR, &pctx->w_street_2_len);
   OCIBNDPL(pctx->curp0, pctx->w_city_bp[0],
errhp,":w_city",w_city,SIZ(w_city),
         SQLT_STR, &pctx->w_city_len);
   OCIBNDPL(pctx->curp0, pctx->w_state_bp[0],
errhp,":w_state",w_state,
         SIZ(w_state), SQLT_STR, &pctx->w_state_len);
   OCIBNDPL(pctx->curp0, pctx->w_zip_bp[0],
errhp,":w_zip",w_zip,SIZ(w_zip),
         SQLT_STR, &pctx->w_zip_len);
   OCIBNDPL(pctx->curp0, pctx->d_street_1_bp[0],
errhp,":d_street_1",d_street_1,
         SIZ(d_street_1),SQLT_STR, &pctx->d_street_1_len);
   OCIBNDPL(pctx->curp0, pctx->d_street_2_bp[0],
errhp,":d_street_2",d_street_2,
         SIZ(d_street_2),SQLT_STR, &pctx->d_street_2_len);
   OCIBNDPL(pctx->curp0, pctx->d_city_bp[0],
errhp,":d_city",d_city,SIZ(d_city),
         SQLT_STR, &pctx->d_city_len);
   OCIBNDPL(pctx->curp0, pctx->d_state_bp[0], errhp,":d_state",d_state,
         SIZ(d_state), SQLT_STR, &pctx->d_state_len);
  OCIBNDPL(pctx->curp0, pctx->d_zip_bp[0],
errhp,":d_zip",d_zip,SIZ(d_zip),
        SQLT_STR, &pctx->d_zip_len);
  OCIBNDPL(pctx->curp0, pctx->c_first_bp[0], errhp,":c_first",c_first,
        SIZ(c_first), SQLT_STR, &pctx->c_first_len);
  OCIBNDPL(pctx->curp0, pctx->c_middle_bp[0],
errhp,":c_middle",c_middle,2,
        SQLT_AFC, &pctx->c_middle_len);
  OCIBNDPL(pctx->curp0, pctx->c_street_1_bp[0],
errhp,":c_street_1",c_street_1,
        SIZ(c_street_1),SQLT_STR, &pctx->c_street_1_len);
  OCIBNDPL(pctx->curp0, pctx->c_street_2_bp[0],
errhp,":c_street_2",c_street_2,
        SIZ(c_street_2),SQLT_STR, &pctx->c_street_2_len);
  OCIBNDPL(pctx->curp0, pctx->c_city_bp[0],
errhp,":c_city",c_city,SIZ(c_city),
        SQLT_STR, &pctx->c_city_len);
  OCIBNDPL(pctx->curp0, pctx->c_state_bp[0], errhp,":c_state",c_state,
        SIZ(c_state), SQLT_STR, &pctx->c_state_len);
  OCIBNDPL(pctx->curp0, pctx->c_zip_bp[0],
errhp,":c_zip",c_zip,SIZ(c_zip),
        SQLT_STR,&pctx->c_zip_len);
  OCIBNDPL(pctx->curp0, pctx->c_phone_bp[0],
errhp,":c_phone",c_phone,
        SIZ(c_phone), SQLT_STR, &pctx->c_phone_len);
  OCIBNDPL(pctx->curp0, pctx->c_since_bp[0],
errhp,":c_since",&c_since,
        SIZ(OCIDate), SQLT_ODT, &pctx->c_since_len);
  OCIBNDPL(pctx->curp0, pctx->c_credit_bp[0],
errhp,":c_credit",c_credit,
        SIZ(c_credit),SQLT_CHR, &pctx->c_credit_len);
  OCIBNDPL(pctx->curp0, pctx->c_credit_lim_bp[0],
errhp,":c_credit_lim",
        ADR(c_credit_lim),SIZ(int), SQLT_INT, &pctx-
>c_credit_lim_len);
  OCIBNDPL(pctx->curp0, pctx->c_discount_bp[0], errhp,":c_discount",
        ADR(c_discount),SIZ(c_discount), SQLT_FLT, &pctx-
>c_discount_len);
#ifdef USE_IEEE_NUMBER
  OCIBNDPL(pctx->curp0, pctx->c_balance_bp[0], errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_BDOUBLE, &pctx-
>c_balance_len);
#else
  OCIBNDPL(pctx->curp0, pctx->c_balance_bp[0], errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_FLT, &pctx->c_balance_len);
#endif /* USE_IEEE_NUMBER */
  OCIBNDPL(pctx->curp0, pctx->c_data_bp[0],
errhp,":c_data",c_data,SIZ(c_data),
        SQLT_STR, &pctx->c_data_len);
/*
  OCIBNDR(pctx->curp0, pctx->h_date_bp,
errhp,":h_date",h_date,SIZ(h_date),
        SQLT_STR, &pctx->h_date_ind, &pctx->h_date_len, &pctx-
>h_date_rc);
*/
  OCIBNDPL(pctx->curp0, pctx->retries_bp[0],
errhp,":retry",ADR(retries),
        SIZ(int), SQLT_INT, &pctx->retries_len);
  OCIBNDPL(pctx->curp0, pctx->cr_date_bp[0],
errhp,":cr_date",ADR(cr_date),
        SIZ(OCIDate),SQLT_ODT, &pctx->cr_date_len);

/* ---- Binds for  the second cursor   */


  OCIBNDPL(pctx->curp1, pctx->w_id_bp[1],
errhp,":w_id",ADR(w_id),SIZ(int),
        SQLT_INT, &pctx->w_id_len);
  OCIBNDPL(pctx->curp1, pctx->d_id_bp[1],
errhp,":d_id",ADR(d_id),SIZ(int),
        SQLT_INT, &pctx->d_id_len);
  OCIBND(pctx->curp1, pctx->c_w_id_bp[1],
errhp,":c_w_id",ADR(c_w_id),SIZ(int),
        SQLT_INT);
  OCIBND(pctx->curp1, pctx->c_d_id_bp[1],
errhp,":c_d_id",ADR(c_d_id),SIZ(int),
        SQLT_INT);
  OCIBNDPL(pctx->curp1, pctx->c_id_bp[1],
errhp,":c_id",ADR(c_id),SIZ(int),
        SQLT_INT, &pctx->c_id_len);
#ifdef USE_IEEE_NUMBER
  OCIBNDPL(pctx->curp1, pctx->h_amount_bp[1],
errhp,":h_amount",ADR(h_amount),
        SIZ(double),SQLT_Bdouble, &pctx->h_amount_len);
#else
  OCIBNDPL(pctx->curp1, pctx->h_amount_bp[1],
errhp,":h_amount",ADR(h_amount),
        SIZ(int),SQLT_INT, &pctx->h_amount_len);
#endif /* USE_IEEE_NUMBER */
  OCIBND(pctx->curp1, pctx->c_last_bp[1],
errhp,":c_last",c_last,SIZ(c_last),
        SQLT_STR);
  OCIBNDPL(pctx->curp1, pctx->w_street_1_bp[1],
errhp,":w_street_1",w_street_1,
        SIZ(w_street_1),SQLT_STR, &pctx->w_street_1_len);
  OCIBNDPL(pctx->curp1, pctx->w_street_2_bp[1],
errhp,":w_street_2",w_street_2,
        SIZ(w_street_2),SQLT_STR, &pctx->w_street_2_len);
  OCIBNDPL(pctx->curp1, pctx->w_city_bp[1],
errhp,":w_city",w_city,SIZ(w_city),
        SQLT_STR, &pctx->w_city_len);
  OCIBNDPL(pctx->curp1, pctx->w_state_bp[1],
errhp,":w_state",w_state,
        SIZ(w_state), SQLT_STR, &pctx->w_state_len);
  OCIBNDPL(pctx->curp1, pctx->w_zip_bp[1],
errhp,":w_zip",w_zip,SIZ(w_zip),
        SQLT_STR, &pctx->w_zip_len);
  OCIBNDPL(pctx->curp1, pctx->d_street_1_bp[1],
errhp,":d_street_1",d_street_1,
        SIZ(d_street_1),SQLT_STR, &pctx->d_street_1_len);
  OCIBNDPL(pctx->curp1, pctx->d_street_2_bp[1],
errhp,":d_street_2",d_street_2,
        SIZ(d_street_2),SQLT_STR, &pctx->d_street_2_len);
  OCIBNDPL(pctx->curp1, pctx->d_city_bp[1],
errhp,":d_city",d_city,SIZ(d_city),
        SQLT_STR, &pctx->d_city_len);
  OCIBNDPL(pctx->curp1, pctx->d_state_bp[1], errhp,":d_state",d_state,
        SIZ(d_state), SQLT_STR, &pctx->d_state_len);
  OCIBNDPL(pctx->curp1, pctx->d_zip_bp[1],
errhp,":d_zip",d_zip,SIZ(d_zip),
        SQLT_STR, &pctx->d_zip_len);
  OCIBNDPL(pctx->curp1, pctx->c_first_bp[1], errhp,":c_first",c_first,
        SIZ(c_first), SQLT_STR, &pctx->c_first_len);
  OCIBNDPL(pctx->curp1, pctx->c_middle_bp[1],
errhp,":c_middle",c_middle,2,
        SQLT_AFC, &pctx->c_middle_len);

  OCIBNDPL(pctx->curp1, pctx->c_street_1_bp[1],
errhp,":c_street_1",c_street_1,
        SIZ(c_street_1),SQLT_STR, &pctx->c_street_1_len);
  OCIBNDPL(pctx->curp1, pctx->c_street_2_bp[1],
errhp,":c_street_2",c_street_2,
        SIZ(c_street_2),SQLT_STR, &pctx->c_street_2_len);
  OCIBNDPL(pctx->curp1, pctx->c_city_bp[1], errhp,":c_city",c_city,
        SIZ(c_city),SQLT_STR, &pctx->c_city_len);
  OCIBNDPL(pctx->curp1, pctx->c_state_bp[1], errhp,":c_state",c_state,
        SIZ(c_state), SQLT_STR, &pctx->c_state_len);
  OCIBNDPL(pctx->curp1, pctx->c_zip_bp[1],
errhp,":c_zip",c_zip,SIZ(c_zip),
        SQLT_STR, &pctx->c_zip_len);
  OCIBNDPL(pctx->curp1, pctx->c_phone_bp[1],
errhp,":c_phone",c_phone,
        SIZ(c_phone), SQLT_STR, &pctx->c_phone_len);
  OCIBNDPL(pctx->curp1, pctx->c_since_bp[1],
errhp,":c_since",&c_since,
        SIZ(OCIDate), SQLT_ODT, &pctx->c_since_len);
  OCIBNDPL(pctx->curp1, pctx->c_credit_bp[1],
errhp,":c_credit",c_credit,
        SIZ(c_credit),SQLT_CHR, &pctx->c_credit_len);
  OCIBNDPL(pctx->curp1, pctx->c_credit_lim_bp[1],
errhp,":c_credit_lim",
        ADR(c_credit_lim),SIZ(int), SQLT_INT, &pctx-
>c_credit_lim_len);
  OCIBNDPL(pctx->curp1, pctx->c_discount_bp[1], errhp,":c_discount",
        ADR(c_discount),SIZ(c_discount), SQLT_FLT, &pctx-
>c_discount_len);
#ifdef USE_IEEE_NUMBER
  OCIBNDPL(pctx->curp1, pctx->c_balance_bp[1], errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_BDOUBLE, &pctx-
>c_balance_len);
#else
  OCIBNDPL(pctx->curp1, pctx->c_balance_bp[1], errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_FLT, &pctx->c_balance_len);
#endif /* USE_IEEE_NUMBER */
  OCIBNDPL(pctx->curp1, pctx->c_data_bp[1],
errhp,":c_data",c_data,SIZ(c_data),
        SQLT_STR, &pctx->c_data_len);
/*
  OCIBNDR(pctx->curp1, pctx->h_date_bp1,
errhp,":h_date",h_date,SIZ(h_date),
        SQLT_STR, &pctx->h_date_ind, &pctx->h_date_len, &pctx-
>h_date_rc);
*/
  OCIBNDPL(pctx->curp1, pctx->retries_bp[1],
errhp,":retry",ADR(retries),
        SIZ(int), SQLT_INT, &pctx->retries_len);
  OCIBNDPL(pctx->curp1, pctx->cr_date_bp[1],
errhp,":cr_date",ADR(cr_date),
        SIZ(OCIDate),SQLT_ODT, &pctx->cr_date_len);

  return (0);
}



void DBExecution::tkvcpdone ()

{
 if(pctx) {
   free(pctx);
 }
}



int DBExecution::tkvcsinit ()
{
  text stmbuf[SQL_BUF_SIZE];
  sctx = (stoctx *)malloc(sizeof(stoctx));
  memset(sctx,(char)0,sizeof(stoctx));

  sctx->norow=0;

  OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&sctx-
>curs,OCI_HTYPE_STMT,0,(dvoid**)0));
  sprintf ((char *) stmbuf, SQLTXTSTO);
  OCIERROR(errhp,OCIStmtPrepare(sctx->curs,errhp,stmbuf,strlen((char
*)stmbuf),
                    OCI_NTV_SYNTAX,OCI_DEFAULT));
#ifndef PLSQLSTO
  OCIERROR(errhp,
    OCIAttrSet(sctx->curs,OCI_HTYPE_STMT,(dvoid*)&sctx->norow,0,
          OCI_ATTR_PREFETCH_ROWS,errhp));
#endif

  /* bind variables */

  OCIBND(sctx->curs,sctx->w_id_bp,errhp, ":w_id",
ADR(w_id),sizeof(int),
       SQLT_INT);
  OCIBND(sctx->curs,sctx->d_id_bp,errhp, ":d_id",
ADR(d_id),sizeof(int),
    SQLT_INT);
#ifdef USE_IEEE_NUMBER
  OCIBND(sctx->curs,sctx->threshold_bp,errhp, ":threshold",
ADR(threshold),
    sizeof(double),SQLT_Bdouble);
#else
  OCIBND(sctx->curs,sctx->threshold_bp,errhp, ":threshold",
ADR(threshold),
    sizeof(int),SQLT_INT);
#endif /* USE_IEEE_NUMBER */
#ifdef PLSQLSTO
  OCIBND(sctx->curs,sctx->low_stock_bp,errhp,":low_stock" ,
ADR(low_stock),
    sizeof(int), SQLT_INT);
#else
  OCIDEFINE(sctx->curs,sctx->low_stock_bp,errhp, 1, ADR(low_stock),
    sizeof(int), SQLT_INT);
#endif

  return (0);
}



void DBExecution::tkvcsdone ()
{
 if(sctx) free(sctx);
}


/***************************************************************
*********************************
* tkvcn  tkvcd  tkvcp  tkvco  tkvcs                              *
***************************************************************
*********************************/


int DBExecution::tkvcn ()
{

  int i;
  int rcount;

retry:

  status = 0;                  /* number of invalid items */

  /* get number of order lines, and check if all are local */

  o_ol_cnt = NITEMS;
  o_all_local = 1;
  for (i = 0; i < NITEMS; i++) {
    if (nol_i_id[i] == 0) {
      o_ol_cnt = i;
      break;
    }
    if (nol_supply_w_id[i] != w_id) {
```

```
#ifdef USE_IEEE_NUMBER
        nctx->s_remote[i] = 1.0;
#else
        nctx->s_remote[i] = 1;
#endif /* USE_IEEE_NUMBER */
        o_all_local = 0;
      }
      else
        nctx->s_remote[i] = 0;
  }

  nctx->w_id_len = sizeof(w_id);
  nctx->d_id_len = sizeof(d_id);
  nctx->c_id_len = sizeof(c_id);
  nctx->o_all_local_len = sizeof(o_all_local);
  nctx->o_ol_cnt_len = sizeof(o_ol_cnt);
  nctx->w_tax_len = 0;
  nctx->d_tax_len = 0;
  nctx->o_id_len = sizeof(o_id);
  nctx->c_discount_len = 0;
  nctx->c_credit_len = 0;
  nctx->c_last_len = 0;
  nctx->retries_len = sizeof(retries);
  nctx->cr_date_len = sizeof(cr_date);
  /* this is the row count */
  rcount = o_ol_cnt;
  nctx->nol_i_count = o_ol_cnt;
  nctx->nol_q_count = o_ol_cnt;
  nctx->nol_s_count = o_ol_cnt;
  nctx->s_remote_count = 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 < o_ol_cnt; i++) {
    nctx->ol_number[i] = i + 1;
    nctx->nol_i_id_len[i] = sizeof(int);
    nctx->nol_supply_w_id_len[i] = sizeof(int);
    nctx->nol_quantity_len[i] = sizeof(int);
    nctx->nol_amount_len[i] = sizeof(int);
    nctx->ol_o_id_len[i] = sizeof(int);
    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;
  }
  for (i = 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;
  }

  execstatus = OCIStmtExecute(tpcsvc,nctx->curn1,errhp,1,0,0,0,
                                          OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);

  if(execstatus != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    errcode = OCIERROR(errhp,execstatus);
    if(errcode == NOT_SERIALIZABLE) {
              retries++;
              goto retry;
    } else if (errcode == RECOVERR) {
              retries++;
              goto retry;
    } else if (errcode == SNAPSHOT_TOO_OLD) {
              retries++;
              goto retry;
    } else {
              return -1;
    }
  }

  /* did the txn succeed ? */
  if (rcount != o_ol_cnt)
  {
    status = rcount - o_ol_cnt;
    o_ol_cnt = rcount;
  }

  total_amount = 0;
  for (i = 0; i < o_ol_cnt; i++) total_amount += nol_amount[i];
  total_amount *= ((double)(1.0 - c_discount)) *
               (double)(1.0 + (double)(d_tax) + (double)(w_tax));
  total_amount = total_amount/100;

  return (0);
}



int DBExecution::tkvcd (int plsqlflag)
{

  int i;
  int rpc,rcount;
  int invalid;

  if (plsqlflag)
  {

    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);
      del_o_id[i] = 0;
    }
    pldctx->del_date_len = DEL_DATE_LEN;
    DISCARD memcpy(&pldctx->del_date,&cr_date,sizeof(OCIDate));

    pldctx->retry=0;

    DISCARD OCIERROR(errhp,
      OCIStmtExecute(tpcsvc,pldctx->curp2,errhp,1,0,NULLP(CONST
OCISnapshot),
                 NULLP(OCISnapshot),OCI_DEFAULT));
    for (i = 0; i < NDISTS; i++)
    {
      del_o_id[i] = 0;
    }
    for (i = 0; i < (int)pldctx->del_o_id_rcnt; i++)
      del_o_id[pldctx->del_d_id[i] - 1] = pldctx->del_o_id[i];
  }
  else
  {


retry:

    invalid = 0;

  /* initialization for array operations */

    for (i = 0; i < NDISTS; i++)
    {
      dctx->del_o_id_ind[i] = TRUE;
      dctx->d_id_ind[i] = TRUE;
      dctx->c_id_ind[i] = TRUE;
      dctx->del_date_ind[i] = TRUE;
      dctx->carrier_id_ind[i] = TRUE;
      dctx->amt_ind[i] = TRUE;

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

      dctx->w_id[i] = w_id;
      dctx->d_id[i] = i+1;
      dctx->carrier_id[i] = o_carrier_id;
      memcpy(&dctx->del_date[i],&cr_date,sizeof(OCIDate));
    }

    memset(actx,(char)0,sizeof(amtctx));

  /* array select from new_order and orders tables */

    execstatus=OCIStmtExecute(tpcsvc,dctx->curd1,errhp,NDISTS,0,
             NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if((execstatus != OCI_SUCCESS) && (execstatus !=
OCI_NO_DATA))
    {
      DISCARD OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      errcode = OCIERROR(errhp,execstatus);
      if(errcode == NOT_SERIALIZABLE)
      {
        retries++;
        goto retry;
      }
      else if (errcode == RECOVERR)
      {
        retries++;
        goto retry;
      }
      else if (errcode == SNAPSHOT_TOO_OLD)
      {
        retries++;
        goto retry;
      }
      else
      {
        return -1;
      }
    }
    /* mark districts with no new order */
    DISCARD OCIAttrGet(dctx-
>curd1,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
            OCI_ATTR_ROW_COUNT,errhp);
    rpc = rcount;
    if (rcount != NDISTS )
    {
      int j = 0;
      for (i=0;i < NDISTS; i++)
      {
        if (dctx->del_o_id_ind[j] == 0) /* there is data here */
          j++;
        else
          shiftdata(j);
      }
    }

    execstatus=OCIStmtExecute(tpcsvc,dctx->curd3,errhp,rpc,0,
             NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if(execstatus != OCI_SUCCESS)
    {
      DISCARD OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      errcode = OCIERROR(errhp,execstatus);
      if(errcode == NOT_SERIALIZABLE)
      {
        retries++;
        goto retry;
      }
      else if (errcode == RECOVERR)
      {
        retries++;
        goto retry;
      }
      else if (errcode == SNAPSHOT_TOO_OLD)
      {
        retries++;
        goto retry;
      }
      else
      {
        return -1;
      }
    }

    DISCARD OCIAttrGet(dctx-
>curd3,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
           OCI_ATTR_ROW_COUNT,errhp);

    if (rcount != rpc)
    {
      userlog ("Error in TPC-C server %d: %d rows selected, %d ords
updated\n",
           proc_no, rpc, rcount);
      DISCARD OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      return (-1);
    }

    /* array update of order_line table */
    execstatus=OCIStmtExecute(tpcsvc,dctx->curd4,errhp,rpc,0,
             NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if(execstatus != OCI_SUCCESS)
    {
      DISCARD OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      errcode = OCIERROR(errhp,execstatus);
      if(errcode == NOT_SERIALIZABLE)
      {
        retries++;
        goto retry;
      }
      else if (errcode == RECOVERR)
      {
        retries++;
        goto retry;
      }
      else if (errcode == SNAPSHOT_TOO_OLD)
      {
        retries++;
        goto retry;
      }
      else
      {
        return -1;
      }
    }
    DISCARD OCIAttrGet(dctx-
>curd4,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
          OCI_ATTR_ROW_COUNT,errhp);
/* transfer amounts  */
    for (i=0;i<rpc;i++)
    {
      dctx->amt[i]=0;
        if ( actx->ol_amt_rcode[i] == 0)
        {
          dctx->amt[i] = actx->ol_amt[i];
        }
    }
#ifdef OLD
    if (rcount > rpc) {
       userlog
         ("Error in TPC-C server %d: %d ordnrs updated, %d ordl
updated\n",
            proc_no, rpc, rcount);
    }
#endif

    /* array update of customer table */
    execstatus=OCIStmtExecute(tpcsvc,dctx->curd6,errhp,rpc,0,
              NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
              OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);

    if(execstatus != OCI_SUCCESS)
    {
      OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
      errcode = OCIERROR(errhp,execstatus);
      if(errcode == NOT_SERIALIZABLE)
      {
        retries++;
        goto retry;
      }
      else if (errcode == RECOVERR)
      {
        retries++;
        goto retry;
      }
      else if (errcode == SNAPSHOT_TOO_OLD)
      {
        retries++;
        goto retry;
      }
      else
      {
        return -1;
      }
    }

    DISCARD OCIAttrGet(dctx-
>curd6,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
                    OCI_ATTR_ROW_COUNT,errhp);

    if (rcount != rpc) {
      userlog ("Error in TPC-C server %d: %d rows selected, %d cust
updated\n",
            proc_no, rpc, rcount);
      DISCARD OCITransRollback(tpcsvc, errhp, OCI_DEFAULT);
      return (-1);
    }

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

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

}



int DBExecution::tkvco ()
{
   int i;
   int rcount;

#if defined(ISO9)
   int secondread = 0;
   char sdate[30];
   ub4  datelen;
   sysdate(sdate);
   printf("Order Status started at: %s\n", sdate);
#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(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(bylastname)
   {
     cbctx.reexec = FALSE;
     execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,100,0,
              NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
     /* will get OCI_NO_DATA if <100 found */
     if ((execstatus != OCI_NO_DATA) && (execstatus !=
OCI_SUCCESS))
     {
       errcode=OCIERROR(errhp, execstatus);
       if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
       {
         DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
         retries++;
         goto retry;
       } else {
         return -1;
       }
     }
     if (execstatus == 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,errhp);
       if (rcount <1)
       {
/*
                              userlog("ORDERSTATUS  rcount=%d\n",rcount);
*/
                              return (-1);
       }
       octx->cust_idx=(rcount)/2 ;
     }
     else
     {
       /* count the number of rows */
       execstatus=OCIStmtExecute(tpcsvc,octx->curo4,errhp,1,0,
                NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
       if ((execstatus != OCI_NO_DATA) && (execstatus !=
OCI_SUCCESS))
       {
         errcode=OCIERROR(errhp, execstatus);
                if ((errcode == NOT_SERIALIZABLE) || (errcode ==
RECOVERR))
         {
           DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
           retries++;
           goto retry;
         } else {
           return -1;
         }
       }
       cbctx.reexec = TRUE;
       cbctx.count = (octx->rcount+1)/2 ;
       execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,cbctx.count,
                           0,NULLP(CONST OCISnapshot),
                           NULLP(OCISnapshot),OCI_DEFAULT);

       DISCARD OCIAttrGet(octx-
>curo0,OCI_HTYPE_STMT,&rcount,NULL,
                             OCI_ATTR_ROW_COUNT,errhp);

       /* will get OCI_NO_DATA if <100 found */
       if ((int)cbctx.count != rcount)
       {
/*
                              userlog ("did not get all rows ");
*/
          return (-1);
       }

       if ((execstatus != OCI_NO_DATA) && (execstatus !=
OCI_SUCCESS))
       {
         errcode=OCIERROR(errhp, execstatus);
         if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
         {
           DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
           retries++;
           goto retry;
         } else {
           return -1;
         }
       }
       octx->cust_idx=cbctx.count - 1 ;
     }

     octx->c_rowid_cust = octx->c_rowid_ptr[octx->cust_idx];
     execstatus=OCIStmtExecute(tpcsvc,octx->curo1,errhp,1,0,
                 NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
     if (execstatus != OCI_SUCCESS)
     {
       errcode=OCIERROR(errhp,execstatus);
         DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
         if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
           || (errcode == SNAPSHOT_TOO_OLD))
         {
           retries++;
           goto retry;
           } else {
           return -1;
         }
       }
     }
     else
     {
       execstatus=OCIStmtExecute(tpcsvc,octx->curo2,errhp,1,0,
                     NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
                     OCI_DEFAULT);
       if (execstatus != OCI_SUCCESS)
       {
         errcode=OCIERROR(errhp,execstatus);
         DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
         if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
           || (errcode == SNAPSHOT_TOO_OLD))
         {
           retries++;
           goto retry;
         }
         else
         {
           return -1;
         }
       }
#ifdef ISO9
       sysdate (sdate);
if (!secondread)
       printf ("---------- FIRST READ RESULT (out) %s ----------\n", sdate);
else
       printf ("---------- SECOND READ RESULT (out) %s ----------\n",
sdate);

       printf ("c_id = %d\n", c_id);
       printf ("c_last = %s\n", c_last);
       printf ("c_first = %s\n", c_first);
       printf ("c_middle = %s\n", c_middle);
       printf ("c_balance = %7.2f\n", (double)c_balance/100);
       printf ("o_id = %d\n", o_id);
       datelen = sizeof(o_entry_d);

OCIERROR(errhp,OCIDateToText(errhp,&o_entry_d_base,(text*)FULLD
ATE,SIZ(FULLDATE),(text*)0,0,&datelen,o_entry_d));
       printf ("o_entry_d = %s\n", o_entry_d);
       printf ("o_carrier_id = %d\n", o_carrier_id);
       printf ("o_ol_cnt = %d\n", o_ol_cnt);
       printf ("---------------------------------------------\n\n", sdate);

if (!secondread) {
       printf ("Sleep before re-read order at: %s\n", sdate);
       sleep (30);
       sysdate (sdate);
       printf ("Wake up and reread at: %s\n", sdate);
       secondread = 1;
       goto retry;
}
#endif /* ISO9 */
     }
     octx->ol_w_id_len = sizeof(int);
     octx->ol_d_id_len = sizeof(int);
     octx->ol_o_id_len = sizeof(int);

     execstatus = OCIStmtExecute(tpcsvc,octx->curo3,errhp,o_ol_cnt,0,
                       NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
                    OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
    if (execstatus != OCI_SUCCESS )
    {
      errcode=OCIERROR(errhp,execstatus);
      DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
      if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
          || (errcode == SNAPSHOT_TOO_OLD))
      {
        retries++;
        goto retry;
      }
      else
      {
        return -1;
      }

    }
   /* clean up and convert the delivery dates */
    for (i = 0; i < o_ol_cnt; i++)
    {
      ol_del_len[i]=sizeof(ol_delivery_d[i]);
      DISCARD OCIERROR(errhp,OCIDateToText(errhp,&ol_d_base[i],
         (const text*)SHORTDATE,(ub1)strlen(SHORTDATE),(text*)0,0,
          &ol_del_len[i], ol_delivery_d[i]));
/*
                cvtdmy(ol_d_base[i],ol_delivery_d[i]);
*/
    }

   return (0);

}



int DBExecution::tkvcp ()
{

retry:

  pctx->w_id_len = SIZ(w_id);
  pctx->d_id_len = SIZ(d_id);
  pctx->c_w_id_len = 0;
  pctx->c_d_id_len = 0;
  pctx->c_id_len = 0;
  pctx->h_amount_len = SIZ(h_amount);
  pctx->c_last_len = SIZ(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 = 0;
  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(retries);
  pctx->cr_date_len = 7;

  w_street_1[0]='\0';
  w_street_2[0]='\0';
  w_city[0]='\0';
  w_state[0]='\0';
  w_zip[0]='\0';
  c_first[0]='\0';
  c_middle[0]='\0';
  c_street_1[0]='\0';
  c_street_2[0]='\0';
  c_city[0]='\0';
  c_state[0]='\0';
  c_zip[0]='\0';
  c_phone[0]='\0';
  c_credit[0]='\0';
  c_credit_lim=0;
  c_discount=0.0;
  c_balance=0.0;
  c_data[0]='\0';
  d_street_1[0]='\0';
  d_street_2[0]='\0';
  d_city[0]='\0';
  d_state[0]='\0';
  d_zip[0]='\0';

  if (bylastname)
                c_id=0;
  else
                c_last[0]='\0';


  if(bylastname) {
    execstatus=OCIStmtExecute(tpcsvc,pctx->curp1,errhp,1,0,
             NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
             OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
  } else {
    execstatus=OCIStmtExecute(tpcsvc,pctx->curp0,errhp,1,0,
             NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
             OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
  }

  if(execstatus != OCI_SUCCESS) {
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     errcode = OCIERROR(errhp,execstatus);
     if(errcode == NOT_SERIALIZABLE) {
        retries++;
        goto retry;
     } else if (errcode == RECOVERR) {
        retries++;
        goto retry;
     } else if (errcode == SNAPSHOT_TOO_OLD) {
        retries++;
        goto retry;
     } else {
        return -1;
     }
   }
   return 0;
}



int DBExecution::tkvcs ()
{

retry:
    execstatus= OCIStmtExecute(tpcsvc,sctx->curs,errhp,1,0,0,0,
                          OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);

                if (execstatus != OCI_SUCCESS)
    {

      errcode=OCIERROR(errhp,execstatus);
      OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
      if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
          || (errcode == SNAPSHOT_TOO_OLD))
      {
        retries++;
        goto retry;
      } else {
        return -1;
      }
     }

  return (0);
}


/************************************************************
**********************************
*  TPCnew  TPCpay  TPCdel  TPCord  TPCsto
*
*************************************************************
**********************************/


int DBExecution::TPCnew (struct newstruct *str)

{

  int i;

  w_id = str->newin.w_id;
  d_id = str->newin.d_id;
  c_id = str->newin.c_id;
  for (i = 0; i < 15; i++) {
    nol_i_id[i] = str->newin.ol_i_id[i];
    nol_supply_w_id[i] = str->newin.ol_supply_w_id[i];
    nol_quantity[i] = str->newin.ol_quantity[i];
  }
  retries = 0;

#ifndef AVOID_DEADLOCK

  for (i = NITEMS; i > 0; i--) {
    if (nol_i_id[i-1] > 0) {
      ordl_cnt = i;
      break;
    }
  }

  for (i = 0; i < NITEMS; i++) indx[i] = i;

  q_sort(nol_i_id, str, 0, ordl_cnt-1);

#endif

/*
  vgetdate(cr_date); */

  OCIERROR(errhp,OCIDateSysDate(errhp,&cr_date));

  if (str->newout.terror = tkvcn ()) {
    if (str->newout.terror != RECOVERR)
      str->newout.terror = IRRECERR;
      return (-1);
    }

  /* fill in date for o_entry_d from time in beginning of txn*/
/*
  cvtdmyhms(cr_date,o_entry_d);
*/
   datelen = sizeof(o_entry_d);
   OCIERROR(errhp,

OCIDateToText(errhp,&cr_date,(text*)FULLDATE,SIZ(FULLDATE),(te
xt*)0,0,
                  &datelen,o_entry_d));

  str->newout.terror = NOERR;
  str->newout.o_id = o_id;
  str->newout.o_ol_cnt = o_ol_cnt;
  strncpy (str->newout.c_last, c_last, 17);
  strncpy (str->newout.c_credit, c_credit, 3);
  str->newout.c_discount = c_discount;
  str->newout.w_tax = (double)(w_tax);
  str->newout.d_tax = (double)(d_tax);
  strncpy (str->newout.o_entry_d, (char*)o_entry_d, 20);
  str->newout.total_amount = total_amount;
  for (i = 0; i < o_ol_cnt; i++) {
    strncpy (str->newout.i_name[i], i_name[i], 25);
    str->newout.brand_generic[i] = brand_generic[i][0];
#ifdef USE_IEEE_NUMBER
    str->newout.s_quantity[i] = (int) s_quantity[i];
    str->newout.i_price[i] = i_price[i]/100;
    str->newout.ol_amount[i] = nol_amount[i]/100;
#else
    str->newout.s_quantity[i] = s_quantity[i];
    str->newout.i_price[i] = (double)(i_price[i])/100;
    str->newout.ol_amount[i] = (double)(nol_amount[i])/100;
#endif /* USE_IEEE_NUMBER */
  }

#ifndef AVOID_DEADLOCK
   q_sort(indx, str, 0, ordl_cnt-1);
#endif

  if (status)
    strcpy (str->newout.status, "Item number is not valid");
  else
    str->newout.status[0] = '\0';
  str->newout.retry = retries;
  return(1);
}



int DBExecution::TPCpay (struct paystruct *str)
{

  w_id = str->payin.w_id;
  d_id = str->payin.d_id;
  c_w_id = str->payin.c_w_id;
  c_d_id = str->payin.c_d_id;
#ifdef USE_IEEE_NUMBER
  h_amount = (double) str->payin.h_amount;
#else
  h_amount = str->payin.h_amount;
#endif /* USE_IEEE_NUMBER */
  bylastname = str->payin.bylastname;

/*
  vgetdate(cr_date);  */
  OCIERROR(errhp,OCIDateSysDate(errhp,&cr_date));
  if (bylastname) {
    c_id = 0;
    strncpy (c_last, str->payin.c_last, 17);
  }
  else {
    c_id = str->payin.c_id;
    strcpy (c_last, " ");
  }
  retries = 0;

  if (str->payout.terror = tkvcp ()) {
    if (str->payout.terror != RECOVERR)
      str->payout.terror = IRRECERR;
    return (-1);
  }

/*
  cvtdmyhms(cr_date,h_date);
*/
        hlen=SIZ(h_date);
        OCIERROR(errhp,OCIDateToText(errhp,&cr_date,

(text*)FULLDATE,strlen(FULLDATE),(text*)0,0,&hlen,h_date));

/*
  cvtdmy(c_since,c_since_d);
*/
        sincelen=SIZ(c_since_d);
        OCIERROR(errhp,OCIDateToText(errhp,&c_since,

(text*)SHORTDATE,strlen(SHORTDATE),(text*)0,0,&sincelen,c_since_
d));



  str->payout.terror = NOERR;
  strncpy (str->payout.w_street_1, w_street_1, 21);
  strncpy (str->payout.w_street_2, w_street_2, 21);
  strncpy (str->payout.w_city, w_city, 21);
  strncpy (str->payout.w_state, w_state, 3);
  strncpy (str->payout.w_zip, w_zip, 10);
  strncpy (str->payout.d_street_1, d_street_1, 21);
  strncpy (str->payout.d_street_2, d_street_2, 21);
  strncpy (str->payout.d_city, d_city, 21);
  strncpy (str->payout.d_state, d_state, 3);
  strncpy (str->payout.d_zip, d_zip, 10);
  str->payout.c_id = c_id;
  strncpy (str->payout.c_first, c_first, 17);
  strncpy (str->payout.c_middle, c_middle, 3);
  strncpy (str->payout.c_last, c_last, 17);
  strncpy (str->payout.c_street_1, c_street_1, 21);
  strncpy (str->payout.c_street_2, c_street_2, 21);
  strncpy (str->payout.c_city, c_city, 21);
  strncpy (str->payout.c_state, c_state, 3);
  strncpy (str->payout.c_zip, c_zip, 10);
  strncpy (str->payout.c_phone, c_phone, 17);
  strncpy (str->payout.c_since, (char*)c_since_d, 11);
  strncpy (str->payout.c_credit, c_credit, 3);
  str->payout.c_credit_lim = (double)(c_credit_lim)/100;
  str->payout.c_discount = c_discount;
  str->payout.c_balance = (double)(c_balance)/100;
  strncpy (str->payout.c_data, c_data, 201);
  strncpy (str->payout.h_date, (char*)h_date, 20);
  str->payout.retry = retries;
  return(1);
}



int DBExecution::TPCord (struct ordstruct *str)
{

  int i;
  w_id = str->ordin.w_id;
  d_id = str->ordin.d_id;
  bylastname = str->ordin.bylastname;
  if (bylastname) {
    c_id = 0;
    strncpy (c_last, str->ordin.c_last, 17);
  }
  else {
    c_id = str->ordin.c_id;
    strcpy (c_last, " ");
  }
  retries = 0;

  if (str->ordout.terror = tkvco ()) {
    if (str->ordout.terror != RECOVERR)
      str->ordout.terror = IRRECERR;
    return (-1);
  }

  datelen = sizeof(o_entry_d);
  OCIERROR(errhp,

OCIDateToText(errhp,&o_entry_d_base,(text*)FULLDATE,SIZ(FULLD
ATE),(text*)0,0,
                 &datelen,o_entry_d));

  str->ordout.terror = NOERR;
  str->ordout.c_id = c_id;
  strncpy (str->ordout.c_last, c_last, 17);
  strncpy (str->ordout.c_first, c_first, 17);
  strncpy (str->ordout.c_middle, c_middle, 3);
  str->ordout.c_balance = c_balance/100;
  str->ordout.o_id = o_id;
  strncpy (str->ordout.o_entry_d, (char*)o_entry_d, 20);
  if ( o_carrier_id == 11 )
     str->ordout.o_carrier_id = 0;
  else
     str->ordout.o_carrier_id = o_carrier_id;
  str->ordout.o_ol_cnt = o_ol_cnt;
  for (i = 0; i < o_ol_cnt; i++) {
    ol_delivery_d[i][10] = '\0';
    if ( !strcmp((char*)ol_delivery_d[i],"15-09-1911") )
              strncpy((char*)ol_delivery_d[i],"NOT DELIVR",10);
    str->ordout.ol_supply_w_id[i] = ol_supply_w_id[i];
    str->ordout.ol_i_id[i] = ol_i_id[i];
#ifdef USE_IEEE_NUMBER
    str->ordout.ol_quantity[i] = (int) ol_quantity[i];
    str->ordout.ol_amount[i] = ol_amount[i]/100;
#else
    str->ordout.ol_quantity[i] = ol_quantity[i];
    str->ordout.ol_amount[i] = (double)(ol_amount[i])/100;
#endif /* USE_IEEE_NUMBER */
    strncpy (str->ordout.ol_delivery_d[i], (char*)ol_delivery_d[i], 11);
  }
  str->ordout.retry = retries;
  return(1);
}



int DBExecution::TPCdel (struct delstruct *str)
{
  int i;

  w_id = str->delin.w_id;
  o_carrier_id = str->delin.o_carrier_id;
  retries = 0;
```

```
/*
   vgetdate(cr_date);  */
   OCIERROR(errhp,OCIDateSysDate(errhp,&cr_date));

   if (str->delout.terror = tkvcd (str->delin.plsqlflag)) {
     if(str->delout.terror == DEL_ERROR)
       return DEL_ERROR;
      if (str->delout.terror != RECOVERR)
         str->delout.terror = IRRECERR;
      return (-1);
   }

   for (i = 0; i < 10; i++) {
      if (del_o_id[i] <= 0) {
         userlog ("DELIVERY: no new order for w_id: %d, d_id %d\n",
                  w_id, i + 1);
      }
   }
   str->delout.terror = NOERR;
   str->delout.retry = retries;
   return(1);
}



int DBExecution::TPCsto (struct stostruct *str)
{
   w_id = str->stoin.w_id;
   d_id = str->stoin.d_id;
#ifdef USE_IEEE_NUMBER
   threshold = (double) str->stoin.threshold;
#else
   threshold = str->stoin.threshold;
#endif /* USE_IEEE_NUMBER */
   retries = 0;

   if (str->stoout.terror = tkvcs ()) {
      if (str->stoout.terror != RECOVERR)
         str->stoout.terror = IRRECERR;
      return (-1);
   }


   str->stoout.terror = NOERR;
   str->stoout.low_stock = low_stock;
   str->stoout.retry = retries;
   return(1);
}



#ifndef AVOID_DEADLOCK

void DBExecution::q_sort(int *arr,struct newstruct *str,int left, int right)
{
  int i, last;

  if(left >= right)
    return;
  swap(str,left,(left+right)/2);
  last = left;
  for(i=left+1;i<=right;i++)
    if(arr[i] < arr[left])
      swap(str,last,i);
  swap(str,left,last);
  q_sort(arr,str,left,last-1);
  q_sort(arr,str,last+1,right);
}

void DBExecution::swap(struct newstruct *str, int i, int j)
{
  int temp;
  double tempf;
  char tmpstr[25];
  char tmpch;
#ifdef USE_IEEE_NUMBER
  double temp_double;
#endif;

  temp = indx[i];
  indx[i] = indx[j];
  indx[j] = temp;

  temp = nol_i_id[i];
  nol_i_id[i] = nol_i_id[j];
  nol_i_id[j] = temp;

  temp = nol_supply_w_id[i];
  nol_supply_w_id[i] = nol_supply_w_id[j];
  nol_supply_w_id[j] = temp;

#ifdef USE_IEEE_NUMBER
  temp_double = nol_quantity[i];
  nol_quantity[i] = nol_quantity[j];
  nol_quantity[j] = temp_double;

  temp_double = str->newout.i_price[i];
  str->newout.i_price[i] = str->newout.i_price[j];
  str->newout.i_price[j] = temp_double;

  temp_double = str->newout.ol_amount[i];
  str->newout.ol_amount[i] = str->newout.ol_amount[j];
  str->newout.ol_amount[j] = temp_double;

  temp_double = (double)str->newout.s_quantity[i];
  str->newout.s_quantity[i] = str->newout.s_quantity[j];
  str->newout.s_quantity[j] = (int)temp_double;
#else
  temp = nol_quantity[i];
  nol_quantity[i] = nol_quantity[j];
  nol_quantity[j] = temp;

  tempf = str->newout.i_price[i];
  str->newout.i_price[i] = str->newout.i_price[j];
  str->newout.i_price[j] = tempf;

  tempf = str->newout.ol_amount[i];
  str->newout.ol_amount[i] = str->newout.ol_amount[j];
  str->newout.ol_amount[j] = tempf;

  temp = str->newout.s_quantity[i];
  str->newout.s_quantity[i] = str->newout.s_quantity[j];
  str->newout.s_quantity[j] = temp;
#endif /* USE_IEEE_NUMBER */

  strncpy(tmpstr,str->newout.i_name[i], 25);
  strncpy(str->newout.i_name[i],str->newout.i_name[j], 25);
  strncpy(str->newout.i_name[j],tmpstr, 25);


  tmpch = str->newout.brand_generic[i];
  str->newout.brand_generic[i] = str->newout.brand_generic[j];
  str->newout.brand_generic[j] = tmpch;
}

#endif
```

```
#ifdef LOOPBACK

int mod_tpcc_neworder(T_neworder_data *output)
{
		Sleep(18);
		output->txn_status= DB_RETURN_OCI_SUCCESS;
		output->d_id=1;
		output->c_id=1;
		output->o_ol_cnt=7;
		output->o_all_local=0;
		strcpy(output->o_entry_d.DateString, "20-01-2004 11:59:10");
		strcpy(output->c_last, "TESTLASTNAME<>\"&");
		strcpy(output->c_credit, "GC");
		output->c_discount=.1791;
		output->w_tax=.093099996;
		output->d_tax=.159700006;
		output->o_id=2101;

		output->o_orderline[0].ol_i_id=98752;
		output->o_orderline[0].ol_supply_w_id=2;
		output->o_orderline[0].ol_quantity=5;
		output->o_orderline[0].ol_amount=2576.48;
		output->o_orderline[0].i_price=3.71;
		output->o_orderline[0].s_quantity=45;
		strcpy(output->o_orderline[0].i_name,  "item98752");
		output->o_orderline[0].b_g[0]='G';

		output->o_orderline[1].ol_i_id=80479;
		output->o_orderline[1].ol_supply_w_id=1;
		output->o_orderline[1].ol_quantity=6;
		output->o_orderline[1].ol_amount=3490.03;
		output->o_orderline[1].i_price=6.81;
		output->o_orderline[1].s_quantity=58;
		strcpy(output->o_orderline[1].i_name,  "item80479");
		output->o_orderline[1].b_g[0]='G';

		output->o_orderline[2].ol_i_id=58617;
		output->o_orderline[2].ol_supply_w_id=1;
		output->o_orderline[2].ol_quantity=6;
		output->o_orderline[2].ol_amount=1234.56;
		output->o_orderline[2].i_price=4.01;
		output->o_orderline[2].s_quantity=22;
		strcpy(output->o_orderline[2].i_name,  "item58617");
		output->o_orderline[2].b_g[0]='G';

		output->o_orderline[3].ol_i_id=3394;
		output->o_orderline[3].ol_supply_w_id=1;
		output->o_orderline[3].ol_quantity=5;
		output->o_orderline[3].ol_amount=2345.67;
		output->o_orderline[3].i_price=1.73;
		output->o_orderline[3].s_quantity=18;
		strcpy(output->o_orderline[3].i_name,  "item3394");
		output->o_orderline[3].b_g[0]='G';

		output->o_orderline[4].ol_i_id=2242;
		output->o_orderline[4].ol_supply_w_id=1;
		output->o_orderline[4].ol_quantity=4;
		output->o_orderline[4].ol_amount=3456.78;
		output->o_orderline[4].i_price=4.48;
		output->o_orderline[4].s_quantity=29;
		strcpy(output->o_orderline[4].i_name,  "item2242");
		output->o_orderline[4].b_g[0]='G';

		output->o_orderline[6].ol_i_id=37310;
		output->o_orderline[6].ol_supply_w_id=1;
		output->o_orderline[6].ol_quantity=5;
		output->o_orderline[6].ol_amount=4567.89;
		output->o_orderline[6].i_price=5.50;
		output->o_orderline[6].s_quantity=21;
		strcpy(output->o_orderline[6].i_name,  "item37310");
		output->o_orderline[6].b_g[0]='G';

		output->o_orderline[5].ol_i_id=19395;
		output->o_orderline[5].ol_supply_w_id=3;
		output->o_orderline[5].ol_quantity=6;
		output->o_orderline[5].ol_amount=5678.90;
		output->o_orderline[5].i_price=10.19;
		output->o_orderline[5].s_quantity=80;
		strcpy(output->o_orderline[5].i_name,  "item19395");
		output->o_orderline[5].b_g[0]='G';

		return SUCCESS;
}



int mod_tpcc_payment(T_payment_data *output)
{
  int i;
  char c;

  Sleep(10);
  output->txn_status= DB_RETURN_OCI_SUCCESS;
  output->d_id=2;
  output->c_id=99;
  strcpy(output->c_last, "paymentCLast");
  output->c_w_id=2;
  output->c_d_id=5;
  output->h_amount=54321.09;
  strcpy(output->h_date.DateString, "20-01-2004 11:59:10");
  strcpy(output->w_street_1, "WareStreet1");
  strcpy(output->w_street_2, "WareStreet2");
  strcpy(output->w_city, "WareCity");
  strcpy(output->w_state, "WareState");
  strcpy(output->w_zip, "WareZip");
  strcpy(output->d_street_1, "DistStreet1");
  strcpy(output->d_street_2, "DistStreet2");
  strcpy(output->d_city, "DistCity");
  strcpy(output->d_state, "DistState");
  strcpy(output->d_zip, "DistZip");
  strcpy(output->c_first, "CFirst");
  strcpy(output->c_middle, "PA");
  strcpy(output->c_street_1, "CustStreet1");
  strcpy(output->c_street_2, "CustStreet2");
  strcpy(output->c_city, "CustCity");
  strcpy(output->c_state, "CustState");
  strcpy(output->c_zip, "CustZip");
  strcpy(output->c_phone, "9876543");
  strcpy(output->c_since.DateString, "20-01-2004 11:59:05");
  strcpy(output->c_credit, "BC");
  output->c_credit_lim=34567.89;
  output->c_discount=.234;
  output->c_balance=876543.21;

  for (i=0, c='a'; i<143; i++, c++) {
    if (c=='z') c='a';
    output->c_data[i]=(char) c;
  }
  return SUCCESS;
}



int mod_tpcc_delivery(T_delivery_data *output, int id)
{
  Sleep(1);
  output->txn_status= DB_RETURN_OCI_SUCCESS;
  output->o_carrier_id=4;
  write_delivery_log(output, id);
  return SUCCESS;
}



int mod_tpcc_orderstatus(T_orderstatus_data *output)
{
  Sleep(7);
  output->txn_status= DB_RETURN_OCI_SUCCESS;
  output->d_id=8;
  output->c_id=4321;
  strcpy(output->c_last, "orderstatusCLast");
  strcpy(output->c_first, "CFirst");
  strcpy(output->c_middle, "OS");
  output->c_balance=7543.21;
  output->o_id=9832;
  output->o_ol_cnt=5;
  output->o_carrier_id=2;
  strcpy(output->o_entry_d.DateString, "20-01-2004 11:59:08");

  output->o_orderline[0].ol_i_id=98752;
  output->o_orderline[0].ol_supply_w_id=2;
  output->o_orderline[0].ol_quantity=5;
  output->o_orderline[0].ol_amount=2576.48;
  strcpy(output->o_orderline[0].ol_delivery_d.DateString, "20-01-2004
11:58:00");

  output->o_orderline[1].ol_i_id=80479;
  output->o_orderline[1].ol_supply_w_id=1;
  output->o_orderline[1].ol_quantity=6;
  output->o_orderline[1].ol_amount=3490.03;
  strcpy(output->o_orderline[1].ol_delivery_d.DateString, "20-01-2004
11:58:01");

  output->o_orderline[2].ol_i_id=58617;
  output->o_orderline[2].ol_supply_w_id=1;
  output->o_orderline[2].ol_quantity=6;
  output->o_orderline[2].ol_amount=1234.56;
  strcpy(output->o_orderline[2].ol_delivery_d.DateString, "20-01-2004
11:58:02");

  output->o_orderline[3].ol_i_id=3394;
  output->o_orderline[3].ol_supply_w_id=1;
  output->o_orderline[3].ol_quantity=5;
  output->o_orderline[3].ol_amount=2345.67;
  strcpy(output->o_orderline[3].ol_delivery_d.DateString, "20-01-2004
11:58:03");

  output->o_orderline[4].ol_i_id=2242;
  output->o_orderline[4].ol_supply_w_id=1;
  output->o_orderline[4].ol_quantity=4;
  output->o_orderline[4].ol_amount=3456.78;
  strcpy(output->o_orderline[4].ol_delivery_d.DateString, "20-01-2004
11:58:04");

  return SUCCESS;
}



int mod_tpcc_stocklevel(T_stocklevel_data *output)
{
  Sleep(4);
  output->threshold=10;
  output->low_stock=1;
  output->txn_status= DB_RETURN_OCI_SUCCESS;
  return SUCCESS;
}
```

```
#endif



- - - - - - - - - - - - - - - - - -
DBConnection.h
- - - - - - - - - - - - - - - - - -
#include "tpccpl.h"
#include "tpccstruct.h"
#include "tpcc_struct.h"
#include "mod_tpcc_error.h"
#include "mod_tpcc.h"


#define MAXLEN 100
#define LogName "log\\DBConnection.log"
#define InitName "DBInit.ini"

// Execution Pool Status
#define IDLE 1
#define IN_USE 2

#define Default_DBConnections "20"
#define DelLogName "log\\DeliveryLog"

#define convert_status(A,B) \
{\
  switch (B) { \
    case OCI_SUCCESS: (A)=DB_RETURN_OCI_SUCCESS; break; \
    case OCI_SUCCESS_WITH_INFO:
(A)=DB_RETURN_OCI_SUCCESS_WITH_INFO; break; \
    case OCI_NEED_DATA: (A)=DB_RETURN_OCI_NEED_DATA;
break; \
    case OCI_NO_DATA: (A)=DB_RETURN_OCI_NO_DATA; break; \
    case OCI_ERROR: (A)=DB_RETURN_OCI_ERROR; break; \
    case OCI_INVALID_HANDLE:
(A)=DB_RETURN_OCI_INVALID_HANDLE; break; \
    case OCI_STILL_EXECUTING:
(A)=DB_RETURN_OCI_STILL_EXECUTING; break; \
    case OCI_CONTINUE: (A)=DB_RETURN_OCI_CONTINUE; break; \
  }; \
}

/***************************************************************
**********************************
* DBExecution_pool_info                                       *
***************************************************************
**********************************/


typedef struct _DBExecution_pool_info {

        int current_status;
        int neworder_count;
        int payment_count;
        int orderstatus_count;
        int delivery_count;
        int stocklevel_count;
        void *pointer;

} DBExecution_pool_info;


/***************************************************************
**********************************
* global functions                                            *
***************************************************************
**********************************/
```

```
sb4 no_data(dvoid *,OCIBind *,ub4,ub4,dvoid **,ub4 *,ub1 *,dvoid **);
sb4 TPC_oid_data(dvoid *,OCIBind *,ub4,ub4,dvoid **,ub4 **,ub1
*,dvoid **,ub2 **);
sb4 cid_data(dvoid *,OCIBind *,ub4,ub4,dvoid **,ub4 **,ub1 *,dvoid
**,ub2 **);
sb4 amt_data(dvoid *,OCIBind *,ub4,ub4,dvoid **,ub4 **,ub1 *,dvoid
**,ub2 **);
void userlog (char *, ...);
void readInit(char *, char *, char *);
int initializeDBExecutionPool();

DBExecution_pool_info* findIdleDBExecution();
int freeDBExecution(DBExecution_pool_info *);

//DBExecution_pool_info* findIdleDBExecution(HANDLE *);
//int freeDBExecution(DBExecution_pool_info *, HANDLE *);

void write_delivery_log(T_delivery_data *pdata, int id);
void initDelLog(int);
void endDelLog(int);


/************************************************************
**********************************
* global variables                                          *
*************************************************************
**********************************/

HANDLE waitIdle;
HANDLE *DBExecution_lock;
CRITICAL_SECTION  *DBExecution_CS;
DWORD TlsPtr;
DBExecution_pool_info *DBExecution_pool;
char DllPath[MAXLEN];
char LogFile[MAXLEN];
char InitFile[MAXLEN];
char DelLogFile[MAXLEN];
int TotalLoop=0;
int findDBExecutionCall=0;
int findDBExecutionWait=0;
int DBConnections;
int ready=0;
FILE **DelFiles;


/************************************************************
**********************************
* DBExecution                                               *
*************************************************************
**********************************/

class DBExecution
{
        public:
                DBExecution();
                ~DBExecution();

                int TPCinit(int, char *, char *);
                int TPCnew(struct newstruct *);
                int TPCpay(struct paystruct *);
                int TPCdel(struct delstruct *);
                int TPCord(struct ordstruct *);
                int TPCsto(struct stostruct *);
                void TPCexit();

#ifndef AVOID_DEADLOCK
                void swap(struct newstruct *, int, int);
   void q_sort(int *, struct newstruct *, int, int);
#endif
                int ocierror(char *, int, OCIError *, sword);
                void shiftdata(int);
                int sqlfile(char *, text *);

                int tkvcninit();
                int tkvcn();
                void tkvcndone();

                int tkvcpinit();
                int tkvcp();
                void tkvcpdone();

                int tkvcoinit();
                int tkvco();
                void tkvcodone();

                int tkvcdinit(int);
                int tkvcd(int);
                void tkvcddone(int);

                int tkvcsinit();
                int tkvcs();
                void tkvcsdone();

                delctx *dctx;
                int execstatus;
                int status;
                int del_o_id[10];


        private:
                int proc_no;
                int logon;
                int new_init;
                int pay_init;
                int ord_init;
                int del_init_oci;
                int del_init_plsql;
                int sto_init;
                int errcode;
                int indx[NITEMS];
                int ordl_cnt;

        /* for stock-level transaction */

                int w_id;
        int d_id;
        int c_id;
#ifdef USE_IEEE_NUMBER
        double threshold;
#else
        int threshold;
#endif /* USE_IEEE_NUMBER */
                int low_stock;

/* for delivery transaction */

                int retries;

/* for order-status transaction */

                int bylastname;
                char c_last[17];
                char c_first[17];
                char c_middle[3];
                double c_balance;
                int o_id;
                text o_entry_d[20];
                ub4  datelen;
                int o_carrier_id;
```

```
                        int o_ol_cnt;
                        int ol_supply_w_id[15];
                        int ol_i_id[15];
#ifdef USE_IEEE_NUMBER
                        double ol_quantity[15];
                        double ol_amount[15];
#else
                        int ol_quantity[15];
                        int ol_amount[15];
#endif /* USE_IEEE_NUMBER */
                        ub4  ol_del_len[15];
                        text ol_delivery_d[15][11];
                        OCIRowid *o_rowid;

/* for payment transaction */

                        int c_w_id;
                        int c_d_id;
#ifdef USE_IEEE_NUMBER
                        double h_amount;
#else
                        int h_amount;
#endif /* USE_IEEE_NUMBER */
                        char w_street_1[21];
                        char w_street_2[21];
                        char w_city[21];
                        char w_state[3];
                        char w_zip[10];
                        char d_street_1[21];
                        char d_street_2[21];
                        char d_city[21];
                        char d_state[3];
                        char d_zip[10];
                        char c_street_1[21];
                        char c_street_2[21];
                        char c_city[21];
                        char c_state[3];
                        char c_zip[10];
                        char c_phone[17];
                        ub4  sincelen;
                        text c_since_d[11];
                        double c_discount;
                        char c_credit[3];
                        int c_credit_lim;
                        char c_data[201];
                        ub4 hlen;
                        text h_date[20];

/* for new order transaction */

                        int nol_i_id[15];
                        int nol_supply_w_id[15];
#ifdef USE_IEEE_NUMBER
                        double nol_quantity[15];
                        double nol_amount[15];
                        double s_quantity[15];
                        double i_price[15];
#else
                        int nol_quantity[15];
                        int nol_amount[15];
                        int s_quantity[15];
                        int i_price[15];
#endif /* USE_IEEE_NUMBER */
                        int nol_quanti10[15];
                        int nol_quanti91[15];
                        int nol_ytdqty[15];
                        int o_all_local;
                        double w_tax;
                        double d_tax;
                        double total_amount;
                        char i_name[15][25];
                        char brand_gen[15];
                        char brand_generic[15][1];
                        int tracelevel;

                        OCIDate cr_date;
                        OCIDate c_since;
                        OCIDate o_entry_d_base;
                        OCIDate ol_d_base[15];
                        dvoid *xmem;

                        OCIEnv *tpcenv;
                        OCIServer *tpcsrv;
                        OCIError *errhp;
                        OCISvcCtx *tpcsvc;
                        OCISession *tpcusr;
                        OCIStmt *curi;


                        newctx *nctx;
                        ordctx *octx;
                        defctx cbctx;
                        pldelctx *pldctx;
                        amtctx *actx;
                        payctx *pctx;
                        stoctx *sctx;
};
```

- - - - - - - - - - - - - - - - - -
loopback.cpp
- - - - - - - - - - - - - - - - - -

```
#include "stdafx.h"
#include "DBConnection.h"
```

- - - - - - - - - - - - - - - - - -
modtpcc.cpp
- - - - - - - - - - - - - - - - - -

```
// modtpcc.cpp : Defines the entry point for the DLL application.
//

#include "stdafx.h"
#include "modtpcc.h"
#include <httpext.h>

//#define DEBUG
//#define DELIVERY_MUTEX
#define NEW_ALLOCATE_FORM

BOOL APIENTRY DllMain( HANDLE hModule,
                DWORD  ul_reason_for_call,
                LPVOID lpReserved
                                                        )
{
        char string[MAXLEN];

        if (ul_reason_for_call == DLL_PROCESS_ATTACH) {
       int i;
                        GetModuleFileName((HMODULE)hModule,
DllPath, MAXLEN-1);

                        strcpy(origin, DllPath);
                        if (DllPath[0]=='\\' && DllPath[1]=='\\' &&
DllPath[2]=='?' && DllPath[3]=='\\')
```

```
				strcpy(DllPath, DllPath+4);
		for (i=strlen(DllPath); DllPath[i]!='\\' && i; i--);
			DllPath[i]='\0';
			sprintf(InitFile, "%s\\%s", DllPath, InitName);
			sprintf(DllFile, "%s\\%s", DllPath, DllName);
			sprintf(LogFile, "%s\\%s", DllPath, LogName);

	OCIInitialize(OCI_THREADED|OCI_OBJECT,(dvoid *)0,0,0,0);

			//			sprintf(LogFile, "d:\\%s",
LogName);

			/* load DBConnection.dll */


			if ((dllinstance = LoadLibrary(DllFile)) == NULL)
				return FALSE;


			if ((mod_tpcc_neworder=(int
(FAR*)(T_neworder_data *)) GetProcAddress((HMODULE)dllinstance,
"mod_tpcc_neworder"))==NULL)
				return FALSE;

			if ((mod_tpcc_payment=(int
(FAR*)(T_payment_data *)) GetProcAddress((HMODULE)dllinstance,
"mod_tpcc_payment"))==NULL)
				return FALSE;

			if ((mod_tpcc_delivery=(int
(FAR*)(T_delivery_data *, int)) GetProcAddress((HMODULE)dllinstance,
"mod_tpcc_delivery"))==NULL)
				return FALSE;

			if ((mod_tpcc_orderstatus=(int
(FAR*)(T_orderstatus_data *)) GetProcAddress((HMODULE)dllinstance,
"mod_tpcc_orderstatus"))==NULL)
				FALSE;

			if ((mod_tpcc_stocklevel=(int
(FAR*)(T_stocklevel_data *)) GetProcAddress((HMODULE)dllinstance,
"mod_tpcc_stocklevel"))==NULL)
				return FALSE;

			if ((userlog=(void (FAR*)(char * str, ...))
GetProcAddress((HMODULE)dllinstance, "userlog"))==NULL)
				return FALSE;

			if ((initDelLog=(void (FAR*)(int))
GetProcAddress((HMODULE)dllinstance, "initDelLog"))==NULL)
				return FALSE;

			if ((endDelLog=(void (FAR*)(int))
GetProcAddress((HMODULE)dllinstance, "endDelLog"))==NULL)
				return FALSE;

			userlog("load modtpcc.dll, DllPath: %s\n", DllPath);


			if ((TlsPointer = TlsAlloc()) == 0xFFFFFFFF) {
				userlog("Error during TlsAlloc\n");
				return FALSE;
			}
			InitializeCriticalSection(&critical_initDelQueue);
			InitializeCriticalSection(&critical_memory);
			InitializeCriticalSection(&critical_DelQueue_free);
			InitializeCriticalSection(&critical_DelQueue_work);

			/* read ini parameters */
			readInit(string, "DBConnections",
Default_DBConnections);
			DBConnections = atoi(string);
			userlog("number of DBConnections is %d\n",
DBConnections);

#ifdef NEW_ALLOCATE_FORM
			readInit(string, "StartTerm", Default_StartTerm);
			userlog("number of Start Term is %s\n", string);
			/* StartTerm starts from 1 */
			if ((StartTerm = atoi(string) ) < 0) {
				userlog("error: Start Term is %d\n",
StartTerm);
				return FALSE;
			}

			/* w_id starts from 1, d_id starts from 1 */
			StartTerm+=10;
#endif

			readInit(string, "KMaxterms", Default_Maxterms);
			userlog("number of Max Terms is %s000\n", string);
			/* add one more form for special characters */
			if ((Maxterms = atoi(string) * 1000 + 1) <= 1) {
				userlog("number of Max Terms is %d\n",
Maxterms - 1);
				return FALSE;
			}
			readInit(string, "DeliveryQueues",
Default_DeliveryQueues);
			userlog("number of Delivery Queues is %s\n",
string);
			if ((DeliveryQueues = atoi(string)) <= 0) {
				userlog("number of Delivery Queues is
%d\n", DeliveryQueues);
				return FALSE;
			}

			readInit(string, "DeliveryThreads",
Default_DeliveryThreads);
			userlog("number of Delivery Threads is %s\n",
string);
			if ((DeliveryThreads = atoi(string)) <= 0) {
				userlog("number of Delivery Threads is
%d\n", DeliveryThreads);
				return FALSE;
			}
			initDelLog(DeliveryThreads);

			modtpcc_ready=1;
		}
		else if (ul_reason_for_call == DLL_PROCESS_DETACH) {

			endDelLog(DeliveryThreads);

			if ((TlsFree(TlsPointer)) == NULL) {
				userlog("Error during TlsFree\n");
			  return FALSE;
			}
			if (!deleteDelQueue())
			{
				userlog("Error during
deleteDelQueue\n");
				return FALSE;
			}
			DeleteCriticalSection(&critical_initDelQueue);
			DeleteCriticalSection(&critical_memory);
			DeleteCriticalSection(&critical_DelQueue_free);
			DeleteCriticalSection(&critical_DelQueue_work);
		DeleteCriticalSection(&(resp_global_pool.form_template_spinl
ock));

DeleteCriticalSection(&(txn_data_pool.form_template_spinlock));

			int i_type, i_pool;
#define GPOOL txn_global_pool[i_type][i_pool]
			for (i_type = 0; i_type < POOL_TYPE_TXN_MAX;
i_type++)
				for (i_pool = 0; i_pool <
TXN_TYPE_MAX; i_pool++)

DeleteCriticalSection(&(GPOOL.form_template_spinlock));
#undef GPOOL
		}

    return TRUE;
}



BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO *pVer)
{
  pVer->dwExtensionVersion = HSE_VERSION;
  strncpy(pVer->lpszExtensionDesc,
    "IIS ISAPI Extension", HSE_MAX_EXT_DLL_NAME_LEN);
  return TRUE;
}



DWORD WINAPI
HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{

		if (!modtpcc_ready)
			return FALSE;

		if (!memory_ready) {
			EnterCriticalSection(&critical_memory);
			if (!memory_ready) {
				allocateMemoryPool();
				memory_ready=1;
			}
			LeaveCriticalSection(&critical_memory);
		}

		if (!queue_ready) {
			EnterCriticalSection(&critical_initDelQueue);
			if (!queue_ready) {
				if (!initDelQueue()) {
					userlog("init Delivery Queue
failed\n");

		LeaveCriticalSection(&critical_initDelQueue);
					return FALSE;
				}
				queue_ready=1;
			}
			LeaveCriticalSection(&critical_initDelQueue);
		}

		return process_query(pECB)==TRUE ?
HSE_STATUS_SUCCESS_AND_KEEP_CONN :
HSE_STATUS_ERROR;
/*


  HSE_SEND_HEADER_EX_INFO info = { 0 };
  char szOut[256];
  DWORD nOut;


  nOut = sprintf(szOut, "%s is the input, LogFile:%s, DllPath:%s,
DllFile:%s, origin:%s, ORACLE_HOME: %s", pECB-
>lpszQueryString,LogFile, DllPath, DllFile, origin,
getenv("ORACLE_HOME"));

  char szHeader[256];
  DWORD nHeader = sprintf(szHeader, "Content-Type: text/html\r\n"
"Contest-Length: %d\r\n\r\n", nOut);

  info.pszStatus = "200 OK";
  info.cchStatus = strlen(info.pszStatus);
  info.pszHeader = szHeader;
  info.cchHeader = nHeader;
  info.fKeepConn = false;

  if (!pECB->ServerSupportFunction(pECB->ConnID,
HSE_REQ_SEND_RESPONSE_HEADER_EX, &info, 0, 0))
		return HSE_STATUS_ERROR;

  if (!pECB->WriteClient(pECB->ConnID, szOut, &nOut,
HSE_IO_SYNC))
		return HSE_STATUS_ERROR;

  return HSE_STATUS_SUCCESS;
*/
}



/*************************************************************
***********************************
* initialize / delete Delivery Queue                          *
*************************************************************
***********************************/

int deleteDelQueue()
{
		DelQueue_info *ptr = DelQueue_begin, *next;

		DeliveryThreadstop = 1;

		for (int i=0; i<DeliveryThreads; i++) {

			if (!SetEvent(waitDelWork)) {
				userlog("Error on
SetEvent(waitDelWork) on deleteDelQueue\n");
			}

			if (WaitForSingleObject(DelThreadRunning,
100000) != WAIT_OBJECT_0) {
				userlog("Delivery Thread is not loaded
after 100 seconds\n");
			}
		}

		if (waitAvailableDelQueue != 0) {
			if (!CloseHandle(waitAvailableDelQueue))
				userlog("error on
CloseHandle(waitAvailableDelQueue)\n");
			waitAvailableDelQueue = 0;
		}

		if (waitDelWork != 0) {
			if (!CloseHandle(waitDelWork))
```

```
				userlog("error on 
CloseHandle(waitDelWork)\n");
			waitDelWork = 0;
		}

		if (DelThreadRunning != 0) {
			if (!CloseHandle(DelThreadRunning))
				userlog("error on 
CloseHandle(DelThreadRunning)\n");
			DelThreadRunning = 0;
		}


		while (ptr != NULL) {
			next=ptr->Next;

#ifdef DELIVERY_MUTEX
			CloseHandle(ptr->queue_lock);
#endif

			free(ptr->pdata);
			free(ptr);
			ptr=next;
		}

		ptr = DelQueue_free;
		while (ptr != NULL) {
			next=ptr->Next;

#ifdef DELIVERY_MUTEX
			CloseHandle(ptr->queue_lock);
#endif

			free(ptr->pdata);
			free(ptr);
			ptr=next;
		}

		return TRUE;
}



int initDelQueue()
{
		int i;
		DelQueue_info *ptr, *curr;

		userlog("execute initDelQueue\n");

		for (i=0; i<DeliveryQueues; i++) {
			if ((ptr = (DelQueue_info *) 
malloc(sizeof(DelQueue_info))) == NULL) {
				userlog("malloc error in 
initDelQueue\n");
				return FALSE;
			}

			ptr->pdata=(T_delivery_data 
*)malloc(sizeof(T_delivery_data));

#ifdef DELIVERY_MUTEX
			if ((ptr->queue_lock=CreateMutex(NULL, FALSE, 
NULL))==NULL) {
				userlog("Cannot create mutex on queue 
lock\n");
				return FALSE;
			}
#endif
			if (!i)
				DelQueue_free=curr=ptr;
			else {
				curr->Next = ptr;
				curr = ptr;
			}
		}

		DelQueue_begin = DelQueue_end = curr->Next = NULL;

		if ((waitAvailableDelQueue = CreateEvent(NULL, FALSE, 
FALSE, "Wait Empty Delivery Queue")) == NULL) {
			userlog("Cannot create event : 
waitAvailableDelQueue\n");
			return FALSE;
		}

		if ((waitDelWork = CreateEvent(NULL, FALSE, FALSE, 
"Wait Delivery Work")) == NULL) {
			userlog("Cannot create event : waitDelWork\n");
			return FALSE;
		}

		if ((DelThreadRunning = CreateEvent(NULL, FALSE, FALSE, 
"Delivery Thread Running")) == NULL) {
			userlog("Cannot create event : 
DelThreadRunning\n");
			return FALSE;
		}

		for (i=0; i < DeliveryThreads; i++) {

			if (_beginthread(initDeliveryThread, 0, (void *) &i) 
== -1) {
				userlog("Error on initDeliveryThread 
%d\n", i);
				return FALSE;
			}

			/* wait for 100 seconds */
			if (WaitForSingleObject(DelThreadRunning, 
100000) != WAIT_OBJECT_0) {
				userlog("Delivery Thread (%d) hasn't 
initialized after 100 seconds\n", i);
				return FALSE;
			}

			userlog("receive Delivery Thread %d 
confirmation\n", i);
		}

		return TRUE;
}



void initDeliveryThread(void *thread_no)
{
		int thread_number=*((int *)thread_no);
		DelQueue_info *queue_info;

		if (!SetEvent(DelThreadRunning))
			userlog("SetEvent Error on 
initDeliveryThread(%d)\n", thread_number);
		else {

			userlog("Delivery Thread %d is created\n", 
thread_number);

			while (!DeliveryThreadstop) {
```

```
                              queue_info = NULL;
                              while (!DeliveryThreadstop &&
queue_info == NULL) {
                                        queue_info=DequeueDel();
                                        if (queue_info == NULL) {
                                                  if
(WaitForSingleObject(waitDelWork, INFINITE) != WAIT_OBJECT_0) {

          userlog("Error on WaitForSingleObject(waitDelQueueWork) in
initDeliveryThread\n");

          endDeliveryThread(thread_number);
                                                            return;
                                                  }
                                        }
                              }

                              if (!DeliveryThreadstop) {

          (void)mod_tpcc_delivery(queue_info->pdata, thread_number);

          addFreeDelQueue(queue_info);
                              }
                    }
          }

          endDeliveryThread(thread_number);
}



void endDeliveryThread(int thread_number)
{
          if (!SetEvent(DelThreadRunning)) {
                    userlog("SetEvent Error on
endDeliveryThread(%d)\n", thread_number);
          }
          _endthread();
}



/************************************************************
**********************************
* Delivery Queue dequeue/enqueue                                *
************************************************************
*********************************/

DelQueue_info *DequeueDel()
{
          DelQueue_info *ptr;

          if (DelQueue_begin == NULL) return NULL;

          EnterCriticalSection(&critical_DelQueue_work);

          if (DelQueue_begin == NULL) {
                    LeaveCriticalSection(&critical_DelQueue_work);
                    return NULL;
          }

          if (DelQueue_begin == DelQueue_end) {
                    ptr = DelQueue_begin;
                    DelQueue_begin = DelQueue_end = NULL;
          }
          else {
                    ptr = DelQueue_begin;
                    DelQueue_begin = DelQueue_begin->Next;
          }
          LeaveCriticalSection(&critical_DelQueue_work);

          return ptr;
}


void EnqueueDel(DelQueue_info *queue_info)
{
          EnterCriticalSection(&critical_DelQueue_work);
          if (DelQueue_begin == NULL)
                    DelQueue_begin=DelQueue_end=queue_info;
          else {
                    DelQueue_end->Next = queue_info;
                    queue_info->Next = NULL;
                    DelQueue_end = queue_info;
          }

          LeaveCriticalSection(&critical_DelQueue_work);
}



void addFreeDelQueue(DelQueue_info *ptr)
{
          EnterCriticalSection(&critical_DelQueue_free);

          if (DelQueue_free==NULL) {
                    DelQueue_free = ptr;
                    ptr->Next = NULL;
          }
          else {
                    ptr->Next = DelQueue_free;
                    DelQueue_free = ptr;
          }
#ifdef DEBUG
          useddel--;
          if (useddel != 0 && useddel % 300 == 0)
                    userlog("free a del queue: current: %d\n", useddel);
#endif
          LeaveCriticalSection(&critical_DelQueue_free);
          if (!SetEvent(waitAvailableDelQueue))
                    userlog("SetEvent Error on addFreeDelQueue\n");
}



DelQueue_info *findFreeDelQueue()
{
          DelQueue_info *ptr=NULL;

          EnterCriticalSection(&critical_DelQueue_free);

          while (ptr==NULL) {
                    if (DelQueue_free==NULL) {

          LeaveCriticalSection(&critical_DelQueue_free);
                              if
(WaitForSingleObject(waitAvailableDelQueue, INFINITE) !=
WAIT_OBJECT_0) {

          userlog("WaitForSingleObject(waitAvailableDelQueue) in
findFreeDelQueue\n");
                              }
                              userlog("Delivery queue is full, sleep for
10 seconds\n");
#ifdef DEBUG
                              userlog("used del queue: %d\n", useddel);
#endif
                              /* sleep for 10 seconds */
```

```
                                        Sleep(10000);

                EnterCriticalSection(&critical_DelQueue_free);
                        }
                        else {
                                ptr = DelQueue_free;
                                DelQueue_free = DelQueue_free->Next;
#ifdef DEBUG
                                useddel++;
                                if (useddel % 300 == 0)
                                        userlog("allocate a del queue
current used: %d\n", useddel);
#endif
                        }
                }

                LeaveCriticalSection(&critical_DelQueue_free);

                return ptr;
}



/************************************************************
**********************************
* process query                                             *
************************************************************
**********************************/

int process_query(EXTENSION_CONTROL_BLOCK *pECB)
{
   int w_id, ld_id, form;
   char *ptr, *cmd;

   form = w_id = ld_id = 0;

   /*
     This process the request_rec http:server/tpcc
   */

   if (strlen(pECB->lpszQueryString) == 0)
                        return sendform_welcome(pECB, "Welcome!");

   if (getcharvalue(pECB->lpszQueryString, '3', &ptr)) {
      form = *ptr++;
      if (get_wid_did(ptr, &w_id, &ld_id, &ptr) == FALSE) {
                                return send_error_message(pECB, 0,
INVALID_TERMID, "", w_id, ld_id, 0);
      }
   } else {
      form = '\0';
   }

   if (getcharvalue(ptr, '0', &cmd) == FALSE)
      return send_error_message(pECB, 0, COMMAND_UNDEFINED, "",
w_id, ld_id, 0);

   if ((form == '\0') && !(CMD_BEGIN(cmd)))
      return send_error_message(pECB, 0,
INVALID_FORM_AND_CMD_NOT_BEGIN, "", w_id, ld_id, 0);

   if (CMD_PROCESS(cmd)) {   /* cmd = Process */

      if (form == 'N') {
                                /* New Order transaction */
         return mod_neworder_query(pECB, w_id, ld_id, ptr);
      } else if (form == 'P') {
                                /* Payment order transaction */
         return mod_payment_query(pECB, w_id, ld_id, ptr);
      } else if (form == 'D') {
                                /* Delivery order transaction */
         return mod_delivery_query(pECB, w_id, ld_id, ptr);
      } else if (form == 'O') {
                                /* Order Status order transaction */
         return mod_orderstatus_query(pECB, w_id, ld_id, ptr);
      } else if (form == 'S') {
                                /* Stock Level order transaction */
         return mod_stocklevel_query(pECB, w_id, ld_id, ptr);
      } else
                                return send_error_message(pECB, 0,
INVALID_FORM, "", w_id, ld_id, 0);
                }
                else if (CMD_BEGIN(cmd))           return
mod_begin_cmd(pECB);
                else if (CMD_NEWORDER(cmd))    return
mod_neworder_cmd(pECB, w_id, ld_id);
                else if (CMD_PAYMENT(cmd))     return
mod_payment_cmd(pECB, w_id, ld_id);
   else if (CMD_DELIVERY(cmd))    return mod_delivery_cmd(pECB,
w_id, ld_id);
   else if (CMD_ORDERSTATUS(cmd)) return
mod_orderstatus_cmd(pECB, w_id, ld_id);
   else if (CMD_STOCKLEVEL(cmd))  return
mod_stocklevel_cmd(pECB, w_id, ld_id);
   else if (CMD_EXIT(cmd))        return mod_exit_cmd(pECB);
   else if (CMD_MENU(cmd))        return mod_menu_cmd(pECB, w_id,
ld_id);
                else
     return send_error_message(pECB, 0, COMMAND_UNDEFINED, "",
w_id, ld_id, 0);

   return TRUE;
}



int getcharvalue(char *iptr, char key, char **optr)
{
   *optr = iptr;

   while (iptr) {
     if ((key == *iptr) && ('=' == *++iptr)) {
        *optr = ++iptr;
        return TRUE;
     }
     while (iptr) {
        if ('&' == *iptr) {
          iptr++; break;
        }
        iptr++;
     }
   }
   return FALSE;
}



void readInit(char *output, char *parameter, char *default_value)
{
                if (_access(InitFile, 0x00) != NULL) {
                        userlog("Cannot access init file: %s\n", InitFile);
                        strcpy(output, default_value);
                }
                else
                  GetPrivateProfileString("TPCC", parameter, default_value,
output, MAXLEN, InitFile);
}


void allocateMemoryPool()
```

```
{
                userlog("Allocate Memory Pool\n");
                allocate_template_pool();
                allocate_response_pool();
                allocate_transaction_pool();
}


void allocate_response_pool()
{
   int i;

   InitializeCriticalSection(&(resp_global_pool.form_template_spinlock));
   resp_global_pool.form_template_length = BUF_SIZE;
   resp_global_pool.form_template_size =
resp_global_pool.form_template_length * Maxterms;
   resp_global_pool.form_template_storage = (char
*)malloc(resp_global_pool.form_template_size);
   resp_global_pool.free_slot = 0;
   resp_global_pool.free_list = (int *)malloc((Maxterms - 1) * sizeof(int));
   for (i = 0; i < (Maxterms - 2); i++) {
       resp_global_pool.free_list[i] = i + 1;
   }
   resp_global_pool.free_list[Maxterms - 2] = -1;
}



void make_txn_form_template(char *input_form, char
*input_form_template,
     char *response_form, char *response_form_template, int txn_type)
{
     int length;
     /*
        For input form.
      */
     length = sprintf(input_form, FormHeader, mod_name);
     length = build_form_index(input_form, input_form_template,
                   form_index[POOL_TYPE_TXN_INPUT][txn_type], length);
     length = (length + 16) & (~((int)7));

txn_global_pool[POOL_TYPE_TXN_INPUT][txn_type].form_template_le
ngth = length+150;

     /*
        For output form.
      */
     length = sprintf(response_form, FormHeader, mod_name);
     length = build_form_index(response_form, response_form_template,
                   form_index[POOL_TYPE_TXN_OUTPUT][txn_type],
length);
     length = (length + 128) & (~((int)7));

txn_global_pool[POOL_TYPE_TXN_OUTPUT][txn_type].form_template
_length = length+250;
     return;
}



int build_form_index(char *form, char *form_template,
                    form_index_entry *f_index, int length)
{
    int current_index = 0;
    int i = 0;
    int j = 0;
    int current_length = length;

    while (form_template[i]) {
       if (form_template[i] != '#') {
          form[current_length] = form_template[i];
          i++; current_length++;
       } else {
          j = 0;
          f_index->index = current_length;
          while (form_template[i] == '#') {
             j++;
             form[current_length] = form_template[i];
             i++; current_length++;
          }
          f_index->length = j;
          f_index++; current_index++;
       }
    }
    form[current_length] = '\0'; current_length++;
    return current_length;
}




void allocate_template_pool()
{
#define FORM_PAD 64
#define GPOOL txn_global_pool[i_type][i_pool]

   char DeliveryInput[sizeof(DeliveryFormInput_Template)+FORM_PAD];
   char
OrderStatusInput[sizeof(OrderStatusInput_Template)+FORM_PAD];
   char PaymentInput[sizeof(PaymentInput_Template)+FORM_PAD];
   char NewOrderInput[sizeof(NewOrderInput_Template)+FORM_PAD];
   char StockLevelInput[sizeof(StockLevelInput_Template)+FORM_PAD];

   char
DeliveryOutput[sizeof(DeliveryFormOutput_Template)+FORM_PAD];
   char
OrderStatusOutput[sizeof(OrderStatusOutput_Template)+FORM_PAD];
   char PaymentOutput[sizeof(PaymentOutput_Template)+FORM_PAD];
   char
NewOrderOutput[sizeof(NewOrderOutput_Template)+FORM_PAD];
   char
StockLevelOutput[sizeof(StockLevelOutput_Template)+FORM_PAD];
   int  i_type, i_pool, i;

   make_txn_form_template(DeliveryInput, DeliveryFormInput_Template,
       DeliveryOutput, DeliveryFormOutput_Template,
TXN_TYPE_DELIVERY);

   make_txn_form_template(OrderStatusInput, OrderStatusInput_Template,
       OrderStatusOutput, OrderStatusOutput_Template,
TXN_TYPE_ORDERSTATUS);

   make_txn_form_template(PaymentInput, PaymentInput_Template,
       PaymentOutput, PaymentOutput_Template,
TXN_TYPE_PAYMENT);

   make_txn_form_template(NewOrderInput, NewOrderInput_Template,
       NewOrderOutput, NewOrderOutput_Template,
TXN_TYPE_NEWORDER);

   make_txn_form_template(StockLevelInput, StockLevelInput_Template,
       StockLevelOutput, StockLevelOutput_Template,
TXN_TYPE_STOCKLEVEL);

   for (i_type = 0; i_type < POOL_TYPE_TXN_MAX; i_type++) {
       for (i_pool = 0; i_pool < TXN_TYPE_MAX; i_pool++) {
          int  i, form_length;

InitializeCriticalSection(&(GPOOL.form_template_spinlock));
```

```
        GPOOL.form_template_size = Maxterms;
        GPOOL.form_template_storage = (char *)malloc(Maxterms *
GPOOL.form_template_length);
        GPOOL.free_list = (int *)malloc((Maxterms - 1)* sizeof(int));

        GPOOL.free_slot = 0;
        form_length = GPOOL.form_template_length;

        for (i = 0; i < (Maxterms - 2); i++) {
          GPOOL.free_list[i] = i+1;
        }
        GPOOL.free_list[Maxterms-2] = -1;
      }
   }

   i_type = POOL_TYPE_TXN_INPUT; i_pool =
TXN_TYPE_DELIVERY;
   strcpy((char *)(GPOOL.form_template_storage),
        DeliveryInput);

   i_type = POOL_TYPE_TXN_OUTPUT; i_pool =
TXN_TYPE_DELIVERY;
   strcpy((char *)(GPOOL.form_template_storage),
        DeliveryOutput);

   i_type = POOL_TYPE_TXN_INPUT; i_pool =
TXN_TYPE_STOCKLEVEL;
   strcpy((char *)(GPOOL.form_template_storage),
        StockLevelInput);

   i_type = POOL_TYPE_TXN_OUTPUT; i_pool =
TXN_TYPE_STOCKLEVEL;
   strcpy((char *)(GPOOL.form_template_storage),
        StockLevelOutput);

   i_type = POOL_TYPE_TXN_INPUT; i_pool =
TXN_TYPE_NEWORDER;
   strcpy((char *)(GPOOL.form_template_storage),
        NewOrderInput);

   i_type = POOL_TYPE_TXN_OUTPUT; i_pool =
TXN_TYPE_NEWORDER;
   strcpy((char *)(GPOOL.form_template_storage),
        NewOrderOutput);

   i_type = POOL_TYPE_TXN_INPUT; i_pool =
TXN_TYPE_ORDERSTATUS;
   strcpy((char *)(GPOOL.form_template_storage),
        OrderStatusInput);

   i_type = POOL_TYPE_TXN_OUTPUT; i_pool =
TXN_TYPE_ORDERSTATUS;
   strcpy((char *)(GPOOL.form_template_storage),
        OrderStatusOutput);

   i_type = POOL_TYPE_TXN_INPUT; i_pool =
TXN_TYPE_PAYMENT;
   strcpy((char *)(GPOOL.form_template_storage),
        PaymentInput);

   i_type = POOL_TYPE_TXN_OUTPUT; i_pool =
TXN_TYPE_PAYMENT;
   strcpy((char *)(GPOOL.form_template_storage),
        PaymentOutput);

   for (i_type = 0; i_type < POOL_TYPE_TXN_MAX; i_type++) {
      for (i_pool = 0; i_pool < TXN_TYPE_MAX; i_pool++) {
         for (i = 1; i < GPOOL.form_template_size; i++) {
            memcpy((char *)(GPOOL.form_template_storage + i *
GPOOL.form_template_length),
                 (char *)(GPOOL.form_template_storage),
                 GPOOL.form_template_length);
         }
      }
   }

#undef FORM_PAD
#undef GPOOL
}




void allocate_transaction_pool()
{
   int i, pool_size;

   pool_size = 0;
   pool_size = MAX(pool_size, sizeof(T_connect_data));
   pool_size = MAX(pool_size, sizeof(T_delivery_data));
   pool_size = MAX(pool_size, sizeof(T_neworder_data));
   pool_size = MAX(pool_size, sizeof(T_stocklevel_data));
   pool_size = MAX(pool_size, sizeof(T_orderstatus_data));
   pool_size = MAX(pool_size, sizeof(T_payment_data));
   pool_size = MAX(pool_size, sizeof(T_login_data));

   InitializeCriticalSection(&(txn_data_pool.form_template_spinlock));
   txn_data_pool.form_template_length = pool_size;
   txn_data_pool.form_template_size =
txn_data_pool.form_template_length * Maxterms;
   txn_data_pool.form_template_storage = (char
*)malloc(txn_data_pool.form_template_size);
   txn_data_pool.free_slot = 0;
   txn_data_pool.free_list = (int *)malloc((Maxterms - 1) * sizeof(int));
   for (i = 0; i < (Maxterms - 2); i++) {
      txn_data_pool.free_list[i] = i + 1;
   }
   txn_data_pool.free_list[Maxterms - 2] = -1;
}



/*
   This processes the form that provides the w_id and d_id of a terminal.
*/
int mod_begin_cmd(EXTENSION_CONTROL_BLOCK *pECB)
{
   char *ptr;
   int  w_id, ld_id;

   if ((getcharvalue(pECB->lpszQueryString, '4', &ptr) == FALSE) || ((w_id
= atoi(ptr)) <= 0))
      return sendform_welcome(pECB, "Error: Invalid Warehouse ID");

   if ((getcharvalue(ptr, '5', &ptr) == FALSE) || ((ld_id = atoi(ptr)) <= 0) ||
(ld_id > 10))
      return sendform_welcome(pECB, "Error: Invalid District DID");

   /*
     Perform activities related to database logon etc.
   */

   return sendform_mainmenu(pECB, w_id, ld_id);
}


int mod_exit_cmd(EXTENSION_CONTROL_BLOCK *pECB)
{
   return sendform_welcome(pECB, "Goodbye!");
}
```

```
int mod_menu_cmd(EXTENSION_CONTROL_BLOCK *pECB, int w_id,
int ld_id)
{
		return sendform_mainmenu(pECB, w_id, ld_id);
}



int get_wid_did(char *ptr, int *wid, int *did, char **optr)
{
    int total = 0;
    int c, pc;
    int provided = FALSE;

    *wid = *did = 0;
    *optr = ptr;
    pc = (int)(unsigned char) *ptr++;
    if ((pc < '0') || (pc > '9'))
       return FALSE;
    c = (int)(unsigned char) *ptr++;
    while ((c >= '0') && (c <= '9')) {
       total = 10 * total + (pc - '0');
       pc = c;
       c = (int)(unsigned char) *ptr++;
       provided = TRUE;
    }
    if (provided) {
       *wid = total;
       *did = (int) (pc - '0') + 1;
       *optr = ptr;
       return TRUE;
    }
    return FALSE;
}



int sendform_welcome(EXTENSION_CONTROL_BLOCK *pECB, char
*mesg)
{
   char *response;
   int index = -1, ret;

   response = allocate_form(&resp_global_pool, &index);
   sprintf(response, WelcomeForm, mod_name, mesg);
   ret=send_response(pECB, response, strlen(response));
   free_form(&resp_global_pool, response, index);
   return ret;
}



int send_response(EXTENSION_CONTROL_BLOCK *pECB, char
*form, int size)
{
   HSE_RESPONSE_VECTOR vec;
   HSE_VECTOR_ELEMENT elem;
   char szHeader[256];
   DWORD nHeader = sprintf(szHeader, "Content-Type: text/html\r\n"
"Content-Length: %d\r\n" "Connection: Keep-Alive\r\n\r\n" , size);

   elem.ElementType =
HSE_VECTOR_ELEMENT_TYPE_MEMORY_BUFFER;
   elem.cbOffset = 0;
   elem.cbSize = size;
   elem.pvContext = form;

   vec.pszHeaders = szHeader;
   vec.lpElementArray = &elem;
   vec.nElementCount = 1;
   vec.pszStatus = "200 OK";
   vec.dwFlags = HSE_IO_SEND_HEADERS;

   if (!pECB->ServerSupportFunction(pECB->ConnID,
HSE_REQ_VECTOR_SEND, &vec, 0, 0))
   {
		userlog("ServerSupportFunction() returns false");
		return FALSE;
   }

   pECB->dwHttpStatusCode = 200;


   return TRUE;

}


char *allocate_form_new(form_template_pool *pool, int index)
{
		int pool_index=index-StartTerm;
		if (pool_index <= Maxterms)
			return (char *)(pool->form_template_storage +
pool_index * pool->form_template_length);
		else
			userlog("allocate_form_new failed max_threads =
%d", Maxterms);
		return (char *)0;
}


char *allocate_form(form_template_pool *pool, int *pool_index)
{
   int current;

   EnterCriticalSection(&(pool->form_template_spinlock));
   current = pool->free_slot;
   if (current >= 0) {
      pool->free_slot = pool->free_list[current];
      LeaveCriticalSection(&(pool->form_template_spinlock));
      *pool_index = current;
      return (char *)(pool->form_template_storage + current * pool-
>form_template_length);
   }
   LeaveCriticalSection(&(pool->form_template_spinlock));
   userlog("allocate_form failed max_threads = %d", Maxterms);
   *pool_index = -1;
   return (char *)0;
}


void free_form(form_template_pool *pool, char *form_template, int
pool_index)
{
   if (! form_template || pool_index < 0 ) return;

   EnterCriticalSection(&(pool->form_template_spinlock));
   pool->free_list[pool_index] = pool->free_slot;
   pool->free_slot = pool_index;
   LeaveCriticalSection(&(pool->form_template_spinlock));
}


int send_error_message(EXTENSION_CONTROL_BLOCK *pECB, int
error_type, int error,
			char *error_msg, int w_id, int ld_id, void *context)
{
   char *response;
   char *mesg = "";
   int index = -1, ret;
   T_error_message *err = error_message;

   while (err->error_code) {
      if (err->error_code == error) {
         mesg = err->error_mesg; break;
      }
      err++;
   }
   response = allocate_form(&resp_global_pool, &index);
   sprintf(response, ErrorForm, mod_name, WDID(w_id, ld_id), error_type,
error, mesg, error_msg);
   ret=send_response(pECB, response, strlen(response));
   free_form(&resp_global_pool, response, index);
   return ret;
}



int sendform_mainmenu(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id)
{
   char *response;
   int index = -1, ret;

   response = allocate_form(&resp_global_pool, &index);
   sprintf(response, MainForm, mod_name, WDID(w_id, ld_id), "");
   ret=send_response(pECB, response, strlen(response));
   free_form(&resp_global_pool, response, index);
   return ret;
}




int sendform_neworderinput(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id)
{
   char *form;
   int index = w_id*10+ld_id, ret;
   form_template_pool *pool;
#define SUBI POOL_TYPE_TXN_INPUT][TXN_TYPE_NEWORDER

   pool = &txn_global_pool[SUBI];

#ifdef NEW_ALLOCATE_FORM
   form  = allocate_form_new(pool, index);
#else
   form  = allocate_form(pool, &index);
#endif

   fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][NO_TERMID].index,
            form_index[SUBI][NO_TERMID].length);
   fill_number(form, w_id, form_index[SUBI][NO_WID].index,
            form_index[SUBI][NO_WID].length);
   ret=send_response(pECB, form, strlen(form));

#ifndef NEW_ALLOCATE_FORM
              free_form(pool, form, index);
#endif

   return ret;
#undef SUBI
}
```

```
int sendform_deliveryinput(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id)
{
   char *form;
   int index = w_id*10+ld_id, ret;
   form_template_pool *pool;
#define SUBI POOL_TYPE_TXN_INPUT][TXN_TYPE_DELIVERY

   pool = &txn_global_pool[SUBI];

#ifdef NEW_ALLOCATE_FORM
   form  = allocate_form_new(pool, index);
#else
   form  = allocate_form(pool, &index);
#endif

   fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][DE_TERMID].index,
            form_index[SUBI][DE_TERMID].length);
   fill_number(form, w_id, form_index[SUBI][DE_WID].index,
            form_index[SUBI][DE_WID].length);
   ret=send_response(pECB, form, strlen(form));

#ifndef NEW_ALLOCATE_FORM
   free_form(pool, form, index);
#endif

   return ret;
#undef SUBI
}



int sendform_stocklevelinput(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id)
{
   char *form;
   int index = w_id*10+ld_id, ret;
   form_template_pool *pool;
#define SUBI POOL_TYPE_TXN_INPUT][TXN_TYPE_STOCKLEVEL

   pool = &txn_global_pool[SUBI];

#ifdef NEW_ALLOCATE_FORM
   form  = allocate_form_new(pool, index);
#else
   form  = allocate_form(pool, &index);
#endif

   fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][SL_TERMID].index,
            form_index[SUBI][SL_TERMID].length);
   fill_number(form, w_id, form_index[SUBI][SL_WID].index,
            form_index[SUBI][SL_WID].length);
   fill_number(form, ld_id, form_index[SUBI][SL_DID].index,
            form_index[SUBI][SL_DID].length);
   ret=send_response(pECB, form, strlen(form));

#ifndef NEW_ALLOCATE_FORM
   free_form(pool, form, index);
#endif

   return ret;
#undef SUBI
}



int sendform_paymentinput(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id)
```

```
{
   char *form;
   int index = w_id*10+ld_id, ret;
   form_template_pool *pool;
#define SUBI POOL_TYPE_TXN_INPUT][TXN_TYPE_PAYMENT

   pool = &txn_global_pool[SUBI];

#ifdef NEW_ALLOCATE_FORM
   form  = allocate_form_new(pool, index);
#else
   form  = allocate_form(pool, &index);
#endif

   fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][PA_INPUT_TERMID].index,
            form_index[SUBI][PA_INPUT_TERMID].length);

   fill_number(form, w_id, form_index[SUBI][PA_INPUT_WID].index,
            form_index[SUBI][PA_INPUT_WID].length);

   ret=send_response(pECB, form, strlen(form));

#ifndef NEW_ALLOCATE_FORM
   free_form(pool, form, index);
#endif

   return ret;
#undef SUBI
}

int sendform_orderstatusinput(EXTENSION_CONTROL_BLOCK
*pECB, int w_id, int ld_id)
{
   char *form;
   int index = w_id*10+ld_id, ret;
   form_template_pool *pool;
#define SUBI
POOL_TYPE_TXN_INPUT][TXN_TYPE_ORDERSTATUS

   pool = &txn_global_pool[SUBI];

#ifdef NEW_ALLOCATE_FORM
   form  = allocate_form_new(pool, index);
#else
   form  = allocate_form(pool, &index);
#endif

   fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][OS_TERMID].index,
            form_index[SUBI][OS_TERMID].length);
   fill_number(form, w_id, form_index[SUBI][OS_WID].index,
            form_index[SUBI][OS_WID].length);
   ret=send_response(pECB, form, strlen(form));

#ifndef NEW_ALLOCATE_FORM
   free_form(pool, form, index);
#endif

   return ret;
#undef SUBI
}



void fill_string(char *form, char *string, int index, int length, int *shift)
{
  char *ptr;
  int i;

  for (i=0, ptr=string; i<length && (*ptr)!='\0'; i++, ptr++) {
    form[index+i]=(char)(*ptr);
    switch (*ptr) {
      case '\"' : (*shift)+=5;
              break;
      case '&'  : (*shift)+=4;
              break;
      case '>'  : (*shift)+=3;
              break;
      case '<'  : (*shift)+=3;
              break;
    }
  }

  for (; i<length; i++)
    form[index+i]=' ';
}

void adjust_form(char *form, int *indexes, int *length, int size, int formlen,
int totalshift)
{
  int ptr, ptr2, ind;

  for (ptr=formlen, ptr2=formlen+totalshift, ind=size-1; ptr>=0; ptr--) {
    if (ind>=0 && ptr<indexes[ind])
      ind--;
    if (ind<0 || ptr>=indexes[ind]+length[ind])
      form[ptr2--]=form[ptr];
    else if (ptr>=indexes[ind] && ptr<indexes[ind]+length[ind])
      switch (form[ptr]) {
        case '\"' : form[ptr2--]=';'; form[ptr2--]='t'; form[ptr2--]='o';
               form[ptr2--]='u'; form[ptr2--]='q'; form[ptr2--]='&';
               break;
        case '&'  : form[ptr2--]=';'; form[ptr2--]='p'; form[ptr2--]='m';
               form[ptr2--]='a'; form[ptr2--]='&';
               break;
        case '>'  : form[ptr2--]=';'; form[ptr2--]='t';
               form[ptr2--]='l'; form[ptr2--]='&';
               break;
        case '<'  : form[ptr2--]=';'; form[ptr2--]='t';
               form[ptr2--]='g'; form[ptr2--]='&';
               break;
        default : form[ptr2--]=form[ptr];
              break;
      }
  }
}

void fill_double(char *form, double value, int index, int length)
{
  int ptr = index + length - 1, DecPtr = ptr - 2;
  int avalue=abs((int)(value*100.0));
  int is_neg=(value<0.0);
  char asterick[] = "*******************";

  if (avalue==0)
    form[ptr--]='0';

  while ((avalue!=0 && ptr>=index) || ptr > DecPtr) {
    form[ptr--]='0' + avalue % 10;
    avalue/=10;
    if (ptr == DecPtr)
      form[ptr--]='.';
  }

  if (ptr < index && (is_neg || avalue!=0 ))
    memcpy(form+index, asterick, length);
  else {
    if (is_neg)
      form[ptr--]='-';
    while (ptr>=index)
      form[ptr--]=' ';
  }
}

void fill_number(char *form, int value, int index, int length)
{
   char *pstart = (char *)form + index;
   char *pend = pstart + length - 1;
   char asterick[] = "******************";
   int  slen = length;
   int  is_neg, avalue;

   is_neg = (value < 0);
   avalue = abs(value);

   do {
     *pend = (avalue % 10) + '0';
     avalue = avalue / 10;
     if (--length) pend--;
   } while (length);
/*
   if (avalue==0 && length >0) {
     do {
       *pend=' ';
       if (--length) pend--;
     } while (length);
   }
*/
   if (avalue) {
      memcpy(pstart, asterick, slen);
      return;
   }

   if (is_neg) {
     if (*pend == '0') {
        *pend = '-';
     } else {
        memcpy(pstart, asterick, slen);
        return;
     }
   }
}

int parse_query_string(char *iptr, int max_cnt,
                       char *txn_chars, value_index_entry *txn_vals)
{
   char *ptr = iptr;
   int  key, i;

   for (i = 0; i < max_cnt; i++) {
       key = txn_chars[i];
       txn_vals[i].value = NULL;
       txn_vals[i].length = 0;
       if ((key == *ptr) && ('=' == *++ptr)) {
          txn_vals[i].value = ++ptr;
       }
       while (ptr && ptr[0]!='\0') {
          if ('&' == *ptr) {
             ptr++; break;
          }
          ptr++; txn_vals[i].length++;
       }
   }

   return TRUE;
}

int get_number(char *ptr, int *value)
{
   int c, total;
   int has_value = FALSE;
   int is_neg = FALSE;

   if (*ptr == '-') {
      is_neg = TRUE; ptr++;
   }
   c = (int) (unsigned char) *ptr++;

   total = 0;
   while (( c >= '0') && (c <= '9')) {
      total = 10 * total + (c - '0');
      c = (int) (unsigned char) *ptr++;
      has_value = TRUE;
   }
   if ((c == '\0') || (('&' == c) && has_value)) {
      *value = is_neg?(0-total):total;
      return TRUE;
   }
   *value = 0;
   return FALSE;
}


/*************************************************************
**********************************
* mod transaction output                                      *
*************************************************************
**********************************/

int mod_neworder_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr)
{
   T_neworder_data *pdata;
   int index = w_id*10+ld_id, ret;
   int status = SUCCESS;

#ifdef NEW_ALLOCATE_FORM
   pdata = (T_neworder_data *)allocate_form_new(&txn_data_pool, index);
#else
   pdata = (T_neworder_data *)allocate_form(&txn_data_pool, &index);
#endif

   pdata->w_id = w_id; pdata->ld_id = ld_id; pdata->context = (void
*)pECB;

   status = parse_neworder_query(ptr, pdata);
   if (status != SUCCESS) {
     ret=send_error_message(pECB, 0, status, "", w_id, ld_id, 0);

#ifndef NEW_ALLOCATE_FORM
     free_form(&txn_data_pool, (char *) pdata, index);
#endif

     return ret;
   }

   status = mod_tpcc_neworder(pdata);
   ret=sendform_neworderoutput(status, pdata);

#ifndef NEW_ALLOCATE_FORM
   free_form(&txn_data_pool, (char *) pdata, index);
#endif

   return ret;
}
```

```
int mod_delivery_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr)
{
  DelQueue_info *queue_info;
  int index=-1, ret;
  int status = SUCCESS;

  queue_info = findFreeDelQueue();
  queue_info->pdata->w_id = w_id;
  queue_info->pdata->ld_id = ld_id;
  queue_info->pdata->context = (void *)pECB;

  status = parse_delivery_query(ptr, queue_info->pdata);
  if (status != SUCCESS) {
    ret=send_error_message(pECB, 0, status, "", w_id, ld_id, 0);
    return ret;
  }

  EnqueueDel(queue_info);
  if (!SetEvent(waitDelWork)) {
                userlog("Error on SetEvent(waitDelWork)\n");
                ret=sendform_deliveryoutput(status, queue_info->pdata);
                ret=FALSE;
  }
  else ret=sendform_deliveryoutput(status, queue_info->pdata);
  return ret;
}



int mod_payment_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr)
{
  T_payment_data *pdata;
  int index = w_id*10+ld_id, ret;
  int status = SUCCESS;

#ifdef NEW_ALLOCATE_FORM
  pdata = (T_payment_data *)allocate_form_new(&txn_data_pool, index);
#else
  pdata = (T_payment_data *)allocate_form(&txn_data_pool, &index);
#endif

  pdata->w_id = w_id; pdata->ld_id = ld_id; pdata->context = (void
*)pECB;

  status = parse_payment_query(ptr, pdata);
  if (status != SUCCESS) {
    ret=send_error_message(pECB, 0, status, "", w_id, ld_id, 0);

#ifndef NEW_ALLOCATE_FORM
    free_form(&txn_data_pool, (char *) pdata, index);
#endif

    return ret;
  }

  status = mod_tpcc_payment(pdata);
  ret=sendform_paymentoutput(status, pdata);

#ifndef NEW_ALLOCATE_FORM
  free_form(&txn_data_pool, (char *) pdata, index);
#endif

  return ret;
}


int mod_orderstatus_query(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id, char *ptr)
{
  T_orderstatus_data *pdata;
  int index = w_id*10+ld_id, ret;
  int status = SUCCESS;

#ifdef NEW_ALLOCATE_FORM
  pdata = (T_orderstatus_data *)allocate_form_new(&txn_data_pool,
index);
#else
  pdata = (T_orderstatus_data *)allocate_form(&txn_data_pool, &index);
#endif

  pdata->w_id = w_id; pdata->ld_id = ld_id; pdata->context = (void
*)pECB;

  status = parse_orderstatus_query(ptr, pdata);
  if (status != SUCCESS) {
    ret=send_error_message(pECB, 0, status, "", w_id, ld_id, 0);

#ifndef NEW_ALLOCATE_FORM
    free_form(&txn_data_pool, (char *) pdata, index);
#endif

    return ret;
  }

  status = mod_tpcc_orderstatus(pdata);
  ret=sendform_orderstatusoutput(status, pdata);

#ifndef NEW_ALLOCATE_FORM
  free_form(&txn_data_pool, (char *) pdata, index);
#endif

  return ret;
}


int mod_stocklevel_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr)
{
  T_stocklevel_data *pdata;
  int index = w_id*10+ld_id, ret;
  int status = SUCCESS;

#ifdef NEW_ALLOCATE_FORM
  pdata = (T_stocklevel_data *)allocate_form_new(&txn_data_pool,
index);
#else
  pdata = (T_stocklevel_data *)allocate_form(&txn_data_pool, &index);
#endif

  pdata->w_id = w_id; pdata->ld_id = ld_id; pdata->context = (void
*)pECB;

  status =  parse_stocklevel_query(ptr, pdata);
  if (status != SUCCESS) {
    ret=send_error_message(pECB, 0, status, "", w_id, ld_id, 0);

#ifndef NEW_ALLOCATE_FORM
    free_form(&txn_data_pool, (char *) pdata, index);
#endif

    return ret;
  }

  status = mod_tpcc_stocklevel(pdata);
  ret=sendform_stockleveloutput(status, pdata);

#ifndef NEW_ALLOCATE_FORM
  free_form(&txn_data_pool, (char *) pdata, index);
```

```
#endif

   return ret;
}


/***************************************************************
**********************************
* parse transaction query                                      *
***************************************************************
**********************************/

int parse_neworder_query(char *iptr, T_neworder_data *pdata)
{
   int status, i, items;
   value_index_entry value_ptr[NO_INPUT_MAX];
   char *ptr;

   status = parse_query_string(iptr, NO_INPUT_MAX, neworder_chars,
value_ptr);

   if ((ptr = value_ptr[NO_INPUT_DID].value) == NULL) {
      return NEWORDER_MISSING_DID;
   }
   if ((status = get_number(ptr, &pdata->d_id)) == FALSE) {
      return NEWORDER_DISTRICT_INVALID;
   }
/*
   if ((pdata->d_id > 10) || (pdata->d_id < 1)) {
*/
   if ((pdata->d_id < 1)) {
      return NEWORDER_DISTRICT_RANGE;
   }

   if ((ptr = value_ptr[NO_INPUT_CID].value) == NULL) {
      return NEWORDER_CUSTOMER_KEY;
   }
   if ((status = get_number(ptr, &pdata->c_id)) == FALSE) {
      return NEWORDER_CUSTOMER_INVALID;
   }
/*
   if ((pdata->c_id > 3000) || (pdata->c_id <= 0)) {
*/
   if ((pdata->c_id <= 0)) {
      return NEWORDER_CUSTOMER_RANGE;
   }

   pdata->o_all_local = 1;

   for (i = 0, items = 0; i < 15; i++) {
      if ((ptr = value_ptr[i*3 + NO_INPUT_IID00].value) == NULL) {
         return NEWORDER_MISSING_IID_KEY;
      }
      if (value_ptr[i*3 + NO_INPUT_IID00].length > 0) {
         if ((status = get_number(ptr, &pdata->o_orderline[items].ol_i_id))
== FALSE) {
            return NEWORDER_ITEMID_INVALID;
         }
         if ((ptr = value_ptr[i*3 + NO_INPUT_SPW00].value) == NULL) {
            return NEWORDER_MISSING_SUPPW_KEY;
         }
         if ((status = get_number(ptr, &pdata-
>o_orderline[items].ol_supply_w_id)) == FALSE) {
            return NEWORDER_SUPPW_INVALID;
         }
         if ((ptr = value_ptr[i*3 + NO_INPUT_QTY00].value) == NULL) {
            return NEWORDER_MISSING_QTY_KEY;
         }
         if ((status = get_number(ptr, &pdata-
>o_orderline[items].ol_quantity)) == FALSE) {
            return NEWORDER_QTY_INVALID;
         }
         /*
           We use item number 111111 as the bad one.
         */
/*
         if ((pdata->o_orderline[items].ol_i_id > 999999) ||
            (pdata->o_orderline[items].ol_i_id < 1)) {
*/
                              if ((pdata->o_orderline[items].ol_i_id < 1)) {
            return NEWORDER_ITEMID_RANGE;
         }
/*
         if ((pdata->o_orderline[items].ol_quantity >= 100) ||
            (pdata->o_orderline[items].ol_quantity < 1)) {
*/
         if ((pdata->o_orderline[items].ol_quantity < 1)) {
           return NEWORDER_QTY_RANGE;
         }
         if (pdata->o_all_local && pdata-
>o_orderline[items].ol_supply_w_id != pdata->w_id) {
            pdata->o_all_local = 0;
         }
         items++;
      } else {
         if (value_ptr[i*3 + NO_INPUT_SPW00].value == NULL) {
            return NEWORDER_MISSING_SUPPW_KEY;
         }
         if (value_ptr[i*3 + NO_INPUT_SPW00].length > 0) {
            return NEWORDER_SUPPW_WITHOUT_ITEMID;
         }
         if (value_ptr[i*3 + NO_INPUT_QTY00].value == NULL) {
            return NEWORDER_MISSING_QTY_KEY;
         }
         if (value_ptr[i*3 + NO_INPUT_QTY00].length > 0) {
            return NEWORDER_QTY_WITHOUT_ITEMID;
         }
      }
   }
   if (items ==  0) {
      return NEWORDER_NOITEMS_ENTERED;
   }
   pdata->o_ol_cnt = items;
   return SUCCESS;
}

int parse_payment_query(char *iptr, T_payment_data *pdata)
{
   int status, see_dot, i;
   value_index_entry value_ptr[PA_INPUT_MAX];
   char *ptr;

   status = parse_query_string(iptr, PA_INPUT_MAX, payment_chars,
value_ptr);

   if ((ptr = value_ptr[PA_INPUT_DID].value) == NULL) {
      return PAYMENT_MISSING_DID_KEY;
   }
   if ((status = get_number(ptr, &pdata->d_id)) == FALSE) {
      return PAYMENT_DISTRICT_INVALID;
   }
/*
   if ((pdata->d_id > 10) || (pdata->d_id < 1)) {
*/
   if ((pdata->d_id < 1)) {
      return PAYMENT_DISTRICT_RANGE;
   }

   if ((ptr = value_ptr[PA_INPUT_CID].value) == NULL) {
      return PAYMENT_MISSING_CID_KEY;
```

```
    }

    if (value_ptr[PA_INPUT_CID].length == 0) {          /* c_id == 0 */
      pdata->c_id = 0;
      pdata->by_last_name = 1;
      if ((ptr = value_ptr[PA_INPUT_NAME].value) == NULL) {
        return PAYMENT_MISSING_CLASTNAME_KEY;
      }
      if (value_ptr[PA_INPUT_NAME].length == 0) {
        return PAYMENT_MISSING_CLASTNAME;
      }
      memcpy(pdata->c_last, ptr, value_ptr[PA_INPUT_NAME].length);
      pdata->c_last[value_ptr[PA_INPUT_NAME].length] = '\0';
      STRING_UPPERCASE(pdata->c_last);
      if (value_ptr[PA_INPUT_NAME].length > 16) {
        return PAYMENT_LAST_NAME_TO_LONG;
      }
    } else {                                    /* c_id != 0 */
      pdata->by_last_name = 0;
      if ((status = get_number(ptr, &pdata->c_id)) == FALSE) {
        return PAYMENT_CUSTOMER_INVALID;
      }
/*
      if ((pdata->c_id > 3000) || (pdata->c_id <= 0)) {
*/
                  if ((pdata->c_id <= 0)) {
        return PAYMENT_CID_RANGE;
      }
      if ((ptr = value_ptr[PA_INPUT_NAME].value) == NULL) {
        return PAYMENT_MISSING_CLASTNAME_KEY;
      }
      if (value_ptr[PA_INPUT_NAME].length > 0) {
        return PAYMENT_CID_AND_CLASTNAME;
      }
    }

    if ((ptr = value_ptr[PA_INPUT_CDID].value) == NULL) {
      return PAYMENT_MISSING_CDI_KEY;
    }
    if ((status = get_number(ptr, &pdata->c_d_id)) == FALSE) {
      return PAYMENT_CDI_INVALID;
    }
/*
    if ((pdata->c_d_id > 10) || (pdata->c_d_id < 1)) {
*/
    if ((pdata->c_d_id < 1)) {
      return PAYMENT_CDI_RANGE;
    }
    if ((ptr = value_ptr[PA_INPUT_CWID].value) == NULL) {
      return PAYMENT_MISSING_CWI_KEY;
    }
    if ((status = get_number(ptr, &pdata->c_w_id)) == FALSE) {
      return PAYMENT_CWI_INVALID;
    }
    if ((ptr = value_ptr[PA_INPUT_AMT].value) == NULL) {
      return PAYMENT_MISSING_HAM_KEY;
    }

    see_dot = FALSE;

    for (i = 0; i < value_ptr[PA_INPUT_AMT].length; i++) {
      if (ptr[i] == '\0') {
        return PAYMENT_HAM_INVALID;
      }
      if (ptr[i] == '.') {
         if (see_dot) {
           return PAYMENT_HAM_INVALID;
         } else {
          see_dot = TRUE;
         }
      } else {
        if ((ptr[i] > '9') || (ptr[i] < '0')) {
          return PAYMENT_HAM_INVALID;
        }
      }
    }
    pdata->h_amount = atof(ptr);

    if ((pdata->h_amount < 0) || (pdata->h_amount >= 10000.0)) {
      return PAYMENT_HAM_RANGE;
    }
    return SUCCESS;
  }

int parse_delivery_query(char *iptr, T_delivery_data *pdata)
{
  int status = SUCCESS;
  value_index_entry value_ptr[DE_INPUT_MAX];
  int i, see_dot;
  char *ptr;

  status = parse_query_string(iptr, DE_INPUT_MAX, delivery_chars,
value_ptr);

  if ((ptr = value_ptr[DE_INPUT_DID].value) == NULL) {
    return DELIVERY_MISSING_OCD_KEY;
  }
  if ((status = get_number(ptr, &pdata->o_carrier_id)) == FALSE) {
    return DELIVERY_CARRIER_INVALID;
  }
/*
  if ((pdata->o_carrier_id > 10) || (pdata->o_carrier_id < 1)) {
*/
  if ((pdata->o_carrier_id < 1)) {
    return DELIVERY_CARRIER_ID_RANGE;
  }

  if ((ptr = value_ptr[DE_INPUT_QTIME].value) == NULL) {
              GetLocalTime(&(pdata->enqueue_date_time));
    pdata->enqueue_time = GetTickCount();
    return SUCCESS;
  }

  if (value_ptr[DE_INPUT_QTIME].length == 0) {
    return DELIVERY_MISSING_QUEUETIME_KEY;
  }

  see_dot = FALSE;

  for (i = 0; i < value_ptr[DE_INPUT_QTIME].length; i++) {
    if (ptr[i] == '\0') {
      return DELIVERY_MISSING_QUEUETIME_KEY;
    }
    if (ptr[i] == '.') {
       if (see_dot) {
         return DELIVERY_MISSING_QUEUETIME_KEY;
       } else {
        see_dot = TRUE;
       }
    } else {
      if ((ptr[i] > '9') || (ptr[i] < '0')) {
        return DELIVERY_MISSING_QUEUETIME_KEY;
      }
    }
  }

  pdata->enqueue_time = atof(ptr);

  return SUCCESS;
}
```

```
int parse_orderstatus_query(char *iptr, T_orderstatus_data *pdata)
{
  int status = SUCCESS;
  value_index_entry value_ptr[OS_INPUT_MAX];
  char *ptr;

  status = parse_query_string(iptr, OS_INPUT_MAX, orderstatus_chars,
value_ptr);

  if ((ptr = value_ptr[OS_INPUT_DID].value) == NULL) {
     return ORDERSTATUS_MISSING_DID_KEY;
  }
  if ((status = get_number(ptr, &pdata->d_id)) == FALSE) {
     return ORDERSTATUS_DID_INVALID;
  }
/*
  if ((pdata->d_id > 10) || (pdata->d_id < 1)) {
*/
  if ((pdata->d_id < 1)) {
     return ORDERSTATUS_DID_RANGE;
  }

  if ((ptr = value_ptr[OS_INPUT_CID].value) == NULL) {
     return ORDERSTATUS_MISSING_CID_KEY;
  }

  if (value_ptr[OS_INPUT_CID].length == 0) {
    pdata->c_id = 0;
    pdata->by_last_name = 1;
    if ((ptr = value_ptr[OS_INPUT_NAME].value) == NULL) {
       return ORDERSTATUS_MISSING_CLASTNAME_KEY;
    }
    memcpy(pdata->c_last, ptr, value_ptr[OS_INPUT_NAME].length);
    pdata->c_last[value_ptr[OS_INPUT_NAME].length] = '\0';
    STRING_UPPERCASE(pdata->c_last);
    if (value_ptr[OS_INPUT_NAME].length > 16) {
       return ORDERSTATUS_CLASTNAME_RANGE;
    }
  } else {                              /* c_id != 0 */
    pdata->by_last_name = 0;
    if ((status = get_number(ptr, &pdata->c_id)) == FALSE) {
       return ORDERSTATUS_CID_INVALID;
    }
/*
    if ((pdata->c_id > 3000) || (pdata->c_id <= 0)) {
*/
              if ((pdata->c_id <= 0)) {
       return ORDERSTATUS_CID_RANGE;
    }
    if ((ptr = value_ptr[OS_INPUT_NAME].value) == NULL) {
       return ORDERSTATUS_MISSING_CLASTNAME_KEY;
    }
    if (value_ptr[OS_INPUT_NAME].length > 0) {
       return ORDERSTATUS_CID_AND_CLASTNAME;
    }
  }
  return SUCCESS;
}

int parse_stocklevel_query(char *iptr, T_stocklevel_data *pdata)
{

  value_index_entry value_ptr[SL_INPUT_MAX];
  char *ptr;
  int status = SUCCESS;

  status = parse_query_string(iptr, SL_INPUT_MAX, stocklevel_chars,
value_ptr);

  if ((ptr = value_ptr[SL_INPUT_THRESHOLD].value) == NULL) {
     return STOCKLEVEL_MISSING_THRESHOLD_KEY;
  }
  if ((status = get_number(ptr, &pdata->threshold)) == FALSE) {
     return STOCKLEVEL_THRESHOLD_INVALID;
  }
/*
  if ((pdata->threshold >= 100) || (pdata->threshold < 0)) {
*/
  if ((pdata->threshold < 0)) {
     return STOCKLEVEL_THRESHOLD_RANGE;
  }

  return SUCCESS;
}


/************************************************************
*********************************
* sendform output                                          *
************************************************************
*********************************/

int sendform_neworderoutput(int status, T_neworder_data *pdata)
{
   EXTENSION_CONTROL_BLOCK *pECB;
   int w_id, ld_id, ret;
   char *form, *form2;
   char blank[] = "                                        ";
   int index = -1, formlen, strcount=0, shift=0, i, j, lineStart=15;
   int indexes[NO_FORMINDEX_SIZE],
indLen[NO_FORMINDEX_SIZE], index2=-1;
   form_template_pool *pool;

#define SUBI POOL_TYPE_TXN_OUTPUT][TXN_TYPE_NEWORDER

   w_id = pdata->w_id; ld_id = pdata->ld_id;
             pECB = (EXTENSION_CONTROL_BLOCK *) pdata-
>context;

   if (status != SUCCESS && status != DB_SUCCESS) {
     return send_error_message(pECB, 0, status, "", w_id, ld_id, 0);
   }

   if (pdata->txn_status != DB_RETURN_OCI_SUCCESS) {
                        return send_error_message(pECB, 0, pdata-
>txn_status, "   ---   DATABASE ERROR ", w_id, ld_id, 0);
   }

   pool = &txn_global_pool[SUBI];
             index=w_id*10+ld_id;

#ifdef NEW_ALLOCATE_FORM
   form  = allocate_form_new(pool, index);
#else
   form  = allocate_form(pool, &index);
#endif

   formlen=strlen(form);

   fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][NO_TERMID].index,
           form_index[SUBI][NO_TERMID].length);
   fill_number(form, w_id, form_index[SUBI][NO_WID].index,
           form_index[SUBI][NO_WID].length);

   fill_number(form, pdata->d_id, form_index[SUBI][NO_DID].index,
            form_index[SUBI][NO_DID].length);

   if (!pdata->status) {
     fill_string(form, pdata->o_entry_d.DateString,
form_index[SUBI][NO_DATE].index,
               form_index[SUBI][NO_DATE].length, &shift);
     indexes[strcount]=form_index[SUBI][NO_DATE].index;
     indLen[strcount++]=form_index[SUBI][NO_DATE].length;
   } else {
     memcpy(form+form_index[SUBI][NO_DATE].index, blank,
          form_index[SUBI][NO_DATE].length);
   }

   fill_number(form, pdata->c_id, form_index[SUBI][NO_CID].index,
            form_index[SUBI][NO_CID].length);

   fill_string(form, pdata->c_last, form_index[SUBI][NO_NAME].index,
            form_index[SUBI][NO_NAME].length, &shift);
   indexes[strcount]=form_index[SUBI][NO_NAME].index;
   indLen[strcount++]=form_index[SUBI][NO_NAME].length;

   fill_string(form, pdata->c_credit,
form_index[SUBI][NO_CREDIT].index,
            form_index[SUBI][NO_CREDIT].length, &shift);
   indexes[strcount]=form_index[SUBI][NO_CREDIT].index;
   indLen[strcount++]=form_index[SUBI][NO_CREDIT].length;


   fill_double(form, pdata->c_discount,
form_index[SUBI][NO_DISC].index,
           form_index[SUBI][NO_DISC].length);

   fill_number(form, pdata->o_id, form_index[SUBI][NO_OID].index,
            form_index[SUBI][NO_OID].length);

   fill_number(form, pdata->o_ol_cnt,
form_index[SUBI][NO_LINES].index,
            form_index[SUBI][NO_LINES].length);

   fill_double(form, pdata->w_tax, form_index[SUBI][NO_WTAX].index,
           form_index[SUBI][NO_WTAX].length);

   fill_double(form, pdata->d_tax, form_index[SUBI][NO_DTAX].index,
           form_index[SUBI][NO_DTAX].length);

   if (!pdata->status) {

     for (i=0; i<pdata->o_ol_cnt; i++) {
       fill_number(form, pdata->o_orderline[i].ol_supply_w_id,
               form_index[SUBI][NO_SUPPW+i*8].index,
               form_index[SUBI][NO_SUPPW+i*8].length);

       fill_number(form, pdata->o_orderline[i].ol_i_id,
               form_index[SUBI][NO_ITEMID+i*8].index,
               form_index[SUBI][NO_ITEMID+i*8].length);

       fill_string(form, pdata->o_orderline[i].i_name,
               form_index[SUBI][NO_INAME+i*8].index,
               form_index[SUBI][NO_INAME+i*8].length, &shift);
       indexes[strcount]=form_index[SUBI][NO_INAME+i*8].index;
       indLen[strcount++]=form_index[SUBI][NO_INAME+i*8].length;

       fill_number(form, pdata->o_orderline[i].ol_quantity,
               form_index[SUBI][NO_QTY+i*8].index,
               form_index[SUBI][NO_QTY+i*8].length);

       fill_number(form, pdata->o_orderline[i].s_quantity,
               form_index[SUBI][NO_STOCK+i*8].index,
               form_index[SUBI][NO_STOCK+i*8].length);

       fill_string(form, pdata->o_orderline[i].b_g,
               form_index[SUBI][NO_BRAND+i*8].index,
               form_index[SUBI][NO_BRAND+i*8].length, &shift);
       indexes[strcount]=form_index[SUBI][NO_BRAND+i*8].index;
       indLen[strcount++]=form_index[SUBI][NO_BRAND+i*8].length;

       fill_double(form, pdata->o_orderline[i].i_price,
               form_index[SUBI][NO_PRICE+i*8].index,
               form_index[SUBI][NO_PRICE+i*8].length);

       fill_double(form, pdata->o_orderline[i].ol_amount,
               form_index[SUBI][NO_AMOUNT+i*8].index,
               form_index[SUBI][NO_AMOUNT+i*8].length);
     }

     for (j=NO_SUPPW+i*8; j<NO_SUPPW+15*8; j++)

memcpy(form+form_index[SUBI][j].index,blank,form_index[SUBI][j].len
gth);

     for (lineStart=j=i; j<15; j++) {
       form[form_index[SUBI][NO_PRICE+j*8].index-1]=' ';
       form[form_index[SUBI][NO_AMOUNT+j*8].index-1]=' ';
     }

   } else {
/*
     for (j=NO_DISC; j<=NO_DTAX; j++)

memcpy(form+form_index[SUBI][j].index,blank,form_index[SUBI][j].len
gth);
*/

     for (j=NO_SUPPW; j<NO_SUPPW+15*8; j++)

memcpy(form+form_index[SUBI][j].index,blank,form_index[SUBI][j].len
gth);

     for (lineStart=j=0; j<15; j++) {
       form[form_index[SUBI][NO_PRICE+j*8].index-1]=' ';
       form[form_index[SUBI][NO_AMOUNT+j*8].index-1]=' ';
     }
   }

   if (!pdata->status) {
     fill_string(form, "Transaction committed",
              form_index[SUBI][NO_STATUS].index,
              form_index[SUBI][NO_STATUS].length, &shift);
     indexes[strcount]=form_index[SUBI][NO_STATUS].index;
     indLen[strcount++]=form_index[SUBI][NO_STATUS].length;

     fill_double(form, pdata->total_amount,
form_index[SUBI][NO_TOTAL].index,
             form_index[SUBI][NO_TOTAL].length);
   } else {
     fill_string(form, "Item number is not valid",
              form_index[SUBI][NO_STATUS].index,
              form_index[SUBI][NO_STATUS].length, &shift);
     indexes[strcount]=form_index[SUBI][NO_STATUS].index;
     indLen[strcount++]=form_index[SUBI][NO_STATUS].length;

     memcpy(form+form_index[SUBI][NO_TOTAL].index-1, blank,
          form_index[SUBI][NO_TOTAL].length+1);
   }

   if (shift)
     adjust_form(form, indexes, indLen, strcount, formlen, shift);

   ret=send_response(pECB, form, strlen(form));

   if (shift) {
     allocate_last_form(form2,pool);
     memcpy(form, form2, formlen+1);
```

```
    }
    for (j=lineStart; j<15; j++) {
      form[form_index[SUBI][NO_PRICE+j*8].index-1]='$';
      form[form_index[SUBI][NO_AMOUNT+j*8].index-1]='$';
    }

#ifndef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endif

    return ret;
#undef SUBI
}

int sendform_paymentoutput(int status, T_payment_data *pdata)
{
                EXTENSION_CONTROL_BLOCK *pECB;
    int w_id, ld_id, ret;
    char *form, *form2;
    char blank[] = "                                        ";
    int index = -1, formlen, strcount=0, shift=0, i=0, j,datalen;
    int indexes[PA_FORMINDEX_SIZE],
indLen[PA_FORMINDEX_SIZE], index2=-1;
    form_template_pool *pool;

    w_id = pdata->w_id; ld_id = pdata->ld_id;
                pECB = (EXTENSION_CONTROL_BLOCK *) pdata-
>context;

    if (status != SUCCESS && status != DB_SUCCESS) {
      return send_error_message(pECB, 0, status, "", w_id, ld_id, 0);
    }

    if (pdata->txn_status != DB_RETURN_OCI_SUCCESS) {
      return send_error_message(pECB, 0, pdata->txn_status, "   ---
DATABASE ERROR ", w_id, ld_id, 0);
    }

#define SUBI POOL_TYPE_TXN_OUTPUT][TXN_TYPE_PAYMENT

    pool = &txn_global_pool[SUBI];
                index=w_id*10+ld_id;

#ifdef NEW_ALLOCATE_FORM
    form  = allocate_form_new(pool, index);
#else
    form  = allocate_form(pool, &index);
#endif

    formlen=strlen(form);

    fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][PA_TERMID].index,
              form_index[SUBI][PA_TERMID].length);

    fill_string(form, pdata->h_date.DateString,
form_index[SUBI][PA_DATE].index,
               form_index[SUBI][PA_DATE].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_DATE].index;
    indLen[strcount++]=form_index[SUBI][PA_DATE].length;

    fill_number(form, w_id, form_index[SUBI][PA_WID].index,
              form_index[SUBI][PA_WID].length);

    fill_number(form, pdata->d_id, form_index[SUBI][PA_DID].index,
               form_index[SUBI][PA_DID].length);

    fill_string(form, pdata->w_street_1,
form_index[SUBI][PA_WST1].index,
               form_index[SUBI][PA_WST1].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_WST1].index;
    indLen[strcount++]=form_index[SUBI][PA_WST1].length;

    fill_string(form, pdata->d_street_1,
form_index[SUBI][PA_DST1].index,
            form_index[SUBI][PA_DST1].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_DST1].index;
    indLen[strcount++]=form_index[SUBI][PA_DST1].length;

    fill_string(form, pdata->w_street_2,
form_index[SUBI][PA_WST2].index,
            form_index[SUBI][PA_WST2].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_WST2].index;
    indLen[strcount++]=form_index[SUBI][PA_WST2].length;

    fill_string(form, pdata->d_street_2,
form_index[SUBI][PA_DST2].index,
            form_index[SUBI][PA_DST2].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_DST2].index;
    indLen[strcount++]=form_index[SUBI][PA_WST2].length;

    fill_string(form, pdata->w_city, form_index[SUBI][PA_WCITY].index,
            form_index[SUBI][PA_WCITY].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_WCITY].index;
    indLen[strcount++]=form_index[SUBI][PA_WCITY].length;

    fill_string(form, pdata->w_state,
form_index[SUBI][PA_WSTATE].index,
            form_index[SUBI][PA_WSTATE].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_WSTATE].index;
    indLen[strcount++]=form_index[SUBI][PA_WSTATE].length;

    fill_string(form, pdata->w_zip, form_index[SUBI][PA_WZIP].index,
            form_index[SUBI][PA_WZIP].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_WZIP].index;
    indLen[strcount++]=form_index[SUBI][PA_WZIP].length;

    fill_string(form, pdata->d_city, form_index[SUBI][PA_DCITY].index,
            form_index[SUBI][PA_DCITY].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_DCITY].index;
    indLen[strcount++]=form_index[SUBI][PA_DCITY].length;

    fill_string(form, pdata->d_state,
form_index[SUBI][PA_DSTATE].index,
            form_index[SUBI][PA_DSTATE].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_DSTATE].index;
    indLen[strcount++]=form_index[SUBI][PA_DSTATE].length;

    fill_string(form, pdata->d_zip, form_index[SUBI][PA_DZIP].index,
            form_index[SUBI][PA_DZIP].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_DZIP].index;
    indLen[strcount++]=form_index[SUBI][PA_DZIP].length;

    fill_number(form, pdata->c_id, form_index[SUBI][PA_CID].index,
            form_index[SUBI][PA_CID].length);

    fill_number(form, pdata->c_w_id,
form_index[SUBI][PA_CWARE].index,
            form_index[SUBI][PA_CWARE].length);

    fill_number(form, pdata->c_d_id,
form_index[SUBI][PA_CDIST].index,
            form_index[SUBI][PA_CDIST].length);

    fill_string(form, pdata->c_first, form_index[SUBI][PA_CFIRST].index,
            form_index[SUBI][PA_CFIRST].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CFIRST].index;
    indLen[strcount++]=form_index[SUBI][PA_CFIRST].length;
    fill_string(form, pdata->c_middle,
form_index[SUBI][PA_CMIDDLE].index,
              form_index[SUBI][PA_CMIDDLE].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CMIDDLE].index;
    indLen[strcount++]=form_index[SUBI][PA_CMIDDLE].length;

    fill_string(form, pdata->c_last, form_index[SUBI][PA_CLAST].index,
              form_index[SUBI][PA_CLAST].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CLAST].index;
    indLen[strcount++]=form_index[SUBI][PA_CLAST].length;

    fill_string(form, pdata->c_since.DateString,
form_index[SUBI][PA_SINCE].index,
              form_index[SUBI][PA_SINCE].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_SINCE].index;
    indLen[strcount++]=form_index[SUBI][PA_SINCE].length;

    fill_string(form, pdata->c_street_1,
form_index[SUBI][PA_CST1].index,
              form_index[SUBI][PA_CST1].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CST1].index;
    indLen[strcount++]=form_index[SUBI][PA_CST1].length;

    fill_string(form, pdata->c_credit,
form_index[SUBI][PA_CREDIT].index,
              form_index[SUBI][PA_CREDIT].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CREDIT].index;
    indLen[strcount++]=form_index[SUBI][PA_CREDIT].length;

    fill_string(form, pdata->c_street_2,
form_index[SUBI][PA_CST2].index,
              form_index[SUBI][PA_CST2].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CST2].index;
    indLen[strcount++]=form_index[SUBI][PA_CST2].length;

    fill_double(form, pdata->c_discount,
form_index[SUBI][PA_DISC].index,
             form_index[SUBI][PA_DISC].length);

    fill_string(form, pdata->c_city, form_index[SUBI][PA_CCITY].index,
              form_index[SUBI][PA_CCITY].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CCITY].index;
    indLen[strcount++]=form_index[SUBI][PA_CCITY].length;

    fill_string(form, pdata->c_state,
form_index[SUBI][PA_CSTATE].index,
              form_index[SUBI][PA_CSTATE].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CSTATE].index;
    indLen[strcount++]=form_index[SUBI][PA_CSTATE].length;

    fill_string(form, pdata->c_zip, form_index[SUBI][PA_CZIP].index,
              form_index[SUBI][PA_CZIP].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CZIP].index;
    indLen[strcount++]=form_index[SUBI][PA_CZIP].length;

    fill_string(form, pdata->c_phone,
form_index[SUBI][PA_CPHONE].index,
              form_index[SUBI][PA_CPHONE].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_CPHONE].index;
    indLen[strcount++]=form_index[SUBI][PA_CPHONE].length;

    fill_double(form, pdata->h_amount,
form_index[SUBI][PA_AMOUNT].index,
             form_index[SUBI][PA_AMOUNT].length);

    fill_double(form, pdata->c_balance,
form_index[SUBI][PA_CBAL].index,
             form_index[SUBI][PA_CBAL].length);

    fill_double(form, pdata->c_credit_lim,
form_index[SUBI][PA_LIMIT].index,
             form_index[SUBI][PA_LIMIT].length);

    if (pdata->c_credit[0]=='B' && pdata->c_credit[1]=='C') {
      datalen=strlen(pdata->c_data);
      for (i=0; i<4; i++) {
        if (i * form_index[SUBI][PA_CUSTDATA+i].length >= datalen)
break;
        fill_string(form, pdata-
>c_data+(i*form_index[SUBI][PA_CUSTDATA+i].length),
                 form_index[SUBI][PA_CUSTDATA+i].index,
                 form_index[SUBI][PA_CUSTDATA+i].length, &shift);
        indexes[strcount]=form_index[SUBI][PA_CUSTDATA+i].index;
        indLen[strcount++]=form_index[SUBI][PA_CUSTDATA+i].length;
      }
    }

    for (j=i; j<4; j++)
      memcpy(form+form_index[SUBI][PA_CUSTDATA+j].index, blank,
           form_index[SUBI][PA_CUSTDATA+j].length);

    if (shift)
      adjust_form(form, indexes, indLen, strcount, formlen, shift);

    ret=send_response(pECB, form, strlen(form));

    if (shift) {
      allocate_last_form(form2, pool);
      memcpy(form, form2, formlen+1);
    }

#ifndef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endif

    return ret;
#undef SUBI
}



int sendform_orderstatusoutput(int status, T_orderstatus_data *pdata)
{
              EXTENSION_CONTROL_BLOCK *pECB;
    int w_id, ld_id, indexes[OS_FORMINDEX_SIZE],
indLen[OS_FORMINDEX_SIZE];
    char *form, *form2;
    int index = -1, strcount=0, formlen, shift=0, i, j, index2=-1, lineStart=15,
ret;
    form_template_pool *pool;
    char blank[] = "                    ";

    w_id = pdata->w_id; ld_id = pdata->ld_id;
              pECB = (EXTENSION_CONTROL_BLOCK *) pdata-
>context;

    if (status != SUCCESS && status != DB_SUCCESS) {
      return send_error_message(pECB, 0, status, "", w_id, ld_id, 0);
    }

    if (pdata->txn_status != DB_RETURN_OCI_SUCCESS) {
      return send_error_message(pECB, 0, pdata->txn_status, "   ---
DATABASE ERROR ", w_id, ld_id, 0);
    }

#define SUBI
POOL_TYPE_TXN_OUTPUT][TXN_TYPE_ORDERSTATUS

    pool = &txn_global_pool[SUBI];
```

```
                index=w_id*10+ld_id;

#ifdef NEW_ALLOCATE_FORM
    form  = allocate_form_new(pool, index);
#else
                form  = allocate_form(pool, &index);
#endif

    formlen = strlen(form);

    fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][OS_TERMID].index,
            form_index[SUBI][OS_TERMID].length);
    fill_number(form, w_id, form_index[SUBI][OS_WID].index,
            form_index[SUBI][OS_WID].length);
    fill_number(form, pdata->d_id, form_index[SUBI][OS_DID].index,
            form_index[SUBI][OS_DID].length);
    fill_number(form, pdata->c_id, form_index[SUBI][OS_CID].index,
            form_index[SUBI][OS_CID].length);
    fill_string(form, pdata->c_first, form_index[SUBI][OS_FIRST].index,
            form_index[SUBI][OS_FIRST].length, &shift);
    indexes[strcount]=form_index[SUBI][OS_FIRST].index;
    indLen[strcount++]=form_index[SUBI][OS_FIRST].length;

    fill_string(form, pdata->c_middle,
form_index[SUBI][OS_MIDDLE].index,
            form_index[SUBI][OS_MIDDLE].length, &shift);
    indexes[strcount]=form_index[SUBI][OS_MIDDLE].index;
    indLen[strcount++]=form_index[SUBI][OS_MIDDLE].length;

    fill_string(form, pdata->c_last, form_index[SUBI][OS_LAST].index,
            form_index[SUBI][OS_LAST].length, &shift);
    indexes[strcount]=form_index[SUBI][OS_LAST].index;
    indLen[strcount++]=form_index[SUBI][OS_LAST].length;

    fill_double(form, pdata->c_balance,
form_index[SUBI][OS_CBALANCE].index,
            form_index[SUBI][OS_CBALANCE].length);

    fill_number(form, pdata->o_id, form_index[SUBI][OS_OID].index,
            form_index[SUBI][OS_OID].length);

    fill_string(form, pdata->o_entry_d.DateString,
form_index[SUBI][OS_ENTRY_DATE].index,
            form_index[SUBI][OS_ENTRY_DATE].length, &shift);
    indexes[strcount]=form_index[SUBI][OS_ENTRY_DATE].index;
    indLen[strcount++]=form_index[SUBI][OS_ENTRY_DATE].length;

    fill_number(form, pdata->o_carrier_id,
form_index[SUBI][OS_CARID].index,
            form_index[SUBI][OS_CARID].length);

    for (i=0; i < pdata->o_ol_cnt; i++) {
      fill_number(form, pdata->o_orderline[i].ol_supply_w_id,
              form_index[SUBI][OS_SUPW+i*5].index,
              form_index[SUBI][OS_SUPW+i*5].length);

      fill_number(form, pdata->o_orderline[i].ol_i_id,
              form_index[SUBI][OS_ITEMID+i*5].index,
              form_index[SUBI][OS_ITEMID+i*5].length);

      fill_number(form, pdata->o_orderline[i].ol_quantity,
              form_index[SUBI][OS_QTY+i*5].index,
              form_index[SUBI][OS_QTY+i*5].length);

      fill_double(form, pdata->o_orderline[i].ol_amount,
             form_index[SUBI][OS_AMOUNT+i*5].index,
             form_index[SUBI][OS_AMOUNT+i*5].length);

      fill_string(form, pdata->o_orderline[i].ol_delivery_d.DateString,
              form_index[SUBI][OS_DELDATE+i*5].index,
              form_index[SUBI][OS_DELDATE+i*5].length, &shift);
      indexes[strcount]=form_index[SUBI][OS_DELDATE+i*5].index;
      indLen[strcount++]=form_index[SUBI][OS_DELDATE+i*5].length;
    }

    for (lineStart=j=i; j<15;j++) {
      memcpy(form+form_index[SUBI][OS_SUPW+j*5].index, blank,
           form_index[SUBI][OS_SUPW+j*5].length);
      memcpy(form+form_index[SUBI][OS_ITEMID+j*5].index, blank,
           form_index[SUBI][OS_ITEMID+j*5].length);
      memcpy(form+form_index[SUBI][OS_QTY+j*5].index, blank,
           form_index[SUBI][OS_QTY+j*5].length);
      memcpy(form+form_index[SUBI][OS_AMOUNT+j*5].index-1, blank,
           form_index[SUBI][OS_AMOUNT+j*5].length+1);
      memcpy(form+form_index[SUBI][OS_DELDATE+j*5].index, blank,
           form_index[SUBI][OS_DELDATE+j*5].length);
    }

    if (shift)
      adjust_form(form, indexes, indLen, strcount, formlen, shift);

    ret=send_response(pECB, form, strlen(form));

    if (shift) {
      allocate_last_form(form2, pool);
      memcpy(form, form2, formlen+1);
    }

    for (j=lineStart; j<15; j++)
      form[form_index[SUBI][OS_AMOUNT+j*5].index-1]='$';

#ifndef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endif

    return ret;
#undef SUBI
}



int sendform_deliveryoutput(int status, T_delivery_data *pdata)
{
                EXTENSION_CONTROL_BLOCK *pECB;
    int w_id, ld_id;
    char *form;
    int index = -1, ret;
    form_template_pool *pool;

    w_id = pdata->w_id; ld_id = pdata->ld_id;
                pECB = (EXTENSION_CONTROL_BLOCK *) pdata-
>context;
    if (status != SUCCESS && status != DB_SUCCESS) {
      return send_error_message(pECB, 0, status, "", w_id, ld_id, 0);
    }

#define SUBI POOL_TYPE_TXN_OUTPUT][TXN_TYPE_DELIVERY

    pool = &txn_global_pool[SUBI];
                index=w_id*10+ld_id;

#ifdef NEW_ALLOCATE_FORM
    form  = allocate_form_new(pool, index);
#else
    form  = allocate_form(pool, &index);
#endif

    fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][DE_TERMID].index,
            form_index[SUBI][DE_TERMID].length);
    fill_number(form, w_id, form_index[SUBI][DE_WID].index,
            form_index[SUBI][DE_WID].length);
    fill_number(form, pdata->o_carrier_id,
form_index[SUBI][DE_CARID].index,
            form_index[SUBI][DE_CARID].length);

    ret=send_response(pECB, form, strlen(form));

#ifndef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endif

    return ret;
#undef SUBI
}



int sendform_stockleveloutput(int status, T_stocklevel_data *pdata)
{
              EXTENSION_CONTROL_BLOCK *pECB;
    int w_id, ld_id;
    char *form;
    int index = -1, ret;
    form_template_pool *pool;

    w_id = pdata->w_id; ld_id = pdata->ld_id;
              pECB = (EXTENSION_CONTROL_BLOCK *) pdata-
>context;

    if (status != SUCCESS && status != DB_SUCCESS) {
      return send_error_message(pECB, 0, status, "", w_id, ld_id, 0);
    }


    if (pdata->txn_status != DB_RETURN_OCI_SUCCESS) {
      return send_error_message(pECB, 0, pdata->txn_status, "   ---
DATABASE ERROR ", w_id, ld_id, 0);
    }

#define SUBI
POOL_TYPE_TXN_OUTPUT][TXN_TYPE_STOCKLEVEL

    pool = &txn_global_pool[SUBI];
              index=w_id*10+ld_id;

#ifdef NEW_ALLOCATE_FORM
    form  = allocate_form_new(pool, index);
#else
    form  = allocate_form(pool, &index);
#endif


    fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][SL_TERMID].index,
            form_index[SUBI][SL_TERMID].length);
    fill_number(form, w_id, form_index[SUBI][SL_WID].index,
            form_index[SUBI][SL_WID].length);
    fill_number(form, ld_id, form_index[SUBI][SL_DID].index,
            form_index[SUBI][SL_DID].length);
    fill_number(form, pdata->threshold,
form_index[SUBI][SL_THRESHOLD].index,
            form_index[SUBI][SL_THRESHOLD].length);
    fill_number(form, pdata->low_stock,
form_index[SUBI][SL_LOWSTOCK].index,
            form_index[SUBI][SL_LOWSTOCK].length);

    ret=send_response(pECB, form, strlen(form));

#ifndef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endif

    return ret;
#undef SUBI
}


int (FAR * mod_tpcc_neworder)(T_neworder_data *);
int (FAR * mod_tpcc_payment)(T_payment_data *);
int (FAR * mod_tpcc_delivery)(T_delivery_data *, int);
int (FAR * mod_tpcc_orderstatus)(T_orderstatus_data *);
int (FAR * mod_tpcc_stocklevel)(T_stocklevel_data *);
void (FAR * userlog)(char * str, ...);
void (FAR * initDelLog)(int);
void (FAR * endDelLog)(int);


- - - - - - - - - - - - - - - - - -
mod_tpcc_error.h
- - - - - - - - - - - - - - - - - -
/* Copyright (c) 2004, Oracle Corporation.  All rights reserved.  */

/*
   NAME
     mod_tpcc_error.h - <one-line expansion of the name>

   DESCRIPTION
     <short description of facility this file declares/defines>

   RELATED DOCUMENTS
     <note any documents related to this facility>

   EXPORT FUNCTION(S)
     <external functions declared for use outside package - one-line
descriptions>

   INTERNAL FUNCTION(S)
     <other external functions declared - one-line descriptions>

   EXAMPLES

   NOTES
     <other useful comments, qualifications, etc.>

   MODIFIED   (MM/DD/YY)
   xnie        02/09/04 - to make it work with tuxedo
   shuang      01/22/04 - shuang_rte
   shuang      01/21/04 - Creation

*/

#define DB_SUCCESS            0
#define DB_ERROR              1
#define TRANSPORT_ERROR       2
#define DB_INTERFACE          3
#define DB_DEADLOCK_LIMIT     4
#define DB_NOT_COMMITED       5
#define DB_DEAD               6
#define DB_PENDING            7
#define DB_NOT_LOGGED_IN      8
#define DB_LOGIN_FAILED       9
#define DB_USE_FAILED         10
#define DB_LOGOUT_FAILED      11
#define DB_TUXEDO_TPALLOC_ERROR       12
#define DB_TUXEDO_TPCALL_ERROR        13
#define DB_MAX_ERR            13
#define VALID_DB_ERR(err) (((err) >= DB_SUCCESS)&&((err) <=
DB_MAX_ERR))
```

```
#define SUCCESS                         1000
#define COMMAND_UNDEFINED               1001
#define NOT_IMPLEMENTED_YET             1002
#define CANNOT_INIT_TERMINAL            1003
#define OUT_OF_MEMORY                   1004
#define NEW_ORDER_NOT_PROCESSED         1005
#define PAYMENT_NOT_PROCESSED           1006
#define NO_SERVER_SPECIFIED             1007
#define ORDER_STATUS_NOT_PROCESSED      1008
#define W_ID_INVALID                    1009
#define CAN_NOT_SET_MAX_CONNECTIONS     1010
#define UNKNOW_TRANSACTION_TYPE         1011
#define D_ID_INVALID                    1012
#define MAX_CONNECT_PARAM               1013
#define INVALID_SYNC_CONNECTION         1014
#define INVALID_TERMID                  1015
#define PAYMENT_INVALID_CUSTOMER        1016
#define SQL_OPEN_CONNECTION             1017

#define STOCKLEVEL_MISSING_THRESHOLD_KEY 1018
#define STOCKLEVEL_THRESHOLD_INVALID 1019
#define STOCKLEVEL_THRESHOLD_RANGE 1020
#define STOCKLEVEL_NOT_PROCESSED 1021
#define NEWORDER_MISSING_DID 1022
#define NEWORDER_DISTRICT_INVALID 1023
#define NEWORDER_DISTRICT_RANGE 1024
#define NEWORDER_CUSTOMER_KEY 1025
#define NEWORDER_CUSTOMER_INVALID 1026
#define NEWORDER_CUSTOMER_RANGE 1027
#define NEWORDER_MISSING_IID_KEY 1028
#define NEWORDER_ITEM_BLANK_LINES 1029
#define NEWORDER_ITEMID_INVALID 1030
#define NEWORDER_MISSING_SUPPW_KEY 1031
#define NEWORDER_SUPPW_INVALID 1032
#define NEWORDER_MISSING_QTY_KEY 1033
#define NEWORDER_QTY_INVALID 1034
#define NEWORDER_SUPPW_RANGE 1035
#define NEWORDER_ITEMID_RANGE 1036
#define NEWORDER_QTY_RANGE 1037
#define NEWORDER_SUPPW_WITHOUT_ITEMID 1039
#define NEWORDER_QTY_WITHOUT_ITEMID 1040
#define NEWORDER_NOITEMS_ENTERED 1041
#define PAYMENT_MISSING_DID_KEY 1042
#define PAYMENT_DISTRICT_INVALID 1038
#define PAYMENT_DISTRICT_RANGE 1043
#define PAYMENT_MISSING_CID_KEY 1044
#define PAYMENT_CUSTOMER_INVALID 1045
#define PAYMENT_MISSING_CLASTNAME 1046
#define PAYMENT_LAST_NAME_TO_LONG 1047
#define PAYMENT_CID_RANGE  1048
#define PAYMENT_CID_AND_CLASTNAME 1049
#define PAYMENT_MISSING_CDI_KEY 1050
#define PAYMENT_CDI_INVALID 1051
#define PAYMENT_CDI_RANGE 1052
#define PAYMENT_MISSING_CWI_KEY 1053
#define PAYMENT_CWI_INVALID 1054
#define PAYMENT_CWI_RANGE 1055
#define PAYMENT_MISSING_HAM_KEY 1056
#define PAYMENT_HAM_INVALID 1057
#define PAYMENT_HAM_RANGE 1058
#define ORDERSTATUS_MISSING_DID_KEY 1059
#define ORDERSTATUS_DID_INVALID 1060
#define ORDERSTATUS_DID_RANGE 1061
#define ORDERSTATUS_MISSING_CID_KEY 1062
#define ORDERSTATUS_MISSING_CLASTNAME_KEY 1063
#define ORDERSTATUS_CLASTNAME_RANGE 1064
#define ORDERSTATUS_CID_INVALID 1065
#define ORDERSTATUS_CID_RANGE 1066
#define ORDERSTATUS_CID_AND_CLASTNAME 1067
#define DELIVERY_MISSING_OCD_KEY 1068
#define DELIVERY_CARRIER_INVALID 1069
#define DELIVERY_CARRIER_ID_RANGE 1070

#define PAYMENT_MISSING_CLASTNAME_KEY 1071
#define CANT_FIND_TPCC_KEY 1072
#define CANT_FIND_INETINFO_KEY 1073
#define CANT_FIND_POOLTHREADLIMIT 1074
#define DB_DELIVERY_NOT_QUEUED 1075
#define DELIVERY_NOT_PROCESSED 1076
#define TERM_ALLOCATE_FAILED 1077
#define PENDING 1078
#define CANT_START_FRCDINIT_THREAD 1079
#define CANT_START_DELIVERY_THREAD 1080
#define GOVERNOR_VALUE_NOT_FOUND 1081
#define SERVER_MISMATCH 1082
#define DATABASE_MISMATCH 1083
#define USER_MISMATCH 1084
#define PASSWORD_MISMATCH 1085
#define CANT_CREATE_ALL_THREADS_EVENT 1086
#define CANT_CREATE_FORCE_THRED_STRT_EVENT 1087
#define CANT_ALLOCATE_THREAD_LOCAL_STORAGE 1088
#define CANT_SET_THREAD_LOCAL_STORAGE 1089
#define FORCE_CONNECT_THREAD_FAILED 1090
#define CANT_FIND_SERVER_VALUE 1091
#define NO_MESSAGE 1092
#define CANT_FIND_PATH_VALUE 1093
#define CANNOT_CREATE_RESULTS_FILE 1094
#define DELIVERY_PIPE_SECURITY 1095
#define DELIVERY_PIPE_CREATE 1096
#define DELIVERY_PIPE_OPEN 1097
#define DELIVERY_PIPE_READ 1098
#define DELIVERY_PIPE_DISCONNECT 1099
#define CANT_FIND_DATABASE_VALUE 1100
#define CANT_FIND_USER_VALUE 1101
#define CANT_FIND_PASSWORD_VALUE 1102
#define DELIVERY_OUTPUT_PIPE_WRITE 1103
#define DELIVERY_OUTPUT_PIPE_READ 1104
#define DELIVERY_MISSING_QUEUETIME_KEY 1105
#define DELIVERY_QUEUETIME_INVALID 1106
#define ALREADY_LOGGED_IN 1107
#define INVALID_FORM 1109
#define DELIVERY_MUST_CONNECTDB 1110
#define INVALID_FORM_AND_CMD_NOT_BEGIN 1111
#define MAX_CONNECTIONS_EXCEEDED 1112
#define CANNOT_FIND_CONNECTION 1113
#define CKPT_NOT_INITIALIZED 1114
#define PAYMENT_MISSING_CID_CLASTNAME 1115
#define CANT_FIND_MAXDBCONNECTIONS_VALUE 1116
#define PAYMENT_CUSTOMER_RANGE 1117

/* OCI return status */

#define DB_RETURN_OCI_SUCCESS 1118
#define DB_RETURN_OCI_SUCCESS_WITH_INFO 1119
#define DB_RETURN_OCI_NEED_DATA 1120
#define DB_RETURN_OCI_NO_DATA 1121
#define DB_RETURN_OCI_ERROR 1122
#define DB_RETURN_OCI_INVALID_HANDLE 1123
#define DB_RETURN_OCI_STILL_EXECUTING 1124
#define DB_RETURN_OCI_CONTINUE 1125

struct T_error_message
{
   int error_code;
   char error_mesg[80];
};
typedef struct T_error_message T_error_message;

T_error_message error_message [] =
{
  { SUCCESS, "Success, no error." },
  { NO_MESSAGE, "No message string available for the specified error
code." },
  { COMMAND_UNDEFINED, "Command undefined." },
  { NOT_IMPLEMENTED_YET, "Not Implemented Yet." },
  { CANNOT_INIT_TERMINAL, "Cannot initialize client connection." },
  { OUT_OF_MEMORY, "Insufficient memory." },
  { NEW_ORDER_NOT_PROCESSED, "Cannot process new Order
form." },
  { PAYMENT_NOT_PROCESSED, "Cannot process payment form." },
  { NO_SERVER_SPECIFIED, "No Server name specified." },
  { ORDER_STATUS_NOT_PROCESSED, "Cannot process order status
form." },
  { W_ID_INVALID, "Invalid Warehouse ID." },
  { CAN_NOT_SET_MAX_CONNECTIONS, "Insufficient memory to
allocate # connections." },
  { D_ID_INVALID, "Invalid District ID Must be 1 to 10." },
  { MAX_CONNECT_PARAM, "Max client connections exceeded, run
install to increase." },
  { INVALID_SYNC_CONNECTION, "Invalid Terminal Sync ID." },
  { INVALID_TERMID, "Invalid Terminal ID." },
  { PAYMENT_INVALID_CUSTOMER, "Payment Form, No such
Customer." },
  { SQL_OPEN_CONNECTION, "SQLOpenConnection API Failed." },
  { STOCKLEVEL_MISSING_THRESHOLD_KEY, "Stock Level missing
Threshold key \"TT*\"." },
  { STOCKLEVEL_THRESHOLD_INVALID, "Stock Level Threshold
invalid data type range = 1 - 99." },
  { STOCKLEVEL_THRESHOLD_RANGE, "Stock Level Threshold out
of range, range must be 1 - 99." },
  { STOCKLEVEL_NOT_PROCESSED, "Stock Level not processed." },
  { NEWORDER_MISSING_DID, "New Order missing District key
\"DID*\"." },
  { NEWORDER_DISTRICT_INVALID, "New Order District ID Invalid
range 1 - 10." },
  { NEWORDER_DISTRICT_RANGE, "New Order District ID out of
Range. Range = 1 - 10." },
  { NEWORDER_CUSTOMER_KEY, "New Order missing Customer key
\"CID*\"." },
  { NEWORDER_CUSTOMER_INVALID, "New Order customer id
invalid data type, range = 1 to 3000." },
  { NEWORDER_CUSTOMER_RANGE, "New Order customer id out of
range, range = 1 to 3000." },
  { NEWORDER_MISSING_IID_KEY, "New Order missing Item Id key
\"IID*\"." },
  { NEWORDER_ITEM_BLANK_LINES, "New Order blank order lines
all orders must be continuous." },
  { NEWORDER_ITEMID_INVALID, "New Order Item Id is wrong data
type, must be numeric." },
  { NEWORDER_MISSING_SUPPW_KEY, "New Order missing
Supp_W key \"SP##*\"." },
  { NEWORDER_SUPPW_INVALID, "New Order Supp_W invalid data
type must be numeric." },
  { NEWORDER_MISSING_QTY_KEY, "New Order Missing Qty key
\"Qty##*\"." },
  { NEWORDER_QTY_INVALID, "New Order Qty invalid must be
numeric range 1 - 99." },
  { NEWORDER_SUPPW_RANGE, "New Order Supp_W value out of
range range = 1 - Max Warehouses." },
  { NEWORDER_ITEMID_RANGE, "New Order Item Id is out of range.
Range = 1 to 999999." },
  { NEWORDER_QTY_RANGE, "New Order Qty is out of range. Range
= 1 to 99." },
  { PAYMENT_DISTRICT_INVALID, "Payment District ID is invalid
must be 1 - 10." },
  { NEWORDER_SUPPW_WITHOUT_ITEMID, "New Order Supp_W
field entered without a corrisponding Item_Id." },
  { NEWORDER_QTY_WITHOUT_ITEMID, "New Order Qty entered
without a corrisponding Item_Id." },
  { NEWORDER_NOITEMS_ENTERED, "New Order Blank Items
between items, items must be continuous." },
  { PAYMENT_MISSING_DID_KEY, "Payment missing District Key
\"DID*\"." },
  { PAYMENT_DISTRICT_RANGE, "Payment District Out of range,
range = 1 - 10." },
  { PAYMENT_MISSING_CID_KEY, "Payment missing Customer Key
\"CID*\"." },
  { PAYMENT_CUSTOMER_INVALID, "Payment Customer data type
invalid, must be numeric." },
  { PAYMENT_MISSING_CLASTNAME, "Payment missing Customer
Last Name Key \"CLASTNAME*\"." },
  { PAYMENT_MISSING_CID_CLASTNAME, "Payment entered
without Customer ID or last Name. " },
  { PAYMENT_LAST_NAME_TO_LONG, "Payment Customer last name
longer than 16 characters." },
  { PAYMENT_CUSTOMER_RANGE, "Payment Customer ID out of
range, must be 1 to 3000." },
  { PAYMENT_CID_AND_CLASTNAME, "Payment Customer ID and
Last Name entered must be one or other." },
  { PAYMENT_MISSING_CDI_KEY, "Payment missing Customer
district key \"CDI*\"." },
  { PAYMENT_CDI_INVALID, "Payment Customer district invalid must
be numeric." },
  { PAYMENT_CDI_RANGE, "Payment Customer district out of range
must be 1 - 10." },
  { PAYMENT_MISSING_CWI_KEY, "Payment missing Customer
Warehouse key \"CWI*\"." },
  { PAYMENT_CWI_INVALID, "Payment Customer Warehouse invalid
must be numeric." },
  { PAYMENT_CWI_RANGE, "Payment Customer Warehouse out of
range, 1 to Max Warehouses." },
  { PAYMENT_MISSING_HAM_KEY, "Payment missing Amount key
\"HAM*\"." },
  { PAYMENT_HAM_INVALID, "Payment Amount invalid data type
must be numeric." },
  { PAYMENT_HAM_RANGE, "Payment Amount out of range, 0 -
9999.99." },
  { ORDERSTATUS_MISSING_DID_KEY, "Order Status missing
District key \"DID*\"." },
  { ORDERSTATUS_DID_INVALID, "Order Status District invalid, value
must be numeric 1 - 10." },
  { ORDERSTATUS_DID_RANGE, "Order Status District out of range
must be 1 - 10." },
  { ORDERSTATUS_MISSING_CID_KEY, "Order Status missing
Customer key \"CID*\"." },
  { ORDERSTATUS_MISSING_CLASTNAME_KEY, "Order Status
missing Customer Last Name key \"CLASTNAME*\"." },
  { ORDERSTATUS_CLASTNAME_RANGE, "Order Status Customer
last name longer than 16 characters." },
  { ORDERSTATUS_CID_INVALID, "Order Status Customer ID invalid,
range must be numeric 1 - 3000." },
  { ORDERSTATUS_CID_RANGE, "Order Status Customer ID out of
range must be 1 - 3000." },
  { ORDERSTATUS_CID_AND_CLASTNAME, "Order Status Customer
ID and LastName entered must be only one." },
  { DELIVERY_MISSING_OCD_KEY, "Delivery missing Carrier ID key
\"OCD*\"." },
  { DELIVERY_CARRIER_INVALID, "Delivery Carrier ID invalid must
be numeric 1 - 10." },
  { DELIVERY_CARRIER_ID_RANGE, "Delivery Carrier ID out of
range must be 1 - 10." },
  { PAYMENT_MISSING_CLASTNAME_KEY, "Payment missing
Customer Last Name key \"CLASTNAME*\"." },
  { DB_ERROR, "A Database error has occurred." },
  { DB_TUXEDO_TPALLOC_ERROR, "Tuxedo call tpalloc has failed."
},
  { DB_TUXEDO_TPCALL_ERROR, "Tuxedo call tpcall has failed." },
  { DELIVERY_NOT_PROCESSED, "Delivery not processed." },
  { DB_DELIVERY_NOT_QUEUED, "Delivery not queued." },
  { CANT_FIND_TPCC_KEY, "TPCC key not found in registry." },
  { CANT_FIND_INETINFO_KEY, "inetinfo key not found in registry." },
  { CANT_FIND_POOLTHREADLIMIT, "PoolThreadLimit value not set
in inetinfo\\Parameters key." },
  { TERM_ALLOCATE_FAILED, "Failed to allocate terminal data
structure." },
  { DELIVERY_PIPE_SECURITY, "Failed to initialize delivery pipe
security." },
  { DELIVERY_PIPE_CREATE, "Failed to create delivery pipe." },
  { DELIVERY_PIPE_OPEN, "Failed to open delivery pipe." },
  { DELIVERY_PIPE_READ, "Failed to read delivery pipe." },
  { DELIVERY_PIPE_DISCONNECT, "Failed to start delivery pipe
disconnect thread."},
  { PENDING, "Transaction pending."},
  { CANT_START_FRCDINIT_THREAD, "Can't start Forced
Initialization thread." },
  { CANT_START_DELIVERY_THREAD, "Can't start delivery thread."
},
  { GOVERNOR_VALUE_NOT_FOUND, "Governor value not found in
Registry." },
  { SERVER_MISMATCH, "Server does not match registry value." },
  { DATABASE_MISMATCH, "Database name does not match registry
value." },
  { USER_MISMATCH, "User name does not match registry value." },
  { PASSWORD_MISMATCH, "Password does not match registry value."
},
  { CANT_CREATE_ALL_THREADS_EVENT, "Can't create All Threads
Event." },
  { CANT_CREATE_FORCE_THRED_STRT_EVENT, "Can't create
Force Thread Start Event." },
  { CANT_ALLOCATE_THREAD_LOCAL_STORAGE, "Can't allocate
thread local storage" },
  { CANT_SET_THREAD_LOCAL_STORAGE, "Can't set thread local
storage." },
  { FORCE_CONNECT_THREAD_FAILED, "At least one database
connect call failed, check log files for specific error." },
  { CANT_FIND_SERVER_VALUE, "Server value not set in TPCC key."
},
  { CANT_FIND_PATH_VALUE, "PATH value not set in TPCC key." },
  { CANNOT_CREATE_RESULTS_FILE, "Cannot create results file." },
  { CANT_FIND_DATABASE_VALUE, "Database value not set in TPCC
key." },
  { CANT_FIND_USER_VALUE, "User value not set in TPCC key." },
  { CANT_FIND_PASSWORD_VALUE, "Password value not set in
TPCC key." },
  { DELIVERY_OUTPUT_PIPE_WRITE, "Failed to write output delivery
pipe." },
  { DELIVERY_OUTPUT_PIPE_READ, "Failed to read output delivery
pipe." },
  { DELIVERY_MISSING_QUEUETIME_KEY, "Delivery queue time
missing from query." },
  { DELIVERY_QUEUETIME_INVALID, "Delivery queue time is
invalid." },
  { ALREADY_LOGGED_IN, "TPCCConnectDB has already been
called." },
  { DB_NOT_LOGGED_IN, "TPCCConnectDB has not yet been called."
},
  { INVALID_FORM, "The FORM field is missing or invalid." },
  { DELIVERY_MUST_CONNECTDB, "Synchronous transport requires
delivery server connect to database." },
  { INVALID_FORM_AND_CMD_NOT_BEGIN, "The FORM field is
missing and CMD is not Begin." },
  { MAX_CONNECTIONS_EXCEEDED, "The maximum number of
connections has been exceeded." },
  { CANT_FIND_MAXDBCONNECTIONS_VALUE,
"MaxDBConnections value not set in TPCC key." },
  { CANNOT_FIND_CONNECTION, "Transport layer unable to find a
DBContext coresponding to the CallersContext." },
  { CKPT_NOT_INITIALIZED, "The checkpoint subsystem has not been
started." },
  { DB_RETURN_OCI_SUCCESS, "OCI SUCCESS" },
  { DB_RETURN_OCI_SUCCESS_WITH_INFO, "OCI SUCCESS WITH
INFO"},
  { DB_RETURN_OCI_NEED_DATA, "OCI NEED DATA"},
  { DB_RETURN_OCI_NO_DATA, "OCI NO DATA"},
  { DB_RETURN_OCI_ERROR, "OCI ERROR"},
  { DB_RETURN_OCI_INVALID_HANDLE, "OCI INVALID
HANDLE"},
  { DB_RETURN_OCI_STILL_EXECUTING, "OCI STILL
EXECUTING"},
  { DB_RETURN_OCI_CONTINUE, "OCI CONTINUE"},
  { 0, "" }
};


- - - - - - - - - - - - - - - - - -
mod_tpcc.h
- - - - - - - - - - - - - - - - - -
/* Copyright (c) 2004, Oracle Corporation.  All rights reserved.  */

/*
   NAME
     mod_tpcc.h - <one-line expansion of the name>

   DESCRIPTION
     <short description of facility this file declares/defines>

   RELATED DOCUMENTS
     <note any documents related to this facility>

   EXPORT FUNCTION(S)
     <external functions declared for use outside package - one-line
descriptions>

   INTERNAL FUNCTION(S)
     <other external functions declared - one-line descriptions>

   EXAMPLES

   NOTES
     <other useful comments, qualifications, etc.>

   MODIFIED   (MM/DD/YY)
   xnie        01/30/04 - the real mod_tpcc.h
   shuang      01/22/04 - shuang_rte
   shuang      01/21/04 - Creation

*/

#include <httpext.h>

#define CMD_PROCESS(p)       (p[0] == 'P') && (p[1] == 'r')
#define CMD_NEWORDER(p)      (p[0] == 'N')
#define CMD_PAYMENT(p)       (p[0] == 'P') && (p[1] == 'a')
#define CMD_DELIVERY(p)      (p[0] == 'D')
#define CMD_ORDERSTATUS(p)   (p[0] == 'O')
#define CMD_STOCKLEVEL(p)    (p[0] == 'S')
#define CMD_EXIT(p)          (p[0] == 'E')
#define CMD_MENU(p)          (p[0] == 'M')
#define CMD_BEGIN(p)         (p[0] == 'B')

#define TXN_TYPE_DELIVERY     0
#define TXN_TYPE_STOCKLEVEL   1
#define TXN_TYPE_NEWORDER     2
#define TXN_TYPE_ORDERSTATUS  3
#define TXN_TYPE_PAYMENT      4
#define TXN_TYPE_MAX          5

#define POOL_TYPE_TXN_INPUT   0
#define POOL_TYPE_TXN_OUTPUT  1
```

```
#define POOL_TYPE_TXN_MAX     2

#define MAX_FORM_INDEX        164
#define BUF_SIZE              4096
#define FILENAMESIZE          128
#define MYLOGFILE             "/tmp/mod_tpcc.log"
#define WDID(w_id,d_id)       (10 * w_id + (d_id - 1))

#define MAX(a, b)             ((a > b) ? a : b)
#define MIN(a, b)             ((a > b) ? b : a)
#define STRING_UPPERCASE(x) \
      { \
        int str_pos; \
        int len = strlen(x); \
        for (str_pos=0; str_pos < len; str_pos++) \
            x[str_pos] = toupper(x[str_pos]); \
      }

struct value_index_entry
{
   char *value;
   int length;
};
typedef struct value_index_entry value_index_entry;

struct form_index_entry
{
   int index;
   int length;
};
typedef struct form_index_entry form_index_entry;

struct form_template_pool
{
   CRITICAL_SECTION form_template_spinlock;
                                     /* mutex for serialization */
   int form_template_length;                  /* Length of each form  */
   int form_template_size;              /* Number of form in the pool */
   char *form_template_storage;
                              /* The space allocated for the whole pool */
   int free_slot;
   int *free_list;
};
typedef struct form_template_pool form_template_pool;

//static int tpcc_handler(request_rec *r);
//static int tpcc_post_config(apr_pool_t *p, apr_pool_t *pl,
//                    apr_pool_t *pt, server_rec *s);
//static void tpcc_child_init(apr_pool_t *p, server_rec *s);
//static void tpcc_register_hooks(apr_pool_t *p);

void allocate_response_pool();
void allocate_transaction_pool();
void allocate_template_pool();

int sendform_mainmenu(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id);
int sendform_welcome(EXTENSION_CONTROL_BLOCK *, char *);
int sendform_neworderinput(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id);
int sendform_paymentinput(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id);
int sendform_orderstatusinput(EXTENSION_CONTROL_BLOCK
*pECB, int w_id, int ld_id);
int sendform_deliveryinput(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id);
int sendform_stocklevelinput(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id);
int mod_neworder_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr);
int mod_delivery_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr);
int mod_payment_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr);
int mod_orderstatus_query(EXTENSION_CONTROL_BLOCK *pECB,
int w_id, int ld_id, char *ptr);
int mod_stocklevel_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr);
int process_query(EXTENSION_CONTROL_BLOCK *);
int mod_begin_cmd(EXTENSION_CONTROL_BLOCK *);
int mod_menu_cmd(EXTENSION_CONTROL_BLOCK *, int, int);
int mod_exit_cmd(EXTENSION_CONTROL_BLOCK *);
int send_error_message(EXTENSION_CONTROL_BLOCK *, int, int,char
*,int,int,void *);

int get_wid_did(char *iptr, int *wid, int *did, char **optr);
int getcharvalue(char *iptr, char key, char **optr);
char *allocate_form(form_template_pool *pool, int *index);
char *allocate_form_new(form_template_pool *pool, int index);
void free_form(form_template_pool *pool, char *form_template, int
index);
void make_txn_form_template(char *, char *, char *, char *, int);
int build_form_index(char *form, char *form_template, form_index_entry
*f_index, int length);
int send_response(EXTENSION_CONTROL_BLOCK *, char *, int);
void fill_number(char *form, int value, int index, int length);
void fill_double(char *form, double value, int index, int length);
void fill_string(char *form, char *string, int index, int length, int *shift);
void adjust_form(char *form, int *indexes, int *length, int size, int formlen,
int totalshift);
int get_number(char *ptr, int *value);
int parse_query_string(char *iptr, int max_cnt, char *txn_chars,
value_index_entry *txn_vals);

#define mod_neworder_cmd(rec, w_id, ld_id)
sendform_neworderinput(rec, w_id, ld_id)
#define mod_delivery_cmd(rec, w_id, ld_id)
sendform_deliveryinput(rec, w_id, ld_id)
#define mod_payment_cmd(rec, w_id, ld_id)
sendform_paymentinput(rec, w_id, ld_id)
#define mod_orderstatus_cmd(rec, w_id, ld_id)
sendform_orderstatusinput(rec, w_id, ld_id)
#define mod_stocklevel_cmd(rec, w_id, ld_id)
sendform_stocklevelinput(rec, w_id, ld_id)

/*  -------------------------------------------------------------------------------
   The following defines the form layout of the different screens (forms).

   NAME=1 - Command.

    VALUE = NewOrder      - neworder bring out new order input form
           Delivery      - delivery bring out delivery input form
           OrderStatus   - order status bring out order status input form
           Payment       - payment bring out payment input form
           StockLevel    - stock level bring out stock level input form
           Menu          - display main menu
           Process       - perform the specified transaction after providing input
           Begin         - send wid and did

   NAME=2 - Form Type.

     VALUE = d,n,p,s,o [D,N,P,S,O] output/input. Plus terminal ID.
          = W logon
          = M main menu

   Delivery
      3 - district number.
    Order Status
       3 - district number.
       4 - customer id.
       5 - customer last name.


    Payment
       3 - district number.
       4 - customer id.
       5 - customer warehouse.
       6 - customer district.
       7 - name
       8 - amount paid

    Stock Level
       3 - stock level threshould

    New Order
       3 - district number.
       4 - customer number.
 ------------------------------------------------------------------------------------- */

#define TRANSACTION_MENU \
"<HR>"\
"<INPUT TYPE=submit NAME=0 VALUE=NewOrder>"\
"<INPUT TYPE=submit NAME=0 VALUE=Payment>"\
"<INPUT TYPE=submit NAME=0 VALUE=Delivery>"\
"<INPUT TYPE=submit NAME=0 VALUE=StockLevel>"\
"<INPUT TYPE=submit NAME=0 VALUE=OrderStatus>"\
"<INPUT TYPE=submit NAME=0 VALUE=Exit>"


/* static char WelcomeForm [] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=2 VALUE=B000>"
"%s. Please provide your warehouse ID and district ID.<BR>"
"Warehouse ID <INPUT NAME=3 SIZE=7><BR>"
"District  ID <INPUT NAME=4 SIZE=2><BR>"
"<HR>"
"<INPUT TYPE=submit NAME=1 VALUE=Begin>"
"</FORM></BODY>"; */
static char WelcomeForm [] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=3 VALUE=W000>"
"%s. Please provide your warehouse ID and district ID.<BR>"
"Warehouse ID <INPUT NAME=4 SIZE=7><BR>"
"District  ID <INPUT NAME=5 SIZE=2><BR>"
"<HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Begin>"
"</FORM></BODY>";


static char FormHeader [] =
"<BODY><FORM ACTION=%s METHOD=GET>";

#define FORM_BEGIN   "<BODY><FORM ACTION=%s
METHOD=GET>"
#define FORM_END     "</FORM></BODY>"
#define FORM_SUBMIT  "<INPUT TYPE=submit NAME=0
VALUE=Process>"
#define FORM_MENU    "<INPUT TYPE=submit NAME=0
VALUE=Menu>"

static char MainForm [] =
FORM_BEGIN
"<INPUT TYPE=hidden NAME=3 VALUE=M%07d>"
"%60s<BR>"
"Please Select the Next Transaction.<BR>"
TRANSACTION_MENU
FORM_END;

static char ErrorForm [] =
FORM_BEGIN
"<INPUT TYPE=hidden NAME=3 VALUE=e%06d>"
"Error: %d %d %40s %s<BR>"
TRANSACTION_MENU
FORM_END;

/*
static char ErrorForm [] =
FORM_BEGIN
"<INPUT TYPE=hidden NAME=3 VALUE=e%06d>"
"Error: %d (%s): %s<BR>"
TRANSACTION_MENU
FORM_END;
*/
#define DE_EXTRA_ID                                    0
#define DE_INPUT_DID         DE_EXTRA_ID + 1
#define DE_INPUT_QTIME        DE_INPUT_DID + 1
#define DE_INPUT_MAX         DE_INPUT_QTIME + 1

#define OS_INPUT_DID         0
#define OS_INPUT_CID         OS_INPUT_DID + 1
#define OS_INPUT_NAME         OS_INPUT_CID + 1
#define OS_INPUT_MAX         OS_INPUT_NAME + 1

#define PA_INPUT_DID         0
#define PA_INPUT_CID         PA_INPUT_DID + 1
#define PA_INPUT_CWID         PA_INPUT_CID + 1
#define PA_INPUT_CDID        PA_INPUT_CWID + 1
#define PA_INPUT_NAME         PA_INPUT_CDID + 1
#define PA_INPUT_AMT         PA_INPUT_NAME + 1
#define PA_INPUT_MAX         PA_INPUT_AMT + 1

#define SL_INPUT_THRESHOLD   0
#define SL_INPUT_MAX         SL_INPUT_THRESHOLD + 1

#define NO_INPUT_DID         0
#define NO_INPUT_CID         NO_INPUT_DID + 1
#define NO_INPUT_SPW00        NO_INPUT_CID + 1
#define NO_INPUT_IID00       NO_INPUT_SPW00 + 1
#define NO_INPUT_QTY00        NO_INPUT_IID00 + 1
#define NO_INPUT_SPW01        NO_INPUT_QTY00 + 1
#define NO_INPUT_IID01       NO_INPUT_SPW01 + 1
#define NO_INPUT_QTY01        NO_INPUT_IID01 + 1
#define NO_INPUT_SPW02        NO_INPUT_QTY01 + 1
#define NO_INPUT_IID02       NO_INPUT_SPW02 + 1
#define NO_INPUT_QTY02        NO_INPUT_IID02 + 1
#define NO_INPUT_SPW03        NO_INPUT_QTY02 + 1
#define NO_INPUT_IID03       NO_INPUT_SPW03 + 1
#define NO_INPUT_QTY03        NO_INPUT_IID03 + 1
#define NO_INPUT_SPW04        NO_INPUT_QTY03 + 1
#define NO_INPUT_IID04       NO_INPUT_SPW04 + 1
#define NO_INPUT_QTY04        NO_INPUT_IID04 + 1
#define NO_INPUT_SPW05        NO_INPUT_QTY04 + 1
#define NO_INPUT_IID05       NO_INPUT_SPW05 + 1
#define NO_INPUT_QTY05        NO_INPUT_IID05 + 1
#define NO_INPUT_SPW06        NO_INPUT_QTY05 + 1
#define NO_INPUT_IID06       NO_INPUT_SPW06 + 1
#define NO_INPUT_QTY06        NO_INPUT_IID06 + 1
#define NO_INPUT_SPW07        NO_INPUT_QTY06 + 1
#define NO_INPUT_IID07       NO_INPUT_SPW07 + 1
#define NO_INPUT_QTY07        NO_INPUT_IID07 + 1
#define NO_INPUT_SPW08        NO_INPUT_QTY07 + 1
#define NO_INPUT_IID08       NO_INPUT_SPW08 + 1
#define NO_INPUT_QTY08        NO_INPUT_IID08 + 1
#define NO_INPUT_SPW09        NO_INPUT_QTY08 + 1
#define NO_INPUT_IID09       NO_INPUT_SPW09 + 1
#define NO_INPUT_QTY09        NO_INPUT_IID09 + 1
#define NO_INPUT_SPW10        NO_INPUT_QTY09 + 1
```

```
#define NO_INPUT_IID10       NO_INPUT_SPW10 + 1
#define NO_INPUT_QTY10       NO_INPUT_IID10 + 1
#define NO_INPUT_SPW11       NO_INPUT_QTY10 + 1
#define NO_INPUT_IID11       NO_INPUT_SPW11 + 1
#define NO_INPUT_QTY11       NO_INPUT_IID11 + 1
#define NO_INPUT_SPW12       NO_INPUT_QTY11 + 1
#define NO_INPUT_IID12       NO_INPUT_SPW12 + 1
#define NO_INPUT_QTY12       NO_INPUT_IID12 + 1
#define NO_INPUT_SPW13       NO_INPUT_QTY12 + 1
#define NO_INPUT_IID13       NO_INPUT_SPW13 + 1
#define NO_INPUT_QTY13       NO_INPUT_IID13 + 1
#define NO_INPUT_SPW14       NO_INPUT_QTY13 + 1
#define NO_INPUT_IID14       NO_INPUT_SPW14 + 1
#define NO_INPUT_QTY14       NO_INPUT_IID14 + 1
#define NO_INPUT_MAX         NO_INPUT_QTY14 + 1

#define DE_TERMID            0
#define DE_WID               DE_TERMID+1
#define DE_CARID             DE_WID+1
#define DE_FORMINDEX_SIZE    DE_CARID+1

static char DeliveryFormInput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=D#######>"
"<INPUT TYPE=hidden NAME=6 VALUE=0>"
"<PRE>                              Delivery <BR>"
"Warehouse: ####### <BR><BR>"
"Carrier Number: <INPUT NAME=7 SIZE=2><BR><BR>"
"Execution Status: <BR></PRE>"
FORM_MENU
FORM_SUBMIT
FORM_END;

static char DeliveryFormOutput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=d#######>"
"<PRE>                              Delivery <BR>"
"Warehouse: ####### <BR><BR>"
"Carrier Number: ##<BR><BR>"
"Execution Status: Delivery has been queued. <BR></PRE>"
TRANSACTION_MENU
FORM_END;

#define OS_TERMID            0
#define OS_WID               OS_TERMID+1
#define OS_DID               OS_WID+1
#define OS_CID               OS_DID+1
#define OS_FIRST             OS_CID+1
#define OS_MIDDLE            OS_FIRST+1
#define OS_LAST              OS_MIDDLE+1
#define OS_CBALANCE          OS_LAST+1
#define OS_OID               OS_CBALANCE+1
#define OS_ENTRY_DATE        OS_OID+1
#define OS_CARID             OS_ENTRY_DATE+1
#define OS_SUPW              OS_CARID+1
#define OS_ITEMID            OS_SUPW+1
#define OS_QTY               OS_ITEMID+1
#define OS_AMOUNT            OS_QTY+1
#define OS_DELDATE           OS_AMOUNT+1
#define OS_FORMINDEX_SIZE    OS_DELDATE+(14*5)+1


static char OrderStatusInput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=O#######>"
"<PRE>                              Order-Status <BR>"
"Warehouse: #######  District: <INPUT NAME=8 SIZE=2><BR>"
"Customer: <INPUT NAME=9 SIZE=4> Name: <INPUT NAME=Y
SIZE=23> <BR>"
"Cust-Balance:<BR><BR>"
"Order-Number:          Entry-Data:                Carrier-Number:<BR>"
"Supply-W    Item-ID   QTY    Amount      Delivery-
Data<BR></PRE><HR>"
FORM_MENU
FORM_SUBMIT
FORM_END;

static char OrderStatusOutput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=o#######>"
"<PRE>                              Order Status <BR>"
"Warehouse: #######  District: ##<BR>"
"Customer: ####    Name: ################ ##
################<BR>"
"Cust-Balance: $#########<BR><BR>"
"Order-Number: ########    Entry-Date: ###################    Carrier-
Number: ##<BR>"
"Supply-W    Item-ID   QTY    Amount      Delivery-Data<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"  ######      ######    ##    $########      ##########<BR>"
"</PRE>"
TRANSACTION_MENU
FORM_END;

#define PA_INPUT_TERMID          0
#define PA_INPUT_WID             PA_TERMID+1
#define PA_INPUT_FORMINDEX_SIZE  PA_INPUT_WID+1

#define PA_TERMID            0
#define PA_DATE              PA_TERMID+1
#define PA_WID               PA_DATE+1
#define PA_DID               PA_WID+1
#define PA_WST1              PA_DID+1
#define PA_DST1              PA_WST1+1
#define PA_WST2              PA_DST1+1
#define PA_DST2              PA_WST2+1
#define PA_WCITY             PA_DST2+1
#define PA_WSTATE            PA_WCITY+1
#define PA_WZIP              PA_WSTATE+1
#define PA_DCITY             PA_WZIP+1
#define PA_DSTATE            PA_DCITY+1
#define PA_DZIP              PA_DSTATE+1
#define PA_CID               PA_DZIP+1
#define PA_CWARE             PA_CID+1
#define PA_CDIST             PA_CWARE+1
#define PA_CFIRST            PA_CDIST+1
#define PA_CMIDDLE           PA_CFIRST+1
#define PA_CLAST             PA_CMIDDLE+1
#define PA_SINCE             PA_CLAST+1
#define PA_CST1              PA_SINCE+1
#define PA_CREDIT            PA_CST1+1
#define PA_CST2              PA_CREDIT+1
#define PA_DISC              PA_CST2+1
#define PA_CCITY             PA_DISC+1
#define PA_CSTATE            PA_CCITY+1
#define PA_CZIP              PA_CSTATE+1
#define PA_CPHONE            PA_CZIP+1
#define PA_AMOUNT            PA_CPHONE+1
#define PA_CBAL              PA_AMOUNT+1
#define PA_LIMIT             PA_CBAL+1
#define PA_CUSTDATA          PA_LIMIT+1
```

```
#define PA_FORMINDEX_SIZE        PA_CUSTDATA+3+1

static char PaymentInput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=P#######>"
"<PRE>                               Payment<BR>"
"Date: <BR><BR>"
"Warehouse: #######                     District: <INPUT NAME=8
SIZE=2><BR>"
"<BR><BR><BR><BR>"
"Customer: <INPUT NAME=9 SIZE=4>"
"Cust-Warehouse: <INPUT NAME=Z SIZE=7>"
"Cust-District: <INPUT NAME=v SIZE=2><BR>"
"Name: <INPUT NAME=Y SIZE=16>                       Since: <BR>"
"                                          Credit: <BR>"
"                                          Disc: <BR>"
"                                          Phone: <BR><BR>"
"Amount Paid:         $<INPUT NAME=w SIZE=7>        New Cust
Balance: <BR>"
"Credit limit:<BR><BR>Cust-Data:
<BR><BR><BR><BR></PRE><HR>"
FORM_MENU
FORM_SUBMIT
FORM_END;

static char PaymentOutput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=p#######>"
"<PRE>                               Payment<BR>"
"Date: ###################<BR><BR>"
"Warehouse: #######                     District: ##<BR>"
"####################                      ####################<BR>"
"####################                      ####################<BR>"
"#################### ## #########          ####################
## #########<BR>"
"<BR><BR>"
"Customer: #### Cust-Warehouse: ####### Cust-District: ##<BR>"
"Name:  ############### ## #################       Since:
##########<BR>"
"       ####################                    Credit: ##<BR>"
"       ####################                    %Disc:  ####<BR>"
"       #################### ## #########         Phone:
###################<BR>"
"<BR><BR>"
"Amount Paid:      $######     New Cust Balance:
$##############<BR>"
"Credit Limit:   $##############<BR><BR>"
"Cust-Data:
##################################################<BR>"
"
##################################################<BR>"
"
##################################################<BR>"
"
##################################################<BR>"
"</PRE>"
TRANSACTION_MENU
FORM_END;

#define SL_TERMID                0
#define SL_WID                  SL_TERMID+1
#define SL_DID                  SL_WID+1
#define SL_THRESHOLD            SL_DID+1
#define SL_LOWSTOCK             SL_THRESHOLD+1
#define SL_FORMINDXE_SIZE       SL_LOWSTOCK

static char StockLevelInput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=S#######>"
"<PRE>                               Stock-Level<BR>"
"Warehouse: #######  District ##<BR><BR>"
"Stock Level Threshold: <INPUT NAME=x SIZE=2><BR><BR>"
"low stock:     <BR></PRE><HR>"
FORM_MENU
FORM_SUBMIT
FORM_END;

static char StockLevelOutput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=s#######>"
"<PRE>                               Stock Level<BR>"
"Warehouse: #######  District ##<BR><BR>"
"Stock Level Threshold: ##<BR><BR>"
"low stock:  ### <BR></PRE><HR>"
TRANSACTION_MENU
FORM_END;

#define NO_TERMID            0
#define NO_WID               NO_TERMID+1
#define NO_DID               NO_WID+1
#define NO_DATE              NO_DID+1
#define NO_CID               NO_DATE+1
#define NO_NAME              NO_CID+1
#define NO_CREDIT            NO_NAME+1
#define NO_DISC              NO_CREDIT+1
#define NO_OID               NO_DISC+1
#define NO_LINES             NO_OID+1
#define NO_WTAX              NO_LINES+1
#define NO_DTAX              NO_WTAX+1
#define NO_SUPPW             NO_DTAX+1
#define NO_ITEMID            NO_SUPPW+1
#define NO_INAME             NO_ITEMID+1
#define NO_QTY               NO_INAME+1
#define NO_STOCK             NO_QTY+1
#define NO_BRAND             NO_STOCK+1
#define NO_PRICE             NO_BRAND+1
#define NO_AMOUNT            NO_PRICE+1
#define NO_STATUS            NO_AMOUNT + 14*8 + 1
#define NO_TOTAL             NO_STATUS+1
#define NO_FORMINDEX_SIZE    NO_TOTAL+1

static char NewOrderInput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=N#######>"
"<PRE>                               New Order<BR>"
"Warehouse: #######   District: <INPUT NAME=8 SIZE=2>
Date:<BR>"
"Customer: <INPUT NAME=9 size=4>  Name:             Credit:
%Disc:<BR>"
"Order Number:         Number of Lines:           W_tax:
D_tax:<BR><BR>"
" Supp_W   Item-Id  Item Name              Qty   Stock  B/G  Price
Amount<BR>"
"<INPUT NAME=A SIZE=6> <INPUT NAME=B SIZE=7><INPUT
NAME=C SIZE=2><BR>"
"<INPUT NAME=D SIZE=6> <INPUT NAME=E SIZE=7><INPUT
NAME=F SIZE=2><BR>"
"<INPUT NAME=G SIZE=6> <INPUT NAME=H SIZE=7><INPUT
NAME=I SIZE=2><BR>"
"<INPUT NAME=J SIZE=6> <INPUT NAME=K SIZE=7><INPUT
NAME=L SIZE=2><BR>"
"<INPUT NAME=M SIZE=6> <INPUT NAME=N SIZE=7><INPUT
NAME=O SIZE=2><BR>"
"<INPUT NAME=P SIZE=6> <INPUT NAME=Q SIZE=7><INPUT
NAME=R SIZE=2><BR>"
"<INPUT NAME=S SIZE=6> <INPUT NAME=T SIZE=7><INPUT
NAME=U SIZE=2><BR>"
"<INPUT NAME=V SIZE=6> <INPUT NAME=W SIZE=7><INPUT
NAME=X SIZE=2><BR>"
"<INPUT NAME=a SIZE=6> <INPUT NAME=b SIZE=7><INPUT
NAME=c SIZE=2><BR>"
"<INPUT NAME=d SIZE=6> <INPUT NAME=e SIZE=7><INPUT
NAME=f SIZE=2><BR>"
"<INPUT NAME=g SIZE=6> <INPUT NAME=h SIZE=7><INPUT
NAME=i SIZE=2><BR>"
```

```
"<INPUT NAME=j SIZE=6> <INPUT NAME=k SIZE=7><INPUT
NAME=l SIZE=2><BR>"
"<INPUT NAME=m SIZE=6> <INPUT NAME=n SIZE=7><INPUT
NAME=o SIZE=2><BR>"
"<INPUT NAME=p SIZE=6> <INPUT NAME=q SIZE=7><INPUT
NAME=r SIZE=2><BR>"
"<INPUT NAME=s SIZE=6> <INPUT NAME=t SIZE=7><INPUT
NAME=u SIZE=2><BR>"
"Execution Status:                        Total:<BR></PRE><HR>"
FORM_MENU
FORM_SUBMIT
FORM_END;

static char NewOrderOutput_Template [] =
"<INPUT TYPE=hidden NAME=3 VALUE=n#######>"
"<PRE>                        New Order<BR>"
"Warehouse: #######   District: ##                  Date:
###################<BR>"
"Customer:    ####   Name: ################   Credit: ##  %Disc:
##### <BR>"
"Order Number:  ########    Number of Lines: ##       W_tax: #####
D_tax: ##### <BR>"
"<BR>"
" Supp_W   Item-Id   Item Name               Qty  Stock  B/G  Price
Amount<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
" #######   ######   ########################   ##    ##    #
$###### $#######<BR>"
"Execution Status:   ########################                Total:
$#######<BR>"
"</PRE>"
TRANSACTION_MENU
FORM_END;
```

- - - - - - - - - - - - - - - - - -
modtpcc.h
- - - - - - - - - - - - - - - - - -

```
#include "..\DBConnection\mod_tpcc.h"
#include "..\DBConnection\tpcc_struct.h"
#include "..\DBConnection\mod_tpcc_error.h"
#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>

#define allocate_last_form(form, pool) \
(form)=(char *)((pool)->form_template_storage + \
 (Maxterms - 1) * (pool)->form_template_length)


#define MAXLEN 100
#define Default_DBConnections "20"
#define Default_Maxterms "100"
#define Default_DeliveryQueues "500"
#define Default_DeliveryThreads "50"
#define Default_StartTerm "1"
#define LogName "log\\modtpcc.log"
#define InitName "DBInit.ini"
#define DllName "DBConnection.dll"
#define mod_name "/tpcc/modtpcc.dll"


typedef struct _DelQueue_info {
        _DelQueue_info *Next;
        T_delivery_data *pdata;
        HANDLE queue_lock;
} DelQueue_info;


/****************************************************************
**********************************
* global functions                                              *
***************************************************************
**********************************/

//void userlog (char *, ...);
void readInit(char *, char *, char *);
void allocateMemoryPool();
int initDelQueue();
int deleteDelQueue();
void endDeliveryThread(int);
void initDeliveryThread(void *);
DelQueue_info *DequeueDel();
void EnqueueDel(DelQueue_info *);
void addFreeDelQueue(DelQueue_info *);
DelQueue_info *findFreeDelQueue();

int parse_neworder_query(char *ptr, T_neworder_data *pdata);
int parse_payment_query(char *ptr, T_payment_data *pdata);
int parse_delivery_query(char *ptr, T_delivery_data *pdata);
int parse_orderstatus_query(char *ptr, T_orderstatus_data *pdata);
int parse_stocklevel_query(char *ptr, T_stocklevel_data *pdata);

int sendform_neworderoutput(int status, T_neworder_data *pdata);
int sendform_paymentoutput(int status, T_payment_data *pdata);
int sendform_orderstatusoutput(int status, T_orderstatus_data *pdata);
int sendform_deliveryoutput(int status, T_delivery_data *pdata);
int sendform_stockleveloutput(int status, T_stocklevel_data *pdata);

extern int (FAR * mod_tpcc_neworder)(T_neworder_data *);
extern int (FAR * mod_tpcc_payment)(T_payment_data *);
extern int (FAR * mod_tpcc_delivery)(T_delivery_data *, int);
extern int (FAR * mod_tpcc_orderstatus)(T_orderstatus_data *);
extern int (FAR * mod_tpcc_stocklevel)(T_stocklevel_data *);
extern void (FAR *userlog)(char * str, ...);
extern void (FAR *initDelLog)(int);
extern void (FAR *endDelLog)(int);


/****************************************************************
**********************************
* global variables                                              *
```

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

DWORD TlsPointer;
char DllPath[MAXLEN];
char LogFile[MAXLEN];
char InitFile[MAXLEN];
char DllFile[MAXLEN];
char origin[MAXLEN];
CRITICAL_SECTION critical_initDelQueue;
CRITICAL_SECTION critical_memory;
CRITICAL_SECTION critical_DelQueue_free;
CRITICAL_SECTION critical_DelQueue_work;
HANDLE waitAvailableDelQueue;
HANDLE waitDelWork;
HANDLE DelThreadRunning;
HINSTANCE dllinstance;
int useddel=0;
int DBConnections;
int Maxterms;
int DeliveryQueues;
int DeliveryThreads;
int modtpcc_ready=0;
int memory_ready=0;
int queue_ready=0;
int DeliveryThreadstop=0;
int StartTerm=1;
DelQueue_info *DelQueue_begin = NULL;
DelQueue_info *DelQueue_end = NULL;
DelQueue_info *DelQueue_free = NULL;

static form_index_entry
form_index[POOL_TYPE_TXN_MAX][TXN_TYPE_MAX][MAX_FOR
M_INDEX];
static form_template_pool
txn_global_pool[POOL_TYPE_TXN_MAX][TXN_TYPE_MAX];
static form_template_pool txn_data_pool;
static form_template_pool resp_global_pool;


char delivery_chars [] = {'6', '7'};
char orderstatus_chars [] = {'8', '9', 'Y'};
char payment_chars [] = {'8', '9', 'Z', 'v', 'Y', 'w'};
char stocklevel_chars [] = {'x'};
char neworder_chars [] = {'8', '9',
'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
'S', 'T', 'U', 'V', 'W', 'X', 'a', 'b', 'c',
'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u'};
```

- - - - - - - - - - - - - - - - - -
StdAfx.cpp
- - - - - - - - - - - - - - - - - -

```
// stdafx.cpp : source file that includes just the standard includes
//		DBConnection.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
```

- - - - - - - - - - - - - - - - - -
StdAfx.h
- - - - - - - - - - - - - - - - - -

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

#if
!defined(AFX_STDAFX_H__1D53560F_AAD5_4CEE_A8CC_651C9688
A6DF__INCLUDED_)
#define
AFX_STDAFX_H__1D53560F_AAD5_4CEE_A8CC_651C9688A6DF__
INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


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

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <atlbase.h>


// TODO: reference additional headers your program requires here

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_STDAFX_H__1D53560F_AAD5_4CEE_A8CC_651C9688
A6DF__INCLUDED_)
```

- - - - - - - - - - - - - - - - - -
tpccflags.h
- - - - - - - - - - - - - - - - - -

```
//#define USE_IEEE_NUMBER
```

- - - - - - - - - - - - - - - - - -
tpccpl.h
- - - - - - - - - - - - - - - - - -

```
#ifndef TPCCPL_H
#define TPCCPL_H

//#include "tpcc.h"

#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>
#include <time.h>
#include <io.h>
#include "tpccflags.h"

#ifdef TUX
#define DELRT 5.0
#else
#define DELRT 80.0
#endif


#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
*/

/* 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-mm-yyyy.hh24:mi:ss"
#define SHORTDATE "dd-mm-yyyy"


#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 OCIERROR(errp,function)\
        ocierror(__FILE__,__LINE__,(errp),(function));

#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
    ocierror(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
    ocierror(__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__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
    ocierror(__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,cb
f_data) \
    ocierror(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
    ocierror(__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__,__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__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
    ocierror(__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__,__LINE__, (errp), \

OCIHandleAlloc((stmp),&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
    ocierror(__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 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

/* bind in/out for plsql without indicator and rcode */
#define OCIBNDPL(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen) \
   DISCARD ocierror(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
   DISCARD ocierror(__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/out for plsql arrays witout indicator and rcode */
#define
OCIBNDPLA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen,ms,cu) \
      DISCARD ocierror(__FILE__,__LINE__, (errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0));\
      DISCARD ocierror(__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));

#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(__FILE__,__LINE__,(errp), \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                                              (dvoid**)0));\
      ocierror(__FILE__,__LINE__,(errp), \
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),
(progvl),(ftype),\
                          (indp),NULL,NULL, OCI_DYNAMIC_FETCH));\
      ocierror(__FILE__,__LINE__,(errp), \
      OCIDefineDynamic((dfnp),(errp),(ctxp),(cbf_data)));


#endif
```

```
- - - - - - - - - - - - - - - - - -
tpcc_struct.h
- - - - - - - - - - - - - - - - - -
/* Copyright (c) 2004, Oracle Corporation.  All rights reserved.  */

/*
   NAME
     tpcc_struct.h - <one-line expansion of the name>

   DESCRIPTION
     <short description of facility this file declares/defines>

   RELATED DOCUMENTS
     <note any documents related to this facility>

   EXPORT FUNCTION(S)
     <external functions declared for use outside package - one-line
descriptions>

   INTERNAL FUNCTION(S)
     <other external functions declared - one-line descriptions>

   EXAMPLES

   NOTES
     <other useful comments, qualifications, etc.>

   MODIFIED   (MM/DD/YY)
   xnie        02/09/04 - add status field to carry error status
   shuang      01/22/04 - shuang_rte
   shuang      01/21/04 - Creation

*/

#define MAX_ORDERLINE  15
#define SMALL_BUF_SIZE 32

#define TXN_COMMON_DATA \
   int w_id; \
   int ld_id; \
   int txn_status; \
   int db_status; \
   void *context


struct T_connect_data
{
   TXN_COMMON_DATA;
};
typedef struct T_connect_data T_connect_data;

struct T_date
{
   char DateString[20];
};
typedef struct T_date T_date;

struct T_delivery_data
{
   TXN_COMMON_DATA;
   SYSTEMTIME      enqueue_date_time;
   DWORD           enqueue_time;
   DWORD           complete_time;
   int             o_carrier_id;
   int             o_id[10];
};
typedef struct T_delivery_data T_delivery_data;

struct T_orderline
{
   int    ol_i_id;
   int    ol_supply_w_id;
   int    ol_quantity;
   char   i_name[25];
   int    s_quantity;
   char   b_g[2];
   double i_price;
   double ol_amount;
};
typedef struct T_orderline T_orderline;

struct T_neworder_data
{
   TXN_COMMON_DATA;
   int     d_id;
   int     c_id;
   int     o_ol_cnt;
   int     o_all_local;
   T_orderline o_orderline[MAX_ORDERLINE];
   T_date  o_entry_d;
   char    c_last[17];
   char    c_credit[3];
   double  c_discount;
   double  w_tax;
   double  d_tax;
   int     o_id;
   double  total_amount;
   int     status;
};
typedef struct T_neworder_data T_neworder_data;

struct T_stocklevel_data
{
   TXN_COMMON_DATA;
   int    threshold;
   int    low_stock;
};
typedef struct T_stocklevel_data T_stocklevel_data;

struct T_orderline_status
{
   int    ol_supply_w_id;
   int    ol_i_id;
   int    ol_quantity;
   double ol_amount;
   T_date ol_delivery_d;
};
typedef struct T_orderline_status T_orderline_status;

struct T_orderstatus_data
{
   TXN_COMMON_DATA;
   int     by_last_name;
   int     d_id;
   int     c_id;
   char    c_last[17];
   char    c_first[17];
   char    c_middle[3];
   double  c_balance;
   int     o_id;
   T_date  o_entry_d;
   int     o_carrier_id;
   int     o_ol_cnt;
   T_orderline_status  o_orderline[MAX_ORDERLINE];
};
typedef struct T_orderstatus_data T_orderstatus_data;

struct T_payment_data
```

```
{
   TXN_COMMON_DATA;
   int     by_last_name;
   int     d_id;
   int     c_id;
   char    c_last[17];
   int     c_w_id;
   int     c_d_id;
   double  h_amount;
   T_date  h_date;
   char    w_street_1[21];
   char    w_street_2[21];
   char    w_city[21];
   char    w_state[3];
   char    w_zip[10];
   char    d_street_1[21];
   char    d_street_2[21];
   char    d_city[21];
   char    d_state[3];
   char    d_zip[10];
   char    c_first[17];
   char    c_middle[3];
   char    c_street_1[21];
   char    c_street_2[21];
   char    c_city[21];
   char    c_state[3];
   char    c_zip[10];
   char    c_phone[17];
   T_date  c_since;
   char    c_credit[3];
   double  c_credit_lim;
   double  c_discount;
   double  c_balance;
   char    c_data[201];
};
typedef struct T_payment_data T_payment_data;


struct T_transaction_data
{
   int txn_type;
   union {
      T_delivery_data delivery_data;
      T_payment_data payment_data;
      T_neworder_data neworder_data;
      T_stocklevel_data stocklevel_data;
      T_orderstatus_data orderstatus_data;
   } txn_data;

};
typedef struct T_transaction_data T_transaction_data;


struct T_login_data
{
   TXN_COMMON_DATA;
   char   server[SMALL_BUF_SIZE];
   char   database[SMALL_BUF_SIZE];
   char   user[SMALL_BUF_SIZE];
   char   password[SMALL_BUF_SIZE];
   char   application[SMALL_BUF_SIZE];
};
typedef struct T_login_data T_login_data;
```

- - - - - - - - - - - - - - - - - -
tpccstruct.h
- - - - - - - - - - - - - - - - - -

```
#define NITEMS 15
#define ROWIDLEN 20
#define OCIROWLEN 20

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
  double s_remote[NITEMS];
#else
  int s_remote[NITEMS];
#endif
  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;
  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;



#define NDISTS 10
#define ROWIDLEN 20


struct delctx {
   sb2 del_o_id_ind[NDISTS];
   sb2 d_id_ind[NDISTS];
   sb2 c_id_ind[NDISTS];
   sb2 del_date_ind[NDISTS];
   sb2 carrier_id_ind[NDISTS];
   sb2 amt_ind[NDISTS];

   ub4 del_o_id_len[NDISTS];
   ub4 c_id_len[NDISTS];
   int oid_ctx;
   int cid_ctx;
   OCIBind *olamt_bp;

   ub2 w_id_len[NDISTS];
   ub2 d_id_len[NDISTS];
   ub2 del_date_len[NDISTS];
   ub2 carrier_id_len[NDISTS];
   ub2 amt_len[NDISTS];

   ub2 del_o_id_rcode[NDISTS];
   ub2 cons_rcode[NDISTS];
   ub2 w_id_rcode[NDISTS];
   ub2 d_id_rcode[NDISTS];
   ub2 c_id_rcode[NDISTS];
   ub2 del_date_rcode[NDISTS];
   ub2 carrier_id_rcode[NDISTS];
   ub2 amt_rcode[NDISTS];

   int del_o_id[NDISTS];
   int del_d_id[NDISTS];
   int cons[NDISTS];
   int w_id[NDISTS];
   int d_id[NDISTS];
   int c_id[NDISTS];
   int carrier_id[NDISTS];
   int amt[NDISTS];
   ub4 del_o_id_rcnt;
   int retry;
   OCIRowid *no_rowid_ptr[NDISTS];
   OCIRowid *o_rowid_ptr[NDISTS];
   OCIDate del_date[NDISTS];
   OCIStmt *curd0;
   OCIStmt *curd1;
   OCIStmt *curd2;
   OCIStmt *curd3;
   OCIStmt *curd4;
   OCIStmt *curd5;
   OCIStmt *curd6;
   OCIStmt *curdtest;
   OCIBind *w_id_bp;
   OCIBind *w_id_bp3;
   OCIBind *w_id_bp4;
   OCIBind *w_id_bp5;
   OCIBind *w_id_bp6;
   OCIBind *d_id_bp;
   OCIBind *d_id_bp3;
   OCIBind *d_id_bp4;
   OCIBind *d_id_bp6;
   OCIBind *o_id_bp;
   OCIBind *cr_date_bp;
   OCIBind *c_id_bp;
   OCIBind *c_id_bp3;
   OCIBind *no_rowid_bp;
   OCIBind *carrier_id_bp;
   OCIBind *o_rowid_bp;
   OCIBind *del_o_id_bp;
   OCIBind *del_o_id_bp3;
   OCIBind *amt_bp;
   OCIBind *bstr1_bp[10];
   OCIBind *bstr2_bp[10];
   OCIBind *retry_bp;
   OCIDefine *inum_dp;
   OCIDefine *d_id_dp;
   OCIDefine *del_o_id_dp;
   OCIDefine *no_rowid_dp;
   OCIDefine *c_id_dp;
   OCIDefine *o_rowid_dp;
   OCIDefine *cons_dp;
   OCIDefine *amt_dp;

   int norow;
};

typedef struct delctx delctx;
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
   double 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;

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;


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;


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;



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;


/* New order */

struct newinstruct {
   int w_id;
   int d_id;
   int c_id;
   int ol_i_id[15];
   int ol_supply_w_id[15];
   int ol_quantity[15];
};

struct newoutstruct {
   int terror;
   int o_id;
   int o_ol_cnt;
   char c_last[17];
   char c_credit[3];
   double c_discount;
   double w_tax;
   double d_tax;
   char o_entry_d[20];
   double total_amount;
   char i_name[15][25];
   int s_quantity[15];
   char brand_generic[15];
   double i_price[15];
   double ol_amount[15];
   char status[26];
   int retry;
};

struct newstruct {
   struct newinstruct newin;
   struct newoutstruct newout;
};


/* Payment */

struct payinstruct {
   int w_id;
   int d_id;
   int c_w_id;
   int c_d_id;
   int c_id;
   int bylastname;
   int h_amount;
   char c_last[17];
};

struct payoutstruct {
   int terror;
   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];
   char c_since[11];
   char c_credit[3];
   double c_credit_lim;
   double  c_discount;
   double c_balance;
   char c_data[201];
   char h_date[20];
   int retry;
};

struct paystruct {
   struct payinstruct payin;
   struct payoutstruct payout;
};


/* Order status */

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;
   char o_entry_d[20];
   int o_carrier_id;
   int o_ol_cnt;
   int ol_supply_w_id[15];
   int ol_i_id[15];
   int ol_quantity[15];
   double ol_amount[15];
   char ol_delivery_d[15][11];
   int retry;
};

struct ordstruct {
   struct ordinstruct ordin;
   struct ordoutstruct ordout;
};


/* Delivery */

struct delinstruct {
   int w_id;
   int o_carrier_id;
   double qtime;
   int in_timing_int;
   int plsqlflag;
};

struct deloutstruct {
   int terror;
   int retry;
};

struct delstruct {
   struct delinstruct delin;
   struct deloutstruct delout;
};


/* Stock level */

struct stoinstruct {
   int w_id;
   int d_id;
   int threshold;
};
```

10 Appendix B: Database Design

```
– - - - - - - - - - - - - - - - -
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;


- - - - - - - - - - - - - - - - - -
createdb.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreatedb.sh Wed Sep 21 15:47:35
PDT 2011 */
spool createdb.log

set echo on

shutdown abort

startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxinstances 1
  datafile
   '/dev/vol_one//system_1' size 400M reuse
  logfile '/home/oracle/disks/log_2_1' size 164G reuse,
          '/home/oracle/disks/log_2_2' size 164G reuse
  sysaux datafile '/dev/vol_one//tpccaux' size 120M reuse ;




create undo tablespace undo_1 datafile
  '/dev/vol_one//roll1' size 8096M reuse blocksize 8K;

set echo off
exit sql.sqlcode


- - - - - - - - - - - - - - - - - -
createindex_icust1.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:41 PDT 2011 */
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 96
  compute statistics
  tablespace icust1_0 ;
    set echo off
    spool off
    exit sql.sqlcode;


- - - - - - - - - - - - - - - - - -
createindex_icust2.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:41 PDT 2011 */
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 )
  parallel 96
  compute statistics
  tablespace icust2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;


- - - - - - - - - - - - - - - - - -
createindex_idist.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:41 PDT 2011 */
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_iitem.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:42 PDT 2011 */
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_inord.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:43 PDT 2011 */
set timing on
  exit 0;


- - - - - - - - - - - - - - - - - -
createindex_iordl.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:42 PDT 2011 */
set timing on
  exit 0;


- - - - - - - - - - - - - - - - - -
createindex_iordr1.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:42 PDT 2011 */
set timing on
  exit 0;
```

```
- - - - - - - - - - - - - - - - - -
createindex_iordr2.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:42 PDT 2011 */
set timing on
    set sqlblanklines on
    spool createindex_iordr2.log ;
    set echo on ;
    drop index iordr2 ;

create unique index iordr2 on ordr ( o_w_id
, o_d_id
, o_c_id
, o_id )
parallel 96
pctfree 25  initrans 4
storage ( buffer_pool default  )
compute statistics
tablespace iordr2_0 ;

    set echo off
    spool off
    exit sql.sqlcode;


- - - - - - - - - - - - - - - - - -
createindex_istok.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:42 PDT 2011 */
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 96
  compute statistics
  tablespace istok_0 ;
    set echo off
    spool off
    exit sql.sqlcode;


- - - - - - - - - - - - - - - - - -
createindex_iware.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreateindex.sh Wed Sep 21
15:47:41 PDT 2011 */
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;


- - - - - - - - - - - - - - - - - -
createspacestats.sql
- - - - - - - - - - - - - - - - - -
@space_init
@space_get 1053100.32 86000
@space_rpt
spool off
exit sql.sqlcode;


- - - - - - - - - - - - - - - - - -
createstoredprocs.sql
- - - - - - - - - - - - - - - - - -
spool createstoreprocs.log
@tkvcinin.sql
spool off
exit sql.sqlcode;


- - - - - - - - - - - - - - - - - -
createtable_cust.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreatetable.sh Wed Sep 21
15:47:35 PDT 2011 */
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 2580000000
  hash is ( (c_id * ( 86000 * 10 ) + c_w_id * 10 + c_d_id) )
  size 360
  pctfree 0  initrans 3
  storage ( buffer_pool recycle ) parallel ( degree 24 )
  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/kit.4k.86kwh/scripts/buildcreatetable.sh Wed Sep 21
15:47:36 PDT 2011 */
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 860000
  hash is ( ((d_w_id * 10) + d_id) )
  size 3496
    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/kit.4k.86kwh/scripts/buildcreatetable.sh Wed Sep 21
15:47:37 PDT 2011 */
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_item.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreatetable.sh Wed Sep 21
15:47:38 PDT 2011 */
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_nord.sql
- - - - - - - - - - - - - - - - - -
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreatetable.sh Wed Sep 21
15:47:39 PDT 2011 */
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 860000
    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;
```

- - - - - - - - - - - - - - - - - -
createtable_ordl.sql
- - - - - - - - - - - - - - - - - -

```
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreatetable.sh Wed Sep 21
15:47:38 PDT 2011 */
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_ordr.sql
- - - - - - - - - - - - - - - - - -

```
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreatetable.sh Wed Sep 21
15:47:38 PDT 2011 */
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 860000
    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;
```

- - - - - - - - - - - - - - - - - -
createtable_stok.sql
- - - - - - - - - - - - - - - - - -

```
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreatetable.sh Wed Sep 21
15:47:37 PDT 2011 */
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 8600000000
  hash is ( (s_i_id * 86000 + s_w_id) )
  size 270
  pctfree 0  initrans 2 maxtrans 2
  storage ( buffer_pool keep ) parallel ( degree 24 )
  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_ware.sql
- - - - - - - - - - - - - - - - - -

```
/* created automatically by
/home/oracle/kit.4k.86kwh/scripts/buildcreatetable.sh Wed Sep 21
15:47:35 PDT 2011 */
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 86000
  hash is ( (w_id - 1) )
  size 3496
    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;
```

- - - - - - - - - - - - - - - - - -
createts.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

# AVLIET added AUTOEXTEND ON clause to tablespaces
if expr $6 = 1 > /dev/null; then
##  createsql="create temporary tablespace $1 tempfile '$2' size $3 reuse
extent management local uniform size $4;"
  createsql="create temporary tablespace $1 tempfile '$2' size $3 reuse
AUTOEXTEND ON 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 ;"
    createsql="create tablespace $1 datafile '$2' size $3 reuse
AUTOEXTEND ON 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;"
      createsql="create tablespace $1 datafile '$2' size $3 reuse
AUTOEXTEND ON 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 ;"
      createsql="create tablespace $1 datafile '$2' size $3 reuse
AUTOEXTEND ON 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 ;
!
```

- - - - - - - - - - - - - - - - - -
data-dev.sh
- - - - - - - - - - - - - - - - - -

```
lvcreate -r none -n ware_0_0 --size 500M vol_one
lvcreate -r none -n dist_0_0 --size 122M vol_one
lvcreate -r none -n dist_0_1 --size 122M vol_one
lvcreate -r none -n dist_0_2 --size 122M vol_one
lvcreate -r none -n dist_0_3 --size 122M vol_one
lvcreate -r none -n dist_0_4 --size 122M vol_one
lvcreate -r none -n dist_0_5 --size 122M vol_one
lvcreate -r none -n dist_0_6 --size 122M vol_one
lvcreate -r none -n dist_0_7 --size 122M vol_one
lvcreate -r none -n dist_0_8 --size 122M vol_one
lvcreate -r none -n dist_0_9 --size 122M vol_one
lvcreate -r none -n dist_0_10 --size 122M vol_one
lvcreate -r none -n dist_0_11 --size 122M vol_one
lvcreate -r none -n dist_0_12 --size 122M vol_one
lvcreate -r none -n dist_0_13 --size 122M vol_one
lvcreate -r none -n dist_0_14 --size 122M vol_one
lvcreate -r none -n dist_0_15 --size 122M vol_one
lvcreate -r none -n dist_0_16 --size 122M vol_one
lvcreate -r none -n dist_0_17 --size 122M vol_one
lvcreate -r none -n dist_0_18 --size 122M vol_one
lvcreate -r none -n dist_0_19 --size 122M vol_one
lvcreate -r none -n dist_0_20 --size 122M vol_one
lvcreate -r none -n dist_0_21 --size 122M vol_one
lvcreate -r none -n dist_0_22 --size 122M vol_one
lvcreate -r none -n dist_0_23 --size 122M vol_one
lvcreate -r none -n dist_0_24 --size 122M vol_one
lvcreate -r none -n dist_0_25 --size 122M vol_one
lvcreate -r none -n dist_0_26 --size 122M vol_one
lvcreate -r none -n dist_0_27 --size 122M vol_one
lvcreate -r none -n dist_0_28 --size 122M vol_one
lvcreate -r none -n dist_0_29 --size 122M vol_one
lvcreate -r none -n dist_0_30 --size 122M vol_one
lvcreate -r none -n dist_0_31 --size 122M vol_one
lvcreate -r none -n hist_0_0 --size 13562M vol_one
lvcreate -r none -n hist_0_1 --size 13562M vol_one
lvcreate -r none -n hist_0_2 --size 13562M vol_one
lvcreate -r none -n hist_0_3 --size 13562M vol_one
lvcreate -r none -n hist_0_4 --size 13562M vol_one
lvcreate -r none -n hist_0_5 --size 13562M vol_one
lvcreate -r none -n hist_0_6 --size 13562M vol_one
lvcreate -r none -n hist_0_7 --size 13562M vol_one
lvcreate -r none -n hist_0_8 --size 13562M vol_one
lvcreate -r none -n hist_0_9 --size 13562M vol_one
lvcreate -r none -n hist_0_10 --size 13562M vol_one
lvcreate -r none -n hist_0_11 --size 13562M vol_one
lvcreate -r none -n hist_0_12 --size 13562M vol_one
lvcreate -r none -n hist_0_13 --size 13562M vol_one
lvcreate -r none -n hist_0_14 --size 13562M vol_one
lvcreate -r none -n hist_0_15 --size 13562M vol_one
lvcreate -r none -n stok_0_0 --size 16092M vol_one
lvcreate -r none -n stok_0_1 --size 16092M vol_one
lvcreate -r none -n stok_0_2 --size 16092M vol_one
lvcreate -r none -n stok_0_3 --size 16092M vol_one
lvcreate -r none -n stok_0_4 --size 16092M vol_one
lvcreate -r none -n stok_0_5 --size 16092M vol_one
lvcreate -r none -n stok_0_6 --size 16092M vol_one
lvcreate -r none -n stok_0_7 --size 16092M vol_one
lvcreate -r none -n stok_0_8 --size 16092M vol_one
lvcreate -r none -n stok_0_9 --size 16092M vol_one
lvcreate -r none -n stok_0_10 --size 16092M vol_one
lvcreate -r none -n stok_0_11 --size 16092M vol_one
lvcreate -r none -n stok_0_12 --size 16092M vol_one
lvcreate -r none -n stok_0_13 --size 16092M vol_one
lvcreate -r none -n stok_0_14 --size 16092M vol_one
lvcreate -r none -n stok_0_15 --size 16092M vol_one
lvcreate -r none -n stok_0_16 --size 16092M vol_one
lvcreate -r none -n stok_0_17 --size 16092M vol_one
lvcreate -r none -n stok_0_18 --size 16092M vol_one
lvcreate -r none -n stok_0_19 --size 16092M vol_one
lvcreate -r none -n stok_0_20 --size 16092M vol_one
lvcreate -r none -n stok_0_21 --size 16092M vol_one
lvcreate -r none -n stok_0_22 --size 16092M vol_one
lvcreate -r none -n stok_0_23 --size 16092M vol_one
lvcreate -r none -n stok_0_24 --size 16092M vol_one
lvcreate -r none -n stok_0_25 --size 16092M vol_one
lvcreate -r none -n stok_0_26 --size 16092M vol_one
lvcreate -r none -n stok_0_27 --size 16092M vol_one
lvcreate -r none -n stok_0_28 --size 16092M vol_one
lvcreate -r none -n stok_0_29 --size 16092M vol_one
lvcreate -r none -n stok_0_30 --size 16092M vol_one
lvcreate -r none -n stok_0_31 --size 16092M vol_one
lvcreate -r none -n stok_0_32 --size 16092M vol_one
lvcreate -r none -n stok_0_33 --size 16092M vol_one
lvcreate -r none -n stok_0_34 --size 16092M vol_one
lvcreate -r none -n stok_0_35 --size 16092M vol_one
lvcreate -r none -n stok_0_36 --size 16092M vol_one
lvcreate -r none -n stok_0_37 --size 16092M vol_one
lvcreate -r none -n stok_0_38 --size 16092M vol_one
lvcreate -r none -n stok_0_39 --size 16092M vol_one
lvcreate -r none -n stok_0_40 --size 16092M vol_one
lvcreate -r none -n stok_0_41 --size 16092M vol_one
lvcreate -r none -n stok_0_42 --size 16092M vol_one
lvcreate -r none -n stok_0_43 --size 16092M vol_one
lvcreate -r none -n stok_0_44 --size 16092M vol_one
lvcreate -r none -n stok_0_45 --size 16092M vol_one
lvcreate -r none -n stok_0_46 --size 16092M vol_one
lvcreate -r none -n stok_0_47 --size 16092M vol_one
lvcreate -r none -n stok_0_48 --size 16092M vol_one
lvcreate -r none -n stok_0_49 --size 16092M vol_one
lvcreate -r none -n stok_0_50 --size 16092M vol_one
lvcreate -r none -n stok_0_51 --size 16092M vol_one
lvcreate -r none -n stok_0_52 --size 16092M vol_one
lvcreate -r none -n stok_0_53 --size 16092M vol_one
lvcreate -r none -n stok_0_54 --size 16092M vol_one
lvcreate -r none -n stok_0_55 --size 16092M vol_one
lvcreate -r none -n stok_0_56 --size 16092M vol_one
lvcreate -r none -n stok_0_57 --size 16092M vol_one
lvcreate -r none -n stok_0_58 --size 16092M vol_one
lvcreate -r none -n stok_0_59 --size 16092M vol_one
lvcreate -r none -n stok_0_60 --size 16092M vol_one
lvcreate -r none -n stok_0_61 --size 16092M vol_one
lvcreate -r none -n stok_0_62 --size 16092M vol_one
lvcreate -r none -n stok_0_63 --size 16092M vol_one
lvcreate -r none -n stok_0_64 --size 16092M vol_one
lvcreate -r none -n stok_0_65 --size 16092M vol_one
lvcreate -r none -n stok_0_66 --size 16092M vol_one
lvcreate -r none -n stok_0_67 --size 16092M vol_one
lvcreate -r none -n stok_0_68 --size 16092M vol_one
lvcreate -r none -n stok_0_69 --size 16092M vol_one
lvcreate -r none -n stok_0_70 --size 16092M vol_one
lvcreate -r none -n stok_0_71 --size 16092M vol_one
lvcreate -r none -n stok_0_72 --size 16092M vol_one
lvcreate -r none -n stok_0_73 --size 16092M vol_one
lvcreate -r none -n stok_0_74 --size 16092M vol_one
lvcreate -r none -n stok_0_75 --size 16092M vol_one
lvcreate -r none -n stok_0_76 --size 16092M vol_one
lvcreate -r none -n stok_0_77 --size 16092M vol_one
lvcreate -r none -n stok_0_78 --size 16092M vol_one
lvcreate -r none -n stok_0_79 --size 16092M vol_one
lvcreate -r none -n stok_0_80 --size 16092M vol_one
lvcreate -r none -n stok_0_81 --size 16092M vol_one
lvcreate -r none -n stok_0_82 --size 16092M vol_one
lvcreate -r none -n stok_0_83 --size 16092M vol_one
lvcreate -r none -n stok_0_84 --size 16092M vol_one
lvcreate -r none -n stok_0_85 --size 16092M vol_one
lvcreate -r none -n stok_0_86 --size 16092M vol_one
lvcreate -r none -n stok_0_87 --size 16092M vol_one
lvcreate -r none -n stok_0_88 --size 16092M vol_one
lvcreate -r none -n stok_0_89 --size 16092M vol_one
lvcreate -r none -n stok_0_90 --size 16092M vol_one
lvcreate -r none -n stok_0_91 --size 16092M vol_one
lvcreate -r none -n stok_0_92 --size 16092M vol_one
lvcreate -r none -n stok_0_93 --size 16092M vol_one
lvcreate -r none -n stok_0_94 --size 16092M vol_one
lvcreate -r none -n stok_0_95 --size 16092M vol_one
lvcreate -r none -n stok_0_96 --size 16092M vol_one
lvcreate -r none -n stok_0_97 --size 16092M vol_one
lvcreate -r none -n stok_0_98 --size 16092M vol_one
lvcreate -r none -n stok_0_99 --size 16092M vol_one
lvcreate -r none -n stok_0_100 --size 16092M vol_one
lvcreate -r none -n stok_0_101 --size 16092M vol_one
lvcreate -r none -n stok_0_102 --size 16092M vol_one
lvcreate -r none -n stok_0_103 --size 16092M vol_one
lvcreate -r none -n stok_0_104 --size 16092M vol_one
lvcreate -r none -n stok_0_105 --size 16092M vol_one
lvcreate -r none -n stok_0_106 --size 16092M vol_one
lvcreate -r none -n stok_0_107 --size 16092M vol_one
lvcreate -r none -n stok_0_108 --size 16092M vol_one
lvcreate -r none -n stok_0_109 --size 16092M vol_one
lvcreate -r none -n stok_0_110 --size 16092M vol_one
lvcreate -r none -n stok_0_111 --size 16092M vol_one
lvcreate -r none -n stok_0_112 --size 16092M vol_one
lvcreate -r none -n stok_0_113 --size 16092M vol_one
lvcreate -r none -n stok_0_114 --size 16092M vol_one
lvcreate -r none -n stok_0_115 --size 16092M vol_one
lvcreate -r none -n stok_0_116 --size 16092M vol_one
lvcreate -r none -n stok_0_117 --size 16092M vol_one
lvcreate -r none -n stok_0_118 --size 16092M vol_one
lvcreate -r none -n stok_0_119 --size 16092M vol_one
lvcreate -r none -n stok_0_120 --size 16092M vol_one
lvcreate -r none -n stok_0_121 --size 16092M vol_one
lvcreate -r none -n stok_0_122 --size 16092M vol_one
lvcreate -r none -n stok_0_123 --size 16092M vol_one
lvcreate -r none -n stok_0_124 --size 16092M vol_one
lvcreate -r none -n stok_0_125 --size 16092M vol_one
lvcreate -r none -n stok_0_126 --size 16092M vol_one
lvcreate -r none -n stok_0_127 --size 16092M vol_one
lvcreate -r none -n stok_0_128 --size 16092M vol_one
lvcreate -r none -n stok_0_129 --size 16092M vol_one
lvcreate -r none -n stok_0_130 --size 16092M vol_one
lvcreate -r none -n stok_0_131 --size 16092M vol_one
lvcreate -r none -n stok_0_132 --size 16092M vol_one
lvcreate -r none -n stok_0_133 --size 16092M vol_one
lvcreate -r none -n stok_0_134 --size 16092M vol_one
lvcreate -r none -n stok_0_135 --size 16092M vol_one
lvcreate -r none -n stok_0_136 --size 16092M vol_one
lvcreate -r none -n stok_0_137 --size 16092M vol_one
lvcreate -r none -n stok_0_138 --size 16092M vol_one
lvcreate -r none -n stok_0_139 --size 16092M vol_one
lvcreate -r none -n stok_0_140 --size 16092M vol_one
lvcreate -r none -n stok_0_141 --size 16092M vol_one
lvcreate -r none -n stok_0_142 --size 16092M vol_one
lvcreate -r none -n stok_0_143 --size 16092M vol_one
lvcreate -r none -n stok_0_144 --size 16092M vol_one
lvcreate -r none -n stok_0_145 --size 16092M vol_one
lvcreate -r none -n stok_0_146 --size 16092M vol_one
lvcreate -r none -n stok_0_147 --size 16092M vol_one
lvcreate -r none -n stok_0_148 --size 16092M vol_one
lvcreate -r none -n stok_0_149 --size 16092M vol_one
lvcreate -r none -n stok_0_150 --size 16092M vol_one
lvcreate -r none -n stok_0_151 --size 16092M vol_one
lvcreate -r none -n stok_0_152 --size 16092M vol_one
lvcreate -r none -n stok_0_153 --size 16092M vol_one
lvcreate -r none -n stok_0_154 --size 16092M vol_one
lvcreate -r none -n stok_0_155 --size 16092M vol_one
lvcreate -r none -n stok_0_156 --size 16092M vol_one
lvcreate -r none -n stok_0_157 --size 16092M vol_one
lvcreate -r none -n stok_0_158 --size 16092M vol_one
lvcreate -r none -n stok_0_159 --size 16092M vol_one
lvcreate -r none -n item_0_0 --size 22M vol_one
lvcreate -r none -n ordr_0_0 --size 63742M vol_one
lvcreate -r none -n ordr_0_1 --size 63742M vol_one
lvcreate -r none -n ordr_0_2 --size 63742M vol_one
lvcreate -r none -n ordr_0_3 --size 63742M vol_one
lvcreate -r none -n ordr_0_4 --size 63742M vol_one
lvcreate -r none -n ordr_0_5 --size 63742M vol_one
lvcreate -r none -n ordr_0_6 --size 63742M vol_one
lvcreate -r none -n ordr_0_7 --size 63742M vol_one
lvcreate -r none -n ordr_0_8 --size 63742M vol_one
lvcreate -r none -n ordr_0_9 --size 63742M vol_one
lvcreate -r none -n ordr_0_10 --size 63742M vol_one
lvcreate -r none -n ordr_0_11 --size 63742M vol_one
lvcreate -r none -n ordr_0_12 --size 63742M vol_one
lvcreate -r none -n ordr_0_13 --size 63742M vol_one
lvcreate -r none -n ordr_0_14 --size 63742M vol_one
lvcreate -r none -n ordr_0_15 --size 63742M vol_one
lvcreate -r none -n ordr_0_16 --size 63742M vol_one
lvcreate -r none -n ordr_0_17 --size 63742M vol_one
lvcreate -r none -n ordr_0_18 --size 63742M vol_one
lvcreate -r none -n ordr_0_19 --size 63742M vol_one
lvcreate -r none -n ordr_0_20 --size 63742M vol_one
lvcreate -r none -n ordr_0_21 --size 63742M vol_one
lvcreate -r none -n ordr_0_22 --size 63742M vol_one
lvcreate -r none -n ordr_0_23 --size 63742M vol_one
lvcreate -r none -n ordr_0_24 --size 63742M vol_one
lvcreate -r none -n ordr_0_25 --size 63742M vol_one
lvcreate -r none -n ordr_0_26 --size 63742M vol_one
lvcreate -r none -n ordr_0_27 --size 63742M vol_one
lvcreate -r none -n ordr_0_28 --size 63742M vol_one
lvcreate -r none -n ordr_0_29 --size 63742M vol_one
lvcreate -r none -n ordr_0_30 --size 63742M vol_one
lvcreate -r none -n ordr_0_31 --size 63742M vol_one
lvcreate -r none -n ordr_0_32 --size 63742M vol_one
lvcreate -r none -n ordr_0_33 --size 63742M vol_one
lvcreate -r none -n ordr_0_34 --size 63742M vol_one
lvcreate -r none -n ordr_0_35 --size 63742M vol_one
lvcreate -r none -n ordr_0_36 --size 63742M vol_one
lvcreate -r none -n ordr_0_37 --size 63742M vol_one
lvcreate -r none -n ordr_0_38 --size 63742M vol_one
lvcreate -r none -n ordr_0_39 --size 63742M vol_one
lvcreate -r none -n ordr_0_40 --size 63742M vol_one
lvcreate -r none -n ordr_0_41 --size 63742M vol_one
lvcreate -r none -n ordr_0_42 --size 63742M vol_one
lvcreate -r none -n ordr_0_43 --size 63742M vol_one
lvcreate -r none -n ordr_0_44 --size 63742M vol_one
lvcreate -r none -n ordr_0_45 --size 63742M vol_one
lvcreate -r none -n nord_0_0 --size 462M vol_one
lvcreate -r none -n nord_0_1 --size 462M vol_one
lvcreate -r none -n nord_0_2 --size 462M vol_one
lvcreate -r none -n nord_0_3 --size 462M vol_one
lvcreate -r none -n nord_0_4 --size 462M vol_one
lvcreate -r none -n nord_0_5 --size 462M vol_one
lvcreate -r none -n nord_0_6 --size 462M vol_one
lvcreate -r none -n nord_0_7 --size 462M vol_one
lvcreate -r none -n nord_0_8 --size 462M vol_one
lvcreate -r none -n nord_0_9 --size 462M vol_one
lvcreate -r none -n nord_0_10 --size 462M vol_one
lvcreate -r none -n nord_0_11 --size 462M vol_one
lvcreate -r none -n nord_0_12 --size 462M vol_one
lvcreate -r none -n nord_0_13 --size 462M vol_one
lvcreate -r none -n nord_0_14 --size 462M vol_one
lvcreate -r none -n nord_0_15 --size 462M vol_one
lvcreate -r none -n nord_0_16 --size 462M vol_one
lvcreate -r none -n nord_0_17 --size 462M vol_one
lvcreate -r none -n nord_0_18 --size 462M vol_one
lvcreate -r none -n nord_0_19 --size 462M vol_one
lvcreate -r none -n nord_0_20 --size 462M vol_one
lvcreate -r none -n nord_0_21 --size 462M vol_one
lvcreate -r none -n nord_0_22 --size 462M vol_one
lvcreate -r none -n nord_0_23 --size 462M vol_one
lvcreate -r none -n nord_0_24 --size 462M vol_one
lvcreate -r none -n nord_0_25 --size 462M vol_one
lvcreate -r none -n nord_0_26 --size 462M vol_one
lvcreate -r none -n nord_0_27 --size 462M vol_one
lvcreate -r none -n nord_0_28 --size 462M vol_one
lvcreate -r none -n nord_0_29 --size 462M vol_one
lvcreate -r none -n nord_0_30 --size 462M vol_one
lvcreate -r none -n nord_0_31 --size 462M vol_one
lvcreate -r none -n nord_0_32 --size 462M vol_one
lvcreate -r none -n nord_0_33 --size 462M vol_one
lvcreate -r none -n nord_0_34 --size 462M vol_one
lvcreate -r none -n nord_0_35 --size 462M vol_one
lvcreate -r none -n nord_0_36 --size 462M vol_one
lvcreate -r none -n nord_0_37 --size 462M vol_one
lvcreate -r none -n nord_0_38 --size 462M vol_one
lvcreate -r none -n nord_0_39 --size 462M vol_one
lvcreate -r none -n nord_0_40 --size 462M vol_one
lvcreate -r none -n nord_0_41 --size 462M vol_one
lvcreate -r none -n nord_0_42 --size 462M vol_one
lvcreate -r none -n nord_0_43 --size 462M vol_one
lvcreate -r none -n nord_0_44 --size 462M vol_one
lvcreate -r none -n nord_0_45 --size 462M vol_one
lvcreate -r none -n nord_0_46 --size 462M vol_one
lvcreate -r none -n nord_0_47 --size 462M vol_one
lvcreate -r none -n nord_0_48 --size 462M vol_one
lvcreate -r none -n nord_0_49 --size 462M vol_one
lvcreate -r none -n nord_0_50 --size 462M vol_one
lvcreate -r none -n nord_0_51 --size 462M vol_one
lvcreate -r none -n nord_0_52 --size 462M vol_one
lvcreate -r none -n nord_0_53 --size 462M vol_one
lvcreate -r none -n nord_0_54 --size 462M vol_one
lvcreate -r none -n nord_0_55 --size 462M vol_one
lvcreate -r none -n nord_0_56 --size 462M vol_one
lvcreate -r none -n nord_0_57 --size 462M vol_one
lvcreate -r none -n nord_0_58 --size 462M vol_one
lvcreate -r none -n nord_0_59 --size 462M vol_one
lvcreate -r none -n nord_0_60 --size 462M vol_one
lvcreate -r none -n nord_0_61 --size 462M vol_one
lvcreate -r none -n nord_0_62 --size 462M vol_one
lvcreate -r none -n nord_0_63 --size 462M vol_one
lvcreate -r none -n iware_0_0 --size 200M vol_one
lvcreate -r none -n icust1_0_0 --size 32400M vol_one
lvcreate -r none -n icust1_0_1 --size 32400M vol_one
lvcreate -r none -n idist_0_0 --size 432M vol_one
lvcreate -r none -n istok_0_0 --size 45600M vol_one
lvcreate -r none -n istok_0_1 --size 45600M vol_one
lvcreate -r none -n istok_0_2 --size 45600M vol_one
lvcreate -r none -n istok_0_3 --size 45600M vol_one
lvcreate -r none -n iitem_0_0 --size 22M vol_one
lvcreate -r none -n iordr2_0_0 --size 8302M vol_one
lvcreate -r none -n iordr2_0_1 --size 8302M vol_one
lvcreate -r none -n iordr2_0_2 --size 8302M vol_one
lvcreate -r none -n iordr2_0_3 --size 8302M vol_one
lvcreate -r none -n iordr2_0_4 --size 8302M vol_one
lvcreate -r none -n iordr2_0_5 --size 8302M vol_one
lvcreate -r none -n iordr2_0_6 --size 8302M vol_one
lvcreate -r none -n iordr2_0_7 --size 8302M vol_one
lvcreate -r none -n iordr2_0_8 --size 8302M vol_one
lvcreate -r none -n iordr2_0_9 --size 8302M vol_one
lvcreate -r none -n iordr2_0_10 --size 8302M vol_one
lvcreate -r none -n iordr2_0_11 --size 8302M vol_one
lvcreate -r none -n iordr2_0_12 --size 8302M vol_one
lvcreate -r none -n iordr2_0_13 --size 8302M vol_one
lvcreate -r none -n iordr2_0_14 --size 8302M vol_one
lvcreate -r none -n iordr2_0_15 --size 8302M vol_one
lvcreate -r none -n temp_0_0 --size 16080M vol_one
lvcreate -r none -n temp_0_1 --size 16080M vol_one
lvcreate -r none -n temp_0_2 --size 16080M vol_one
lvcreate -r none -n temp_0_3 --size 16080M vol_one
lvcreate -r none -n temp_0_4 --size 16080M vol_one
lvcreate -r none -n temp_0_5 --size 16080M vol_one
lvcreate -r none -n temp_0_6 --size 16080M vol_one
lvcreate -r none -n temp_0_7 --size 16080M vol_one
lvcreate -r none -n temp_0_8 --size 16080M vol_one
lvcreate -r none -n temp_0_9 --size 16080M vol_one
lvcreate -r none -n temp_0_10 --size 16080M vol_one
lvcreate -r none -n temp_0_11 --size 16080M vol_one
lvcreate -r none -n temp_0_12 --size 16080M vol_one
lvcreate -r none -n temp_0_13 --size 16080M vol_one
lvcreate -r none -n temp_0_14 --size 16080M vol_one
lvcreate -r none -n temp_0_15 --size 16080M vol_one
lvcreate -r none -n temp_0_16 --size 16080M vol_one
lvcreate -r none -n temp_0_17 --size 16080M vol_one
lvcreate -r none -n temp_0_18 --size 16080M vol_one
lvcreate -r none -n temp_0_19 --size 16080M vol_one
lvcreate -r none -n temp_0_20 --size 16080M vol_one
lvcreate -r none -n temp_0_21 --size 16080M vol_one
lvcreate -r none -n temp_0_22 --size 16080M vol_one
lvcreate -r none -n temp_0_23 --size 16080M vol_one
lvcreate -r none -n temp_0_24 --size 16080M vol_one
lvcreate -r none -n temp_0_25 --size 16080M vol_one
lvcreate -r none -n temp_0_26 --size 16080M vol_one
lvcreate -r none -n temp_0_27 --size 16080M vol_one
lvcreate -r none -n tpccaux --size 122M vol_one
lvcreate -r none -n control_001 --size 28M vol_one
lvcreate -r none -n control_002 --size 28M vol_one
lvcreate -r none -n sp_0 --size 2050M vol_one
lvcreate -r none -n system_1 --size 402M vol_one
lvcreate -r none -n roll1 --size 8098M vol_one
lvcreate -r none -n icust2_0_0 --size 8452M vol_two
lvcreate -r none -n icust2_0_1 --size 8452M vol_two
lvcreate -r none -n icust2_0_2 --size 8452M vol_two
lvcreate -r none -n icust2_0_3 --size 8452M vol_two
lvcreate -r none -n icust2_0_4 --size 8452M vol_two
lvcreate -r none -n icust2_0_5 --size 8452M vol_two
lvcreate -r none -n icust2_0_6 --size 8452M vol_two
lvcreate -r none -n icust2_0_7 --size 8452M vol_two
lvcreate -r none -n icust2_0_8 --size 8452M vol_two
lvcreate -r none -n icust2_0_9 --size 8452M vol_two
lvcreate -r none -n icust2_0_10 --size 8452M vol_two
lvcreate -r none -n icust2_0_11 --size 8452M vol_two
lvcreate -r none -n icust2_0_12 --size 8452M vol_two
lvcreate -r none -n icust2_0_13 --size 8452M vol_two
lvcreate -r none -n icust2_0_14 --size 8452M vol_two
lvcreate -r none -n icust2_0_15 --size 8452M vol_two
lvcreate -r none -n cust_0_0 --size 16012M vol_two
lvcreate -r none -n cust_0_1 --size 16012M vol_two
lvcreate -r none -n cust_0_2 --size 16012M vol_two
lvcreate -r none -n cust_0_3 --size 16012M vol_two
lvcreate -r none -n cust_0_4 --size 16012M vol_two
lvcreate -r none -n cust_0_5 --size 16012M vol_two
lvcreate -r none -n cust_0_6 --size 16012M vol_two
lvcreate -r none -n cust_0_7 --size 16012M vol_two
lvcreate -r none -n cust_0_8 --size 16012M vol_two
lvcreate -r none -n cust_0_9 --size 16012M vol_two
lvcreate -r none -n cust_0_10 --size 16012M vol_two
lvcreate -r none -n cust_0_11 --size 16012M vol_two
lvcreate -r none -n cust_0_12 --size 16012M vol_two
lvcreate -r none -n cust_0_13 --size 16012M vol_two
lvcreate -r none -n cust_0_14 --size 16012M vol_two
lvcreate -r none -n cust_0_15 --size 16012M vol_two
lvcreate -r none -n cust_0_16 --size 16012M vol_two
lvcreate -r none -n cust_0_17 --size 16012M vol_two
lvcreate -r none -n cust_0_18 --size 16012M vol_two
lvcreate -r none -n cust_0_19 --size 16012M vol_two
lvcreate -r none -n cust_0_20 --size 16012M vol_two
lvcreate -r none -n cust_0_21 --size 16012M vol_two
lvcreate -r none -n cust_0_22 --size 16012M vol_two
lvcreate -r none -n cust_0_23 --size 16012M vol_two
lvcreate -r none -n cust_0_24 --size 16012M vol_two
lvcreate -r none -n cust_0_25 --size 16012M vol_two
lvcreate -r none -n cust_0_26 --size 16012M vol_two
lvcreate -r none -n cust_0_27 --size 16012M vol_two
lvcreate -r none -n cust_0_28 --size 16012M vol_two
lvcreate -r none -n cust_0_29 --size 16012M vol_two
lvcreate -r none -n cust_0_30 --size 16012M vol_two
lvcreate -r none -n cust_0_31 --size 16012M vol_two
lvcreate -r none -n cust_0_32 --size 16012M vol_two
lvcreate -r none -n cust_0_33 --size 16012M vol_two
lvcreate -r none -n cust_0_34 --size 16012M vol_two
lvcreate -r none -n cust_0_35 --size 16012M vol_two
lvcreate -r none -n cust_0_36 --size 16012M vol_two
lvcreate -r none -n cust_0_37 --size 16012M vol_two
lvcreate -r none -n cust_0_38 --size 16012M vol_two
lvcreate -r none -n cust_0_39 --size 16012M vol_two
lvcreate -r none -n cust_0_40 --size 16012M vol_two
lvcreate -r none -n cust_0_41 --size 16012M vol_two
lvcreate -r none -n cust_0_42 --size 16012M vol_two
lvcreate -r none -n cust_0_43 --size 16012M vol_two
lvcreate -r none -n cust_0_44 --size 16012M vol_two
lvcreate -r none -n cust_0_45 --size 16012M vol_two
lvcreate -r none -n cust_0_46 --size 16012M vol_two
lvcreate -r none -n cust_0_47 --size 16012M vol_two
lvcreate -r none -n cust_0_48 --size 16012M vol_two
lvcreate -r none -n cust_0_49 --size 16012M vol_two
lvcreate -r none -n cust_0_50 --size 16012M vol_two
lvcreate -r none -n cust_0_51 --size 16012M vol_two
lvcreate -r none -n cust_0_52 --size 16012M vol_two
lvcreate -r none -n cust_0_53 --size 16012M vol_two
lvcreate -r none -n cust_0_54 --size 16012M vol_two
lvcreate -r none -n cust_0_55 --size 16012M vol_two
lvcreate -r none -n cust_0_56 --size 16012M vol_two
lvcreate -r none -n cust_0_57 --size 16012M vol_two
lvcreate -r none -n cust_0_58 --size 16012M vol_two
lvcreate -r none -n cust_0_59 --size 16012M vol_two
lvcreate -r none -n cust_0_60 --size 16012M vol_two
lvcreate -r none -n cust_0_61 --size 16012M vol_two
lvcreate -r none -n cust_0_62 --size 16012M vol_two
lvcreate -r none -n cust_0_63 --size 16012M vol_two
lvcreate -r none -n cust_0_64 --size 16012M vol_two
lvcreate -r none -n cust_0_65 --size 16012M vol_two
lvcreate -r none -n cust_0_66 --size 16012M vol_two
lvcreate -r none -n cust_0_67 --size 16012M vol_two
lvcreate -r none -n cust_0_68 --size 16012M vol_two
lvcreate -r none -n cust_0_69 --size 16012M vol_two
lvcreate -r none -n cust_0_70 --size 16012M vol_two
lvcreate -r none -n cust_0_71 --size 16012M vol_two
lvcreate -r none -n cust_0_72 --size 16012M vol_two
lvcreate -r none -n cust_0_73 --size 16012M vol_two
lvcreate -r none -n cust_0_74 --size 16012M vol_two
lvcreate -r none -n cust_0_75 --size 16012M vol_two
lvcreate -r none -n cust_0_76 --size 16012M vol_two
lvcreate -r none -n cust_0_77 --size 16012M vol_two
lvcreate -r none -n cust_0_78 --size 16012M vol_two
lvcreate -r none -n cust_0_79 --size 16012M vol_two
lvcreate -r none -n cust_0_80 --size 16012M vol_two
lvcreate -r none -n cust_0_81 --size 16012M vol_two
lvcreate -r none -n cust_0_82 --size 16012M vol_two
lvcreate -r none -n cust_0_83 --size 16012M vol_two
lvcreate -r none -n cust_0_84 --size 16012M vol_two
lvcreate -r none -n cust_0_85 --size 16012M vol_two
lvcreate -r none -n cust_0_86 --size 16012M vol_two
lvcreate -r none -n cust_0_87 --size 16012M vol_two
lvcreate -r none -n cust_0_88 --size 16012M vol_two
lvcreate -r none -n cust_0_89 --size 16012M vol_two
lvcreate -r none -n cust_0_90 --size 16012M vol_two
lvcreate -r none -n cust_0_91 --size 16012M vol_two
lvcreate -r none -n cust_0_92 --size 16012M vol_two
lvcreate -r none -n cust_0_93 --size 16012M vol_two
lvcreate -r none -n cust_0_94 --size 16012M vol_two
lvcreate -r none -n cust_0_95 --size 16012M vol_two
lvcreate -r none -n cust_0_96 --size 16012M vol_two
lvcreate -r none -n cust_0_97 --size 16012M vol_two
lvcreate -r none -n cust_0_98 --size 16012M vol_two
lvcreate -r none -n cust_0_99 --size 16012M vol_two
lvcreate -r none -n cust_0_100 --size 16012M vol_two
lvcreate -r none -n cust_0_101 --size 16012M vol_two
lvcreate -r none -n cust_0_102 --size 16012M vol_two
lvcreate -r none -n cust_0_103 --size 16012M vol_two
lvcreate -r none -n cust_0_104 --size 16012M vol_two
lvcreate -r none -n cust_0_105 --size 16012M vol_two
lvcreate -r none -n cust_0_106 --size 16012M vol_two
lvcreate -r none -n cust_0_107 --size 16012M vol_two
lvcreate -r none -n cust_0_108 --size 16012M vol_two
lvcreate -r none -n cust_0_109 --size 16012M vol_two
lvcreate -r none -n cust_0_110 --size 16012M vol_one
lvcreate -r none -n cust_0_111 --size 16012M vol_one
lvcreate -r none -n cust_0_112 --size 16012M vol_one
lvcreate -r none -n cust_0_113 --size 16012M vol_one
lvcreate -r none -n cust_0_114 --size 16012M vol_one
lvcreate -r none -n cust_0_115 --size 16012M vol_one
lvcreate -r none -n cust_0_116 --size 16012M vol_one
lvcreate -r none -n cust_0_117 --size 16012M vol_one
lvcreate -r none -n cust_0_118 --size 16012M vol_one
lvcreate -r none -n cust_0_119 --size 16012M vol_one
lvcreate -r none -n cust_0_120 --size 16012M vol_one
lvcreate -r none -n cust_0_121 --size 16012M vol_one
lvcreate -r none -n cust_0_122 --size 16012M vol_one
lvcreate -r none -n cust_0_123 --size 16012M vol_one
lvcreate -r none -n cust_0_124 --size 16012M vol_one
lvcreate -r none -n cust_0_125 --size 16012M vol_one
lvcreate -r none -n cust_0_126 --size 16012M vol_one
lvcreate -r none -n cust_0_127 --size 16012M vol_one
lvcreate -r none -n cust_0_128 --size 16012M vol_one
lvcreate -r none -n cust_0_129 --size 16012M vol_one
lvcreate -r none -n cust_0_130 --size 16012M vol_one
lvcreate -r none -n cust_0_131 --size 16012M vol_one
lvcreate -r none -n cust_0_132 --size 16012M vol_one
lvcreate -r none -n cust_0_133 --size 16012M vol_one
lvcreate -r none -n cust_0_134 --size 16012M vol_one
lvcreate -r none -n cust_0_135 --size 16012M vol_one
lvcreate -r none -n cust_0_136 --size 16012M vol_one
lvcreate -r none -n cust_0_137 --size 16012M vol_one
lvcreate -r none -n cust_0_138 --size 16012M vol_one
lvcreate -r none -n cust_0_139 --size 16012M vol_one
lvcreate -r none -n cust_0_140 --size 16012M vol_one
lvcreate -r none -n cust_0_141 --size 16012M vol_one
lvcreate -r none -n cust_0_142 --size 16012M vol_one
lvcreate -r none -n cust_0_143 --size 16012M vol_one
lvcreate -r none -n cust_0_144 --size 16012M vol_one
lvcreate -r none -n cust_0_145 --size 16012M vol_one
lvcreate -r none -n cust_0_146 --size 16012M vol_one
lvcreate -r none -n cust_0_147 --size 16012M vol_one
lvcreate -r none -n cust_0_148 --size 16012M vol_one
lvcreate -r none -n cust_0_149 --size 16012M vol_one
lvcreate -r none -n cust_0_150 --size 16012M vol_one
lvcreate -r none -n cust_0_151 --size 16012M vol_one
lvcreate -r none -n cust_0_152 --size 16012M vol_one
lvcreate -r none -n cust_0_153 --size 16012M vol_one
lvcreate -r none -n cust_0_154 --size 16012M vol_one
lvcreate -r none -n cust_0_155 --size 16012M vol_one
lvcreate -r none -n cust_0_156 --size 16012M vol_one
lvcreate -r none -n cust_0_157 --size 16012M vol_one
lvcreate -r none -n cust_0_158 --size 16012M vol_one
lvcreate -r none -n cust_0_159 --size 16012M vol_one
lvcreate -r none -n cust_0_160 --size 16012M vol_one
lvcreate -r none -n cust_0_161 --size 16012M vol_one
lvcreate -r none -n cust_0_162 --size 16012M vol_one
lvcreate -r none -n cust_0_163 --size 16012M vol_one
```

- - - - - - - - - - - - - - - - - -
db.txt
- - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - -
loadcust.sh
- - - - - - - - - - - - - - - - - -

```
#created automatically by /home/oracle/kit.4k.86kwh/scripts/evenload.sh
Wed Sep 21 15:47:40 PDT 2011
rm -f loadcust*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 86000 -C  -l 1 -m 62 >> loadcust0.log 2>&1 &
```

```
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 63 -m 124 >> loadcust1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 125 -m 186 >> loadcust2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 187 -m 248 >> loadcust3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 249 -m 310 >> loadcust4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 311 -m 372 >> loadcust5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 373 -m 434 >> loadcust6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 435 -m 496 >> loadcust7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 497 -m 558 >> loadcust8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 559 -m 620 >> loadcust9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 621 -m 682 >> loadcust10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 683 -m 744 >> loadcust11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 745 -m 806 >> loadcust12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 807 -m 868 >> loadcust13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 869 -m 930 >> loadcust14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 931 -m 992 >> loadcust15.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 993 -m 1054 >> loadcust16.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1055 -m 1116 >> loadcust17.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1117 -m 1178 >> loadcust18.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1179 -m 1240 >> loadcust19.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1241 -m 1302 >> loadcust20.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1303 -m 1364 >> loadcust21.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1365 -m 1426 >> loadcust22.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1427 -m 1488 >> loadcust23.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1489 -m 1551 >> loadcust24.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1552 -m 1614 >> loadcust25.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1615 -m 1677 >> loadcust26.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1678 -m 1740 >> loadcust27.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1741 -m 1803 >> loadcust28.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1804 -m 1866 >> loadcust29.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1867 -m 1929 >> loadcust30.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1930 -m 1992 >> loadcust31.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 1993 -m 2055 >> loadcust32.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2056 -m 2118 >> loadcust33.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2119 -m 2181 >> loadcust34.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2182 -m 2244 >> loadcust35.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2245 -m 2307 >> loadcust36.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2308 -m 2370 >> loadcust37.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2371 -m 2433 >> loadcust38.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2434 -m 2496 >> loadcust39.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2497 -m 2559 >> loadcust40.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2560 -m 2622 >> loadcust41.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2623 -m 2685 >> loadcust42.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2686 -m 2748 >> loadcust43.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2749 -m 2811 >> loadcust44.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2812 -m 2874 >> loadcust45.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2875 -m 2937 >> loadcust46.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -C  -l 2938 -m 3000 >> loadcust47.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

- - - - - - - - - - - - - - - - - -
loaddist.sh
- - - - - - - - - - - - - - - - - -

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -d > loaddist.log 2>&1
```

- - - - - - - - - - - - - - - - - -
loadhist.sh
- - - - - - - - - - - - - - - - - -

```
#created automatically by /home/oracle/kit.4k.86kwh/scripts/evenload.sh
Wed Sep 21 15:47:39 PDT 2011
rm -f loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 86000 -h  -b 1 -e 5375 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 5376 -e 10750 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 10751 -e 16125 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 16126 -e 21500 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 21501 -e 26875 >> loadhist4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 26876 -e 32250 >> loadhist5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 32251 -e 37625 >> loadhist6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 37626 -e 43000 >> loadhist7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 43001 -e 48375 >> loadhist8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 48376 -e 53750 >> loadhist9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 53751 -e 59125 >> loadhist10.log 2>&1 &
allprocs="$allprocs ${!}"
```

```
$tpcc_load -M 86000 -h  -b 59126 -e 64500 >> loadhist11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 64501 -e 69875 >> loadhist12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 69876 -e 75250 >> loadhist13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 75251 -e 80625 >> loadhist14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -h  -b 80626 -e 86000 >> loadhist15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

- - - - - - - - - - - - - - - - - -
loaditem.sh
- - - - - - - - - - - - - - - - - -

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -i > loaditem.log 2>&1
```

- - - - - - - - - - - - - - - - - -
loadnord.sh
- - - - - - - - - - - - - - - - - -

```
#created automatically by /home/oracle/kit.4k.86kwh/scripts/evenload.sh
Wed Sep 21 15:47:39 PDT 2011
rm -f loadnord*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 86000 -n  -b 1 -e 1791 >> loadnord0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 1792 -e 3582 >> loadnord1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 3583 -e 5373 >> loadnord2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 5374 -e 7164 >> loadnord3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 7165 -e 8955 >> loadnord4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 8956 -e 10746 >> loadnord5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 10747 -e 12537 >> loadnord6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 12538 -e 14328 >> loadnord7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 14329 -e 16119 >> loadnord8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 16120 -e 17910 >> loadnord9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 17911 -e 19701 >> loadnord10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 19702 -e 21492 >> loadnord11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 21493 -e 23283 >> loadnord12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 23284 -e 25074 >> loadnord13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 25075 -e 26865 >> loadnord14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 26866 -e 28656 >> loadnord15.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 28657 -e 30448 >> loadnord16.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 30449 -e 32240 >> loadnord17.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 32241 -e 34032 >> loadnord18.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 34033 -e 35824 >> loadnord19.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 35825 -e 37616 >> loadnord20.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 37617 -e 39408 >> loadnord21.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 39409 -e 41200 >> loadnord22.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 41201 -e 42992 >> loadnord23.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 42993 -e 44784 >> loadnord24.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 44785 -e 46576 >> loadnord25.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 46577 -e 48368 >> loadnord26.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 48369 -e 50160 >> loadnord27.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 50161 -e 51952 >> loadnord28.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 51953 -e 53744 >> loadnord29.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 53745 -e 55536 >> loadnord30.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 55537 -e 57328 >> loadnord31.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 57329 -e 59120 >> loadnord32.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 59121 -e 60912 >> loadnord33.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 60913 -e 62704 >> loadnord34.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 62705 -e 64496 >> loadnord35.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 64497 -e 66288 >> loadnord36.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 66289 -e 68080 >> loadnord37.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 68081 -e 69872 >> loadnord38.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 69873 -e 71664 >> loadnord39.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 71665 -e 73456 >> loadnord40.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 73457 -e 75248 >> loadnord41.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 75249 -e 77040 >> loadnord42.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 77041 -e 78832 >> loadnord43.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 78833 -e 80624 >> loadnord44.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 80625 -e 82416 >> loadnord45.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 82417 -e 84208 >> loadnord46.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -n  -b 84209 -e 86000 >> loadnord47.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

- - - - - - - - - - - - - - - - - -
loadordrordl.sh
- - - - - - - - - - - - - - - - - -

```
#created automatically by /home/oracle/kit.4k.86kwh/scripts/evenload.sh
Wed Sep 21 15:47:40 PDT 2011
rm -f loadordrordl*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy0.dat -b 1 -e 1791
>> loadordrordl0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy1.dat -b 1792 -e
3582 >> loadordrordl1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy2.dat -b 3583 -e
5373 >> loadordrordl2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy3.dat -b 5374 -e
7164 >> loadordrordl3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy4.dat -b 7165 -e
8955 >> loadordrordl4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy5.dat -b 8956 -e
10746 >> loadordrordl5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy6.dat -b 10747 -e
12537 >> loadordrordl6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy7.dat -b 12538 -e
14328 >> loadordrordl7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy8.dat -b 14329 -e
16119 >> loadordrordl8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy9.dat -b 16120 -e
17910 >> loadordrordl9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy10.dat -b 17911 -e
19701 >> loadordrordl10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy11.dat -b 19702 -e
21492 >> loadordrordl11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy12.dat -b 21493 -e
23283 >> loadordrordl12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy13.dat -b 23284 -e
25074 >> loadordrordl13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy14.dat -b 25075 -e
26865 >> loadordrordl14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy15.dat -b 26866 -e
28656 >> loadordrordl15.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy16.dat -b 28657 -e
30448 >> loadordrordl16.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy17.dat -b 30449 -e
32240 >> loadordrordl17.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy18.dat -b 32241 -e
34032 >> loadordrordl18.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy19.dat -b 34033 -e
35824 >> loadordrordl19.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy20.dat -b 35825 -e
37616 >> loadordrordl20.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy21.dat -b 37617 -e
39408 >> loadordrordl21.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy22.dat -b 39409 -e
41200 >> loadordrordl22.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy23.dat -b 41201 -e
42992 >> loadordrordl23.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy24.dat -b 42993 -e
44784 >> loadordrordl24.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy25.dat -b 44785 -e
46576 >> loadordrordl25.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy26.dat -b 46577 -e
48368 >> loadordrordl26.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy27.dat -b 48369 -e
50160 >> loadordrordl27.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy28.dat -b 50161 -e
51952 >> loadordrordl28.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy29.dat -b 51953 -e
53744 >> loadordrordl29.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy30.dat -b 53745 -e
55536 >> loadordrordl30.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy31.dat -b 55537 -e
57328 >> loadordrordl31.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy32.dat -b 57329 -e
59120 >> loadordrordl32.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy33.dat -b 59121 -e
60912 >> loadordrordl33.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy34.dat -b 60913 -e
62704 >> loadordrordl34.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy35.dat -b 62705 -e
64496 >> loadordrordl35.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy36.dat -b 64497 -e
66288 >> loadordrordl36.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy37.dat -b 66289 -e
68080 >> loadordrordl37.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy38.dat -b 68081 -e
69872 >> loadordrordl38.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy39.dat -b 69873 -e
71664 >> loadordrordl39.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy40.dat -b 71665 -e
73456 >> loadordrordl40.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy41.dat -b 73457 -e
75248 >> loadordrordl41.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy42.dat -b 75249 -e
77040 >> loadordrordl42.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy43.dat -b 77041 -e
78832 >> loadordrordl43.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy44.dat -b 78833 -e
80624 >> loadordrordl44.log 2>&1 &
allprocs="$allprocs ${!}"
```

```
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy45.dat -b 80625 -e
82416 >> loadordrordl45.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy46.dat -b 82417 -e
84208 >> loadordrordl46.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -o ${tpcc_disks_location}dummy47.dat -b 84209 -e
86000 >> loadordrordl47.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

- - - - - - - - - - - - - - - - - -
loadstok.sh
- - - - - - - - - - - - - - - - - -

```
#created automatically by /home/oracle/kit.4k.86kwh/scripts/evenload.sh
Wed Sep 21 15:47:40 PDT 2011
rm -f loadstok*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 86000 -S  -j 1 -k 2083 >> loadstok0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 2084 -k 4166 >> loadstok1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 4167 -k 6249 >> loadstok2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 6250 -k 8332 >> loadstok3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 8333 -k 10415 >> loadstok4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 10416 -k 12498 >> loadstok5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 12499 -k 14581 >> loadstok6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 14582 -k 16664 >> loadstok7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 16665 -k 18747 >> loadstok8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 18748 -k 20830 >> loadstok9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 20831 -k 22913 >> loadstok10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 22914 -k 24996 >> loadstok11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 24997 -k 27079 >> loadstok12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 27080 -k 29162 >> loadstok13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 29163 -k 31245 >> loadstok14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 31246 -k 33328 >> loadstok15.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 33329 -k 35411 >> loadstok16.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 35412 -k 37494 >> loadstok17.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 37495 -k 39577 >> loadstok18.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 39578 -k 41660 >> loadstok19.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 41661 -k 43743 >> loadstok20.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 43744 -k 45826 >> loadstok21.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 45827 -k 47909 >> loadstok22.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 47910 -k 49992 >> loadstok23.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 49993 -k 52075 >> loadstok24.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 52076 -k 54158 >> loadstok25.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 54159 -k 56241 >> loadstok26.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 56242 -k 58324 >> loadstok27.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 58325 -k 60407 >> loadstok28.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 60408 -k 62490 >> loadstok29.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 62491 -k 64573 >> loadstok30.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 64574 -k 66656 >> loadstok31.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 66657 -k 68740 >> loadstok32.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 68741 -k 70824 >> loadstok33.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 70825 -k 72908 >> loadstok34.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 72909 -k 74992 >> loadstok35.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 74993 -k 77076 >> loadstok36.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 77077 -k 79160 >> loadstok37.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 79161 -k 81244 >> loadstok38.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 81245 -k 83328 >> loadstok39.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 83329 -k 85412 >> loadstok40.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 85413 -k 87496 >> loadstok41.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 87497 -k 89580 >> loadstok42.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 89581 -k 91664 >> loadstok43.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 91665 -k 93748 >> loadstok44.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 93749 -k 95832 >> loadstok45.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 95833 -k 97916 >> loadstok46.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 86000 -S  -j 97917 -k 100000 >> loadstok47.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

- - - - - - - - - - - - - - - - - -
loadware.sh
- - - - - - - - - - - - - - - - - -

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -w > loadware.log 2>&1
```

- - - - - - - - - - - - - - - - - -
log-dev.sh
- - - - - - - - - - - - - - - - - -

```
ln -sf /dev/sdb1 log_1_1
ln -sf /dev/sdb2 log_1_2
ln -sf /dev/sdb3 log_1_3
ln -sf /dev/sdb5 log_1_4
ln -sf /dev/sdb6 log_1_5
ln -sf /dev/sdb7 log_1_6
ln -sf /dev/sdc1 log_2_1
ln -sf /dev/sdc2 log_2_2
ln -sf /dev/sdc3 log_2_3
ln -sf /dev/sdc5 log_2_4
ln -sf /dev/sdc6 log_2_5
ln -sf /dev/sdc7 log_2_6


- - - - - - - - - - - - - - - - - -
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) + 1 / 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;
```

- - - - - - - - - - - - - - - - - -
pv.sh
- - - - - - - - - - - - - - - - - -

```
pvcreate /dev/vtmsb
vgcreate -s 1G vol_two /dev/vtmsb
pvcreate /dev/vtmsa
vgcreate -s 1G vol_one /dev/vtmsa
pvcreate /dev/sdc
pvcreate /dev/sde
vgcreate backup_one /dev/sdc
vgcreate backup_log /dev/sde
```

- - - - - - - - - - - - - - - - - -
space_get.sql
- - - - - - - - - - - - - - - - - -

```
REM================================================
===============+
REM Copyright (c) 1995, 2008, Oracle and/or its affiliates.
REM All rights reserved.
REM                 OPEN SYSTEMS PERFORMANCE GROUP              |
REM                     All Rights Reserved                     |
REM================================================
===============+
REM FILENAME
REM       space_get.sql
REM DESCRIPTION
REM       Get sizes of tables, indexes and tablespaces.
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @space_get
[<tpm> <# of warehouses>]
REM================================================
===============*/

  set echo on;
  delete from tpcc_data;
  delete from tpcc_space;
  delete from tpcc_totspace;

  insert into tpcc_data
  select substr(segment_name,1,18), substr(segment_type,1,15),
       sum(blocks), t.block_size,
       round(sum(blocks) * 0.05), 0,
       sum(blocks) + round(sum(blocks) * 0.05)
  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 <>
'SP_0'
       AND e.tablespace_name = t.tablespace_name
  group by segment_name, segment_type, t.block_size;

  insert into tpcc_data
    select 'SYSTEM', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name = 'SYSTEM' and t.tablespace_name =
f.tablespace_name
    group by t.block_size;

  insert into tpcc_data
    select 'SYSAUX', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name = 'SYSAUX' and t.tablespace_name =
f.tablespace_name
    group by t.block_size;

  insert into tpcc_data
    select 'ROLL_SEG', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name like '%UNDO_TS%' and f.tablespace_name
= t.tablespace_name
    group by f.tablespace_name, t.block_size;

  insert into tpcc_data
    select 'DB_STAT', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name like '%SP_0%' and f.tablespace_name =
t.tablespace_name
    group by f.tablespace_name, t.block_size;

  update tpcc_data
    set five_pct = 0,
       daily_grow = round(blocks * &&1 / 62.5 / &&2),
       total = blocks + round(blocks * &&1 / 62.5 / &&2)
    where segment = 'HIST' OR segment = 'ORDRCLUSTER_QUEUE'
OR
         segment = 'IORDL';
  insert into tpcc_space
  select substr(ex$.name,1,18), sum(sp$.sz_blocks), sp$.block_size, 0, 0, 0,
0
  from
    (select f.tablespace_name , sum(blocks) sz_blocks, t.block_size
block_size
     from dba_data_files f, dba_tablespaces t
     where f.tablespace_name <> 'SYSTEM' and f.tablespace_name =
t.tablespace_name
     group by f.tablespace_name, t.block_size
    ) sp$,
    (select distinct tablespace_name, segment_name name
     from dba_extents
     where owner = 'TPCC'
       and (segment_type = 'CLUSTER' or segment_type = 'TABLE'
         or segment_type = 'TABLE PARTITION' or segment_type =
'INDEX'
         or segment_type = 'INDEX PARTITION')
       and tablespace_name <> 'SYSTEM'
    ) ex$
  where sp$.tablespace_name = ex$.tablespace_name
  group by ex$.name, sp$.block_size;

  insert into tpcc_space
  select substr(f.tablespace_name,1,18), sum(blocks), t.block_size, 0, 0, 0,
0
  from dba_data_files f, dba_tablespaces t
  where (f.tablespace_name = 'SYSTEM' or f.tablespace_name =
'SYSAUX')
        and f.tablespace_name = t.tablespace_name
  group by f.tablespace_name, t.block_size;

  insert into tpcc_space
  select 'ROLL_SEG', sum(blocks), t.block_size, 0, 0, 0, 0
  from dba_data_files f, dba_tablespaces t
  where f.tablespace_name = 'UNDO_TS' and f.tablespace_name =
t.tablespace_name
  group by f.tablespace_name, t.block_size;

  insert into tpcc_space
  select 'DB_STAT', sum(blocks), t.block_size, 0, 0, 0, 0
  from dba_data_files f, dba_tablespaces t
  where f.tablespace_name = 'SP_0' and f.tablespace_name =
t.tablespace_name
  group by f.tablespace_name, t.block_size;

  update tpcc_space
    set required =
    (
      select sum(total)
      from   tpcc_data
      where  tpcc_data.segment = tpcc_space.segment
    )
    where segment in
    (
      select segment from tpcc_data
    );

  update tpcc_space
    set static =
    (
      select sum(total)
      from   tpcc_data
      where  tpcc_data.segment = tpcc_space.segment
    )
    where segment in
    (
      select segment from tpcc_data
    );
```

```
  update tpcc_space
    set static = 0,
       dynamic =
    (
      select sum(blocks)
      from   tpcc_data
      where  tpcc_data.segment = tpcc_space.segment
    )
    where segment in ('HIST', 'ORDRCLUSTER_QUEUE', 'IORDL');

  update tpcc_space
    set oversize = blocks  - required;


  insert into tpcc_totspace
    select &&1, &&2, sum(static * block_size)/1024, sum(dynamic *
block_size)/1024, sum(oversize * block_size)/1024, 0, 0, 0
    from   tpcc_space;

  update tpcc_totspace
    set daily_grow =
    (
      select sum(daily_grow * block_size)/1024
      from   tpcc_data
    );
  update tpcc_totspace
    set space60 = static + 60 * daily_grow;
  set echo off;
```

- - - - - - - - - - - - - - - - - -
space_init.sql
- - - - - - - - - - - - - - - - - -

```
REM================================================
================+
REM FILENAME
REM       space_init.sql
REM DESCRIPTION
REM       Create tables for space calculations.
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @space_init.sql
REM================================================
================*/
  set echo on;
  drop table tpcc_data;
  drop table tpcc_space;
  drop table tpcc_totspace;
  create table tpcc_data (
    segment      varchar2(18),
    type         varchar2(15),
    blocks       number,
    block_size   number,
    five_pct     number,
    daily_grow   number,
    total        number
  );
  create table tpcc_space (
    segment       varchar2(18),
    blocks        number,
    block_size    number,
    required      number,
    static        number,
    dynamic       number,
    oversize      number
  );
  create table tpcc_totspace (
    tpm          number,
    nware        number,
    static       number,
    dynamic      number,
    oversize     number,
    daily_grow   number,
    daily_spre   number,
    space60      number
  );
  create unique index itpcc_data on tpcc_data (segment);
  create unique index itpcc_space on tpcc_space (segment);
  set echo off;
```

- - - - - - - - - - - - - - - - - -
space_rpt.sql
- - - - - - - - - - - - - - - - - -

```
REM==================================================
================+
REM Copyright (c) 1995, 2008, Oracle and/or its affiliates.
REM All rights reserved.
REM                  OPEN SYSTEMS PERFORMANCE GROUP                |
REM                      All Rights Reserved                   |
REM==================================================
================+
REM FILENAME
REM       space_rpt.sql
REM DESCRIPTION
REM       Generate space report and save it in space.rpt
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @space_rpt.sql
REM==================================================
================*/
  set space 2
  set pagesize 2000
  set echo off
  set termout off
  set verify off
  set feedback off
  set pagesize 60 linesize 120
  spool space.rpt
  select tpm, nware from tpcc_totspace;
  select * from tpcc_data order by segment;
  select * from tpcc_space order by segment;
  select static, dynamic, oversize, daily_grow, daily_spre, space60
     from tpcc_totspace;
  spool off;
```

- - - - - - - - - - - - - - - - - -
tkvcinin.sql
- - - - - - - - - - - - - - - - - -

```
Rem
Rem $Header: tk_perf/benchmark_kits/tpcc-new/scripts/sql/tkvcinin.sql
/main/2 2008/12/15 05:58:45 avliet Exp $
Rem
Rem tkvcinin.sql
Rem
Rem Copyright (c) 2001, 2008, Oracle and/or its affiliates.
Rem All rights reserved.
Rem
Rem    NAME
Rem      tkvcinin.sql - <one-line expansion of the name>
Rem
Rem    DESCRIPTION
Rem      <short description of component this file declares/defines>
Rem
Rem    NOTES
Rem      <other useful comments, qualifications, etc.>
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    heri        05/03/02 - Short table names.
Rem    lwang       07/24/01 - remove SET
Rem    lwang       05/22/01 - Merged lwang_createdb
Rem    lwang       05/21/01 - Created
Rem

-- The initnew package for storing variables used in the
-- New Order anonymous block

CREATE OR REPLACE PACKAGE inittpcc
AS
 TYPE intarray IS TABLE OF INTEGER index by binary_integer;
 TYPE distarray IS TABLE OF VARCHAR(24) index by binary_integer;
 TYPE rowidarray IS TABLE OF ROWID INDEX BY
BINARY_INTEGER;
 nulldate       DATE;
 s_dist                                      distarray;
 idx1arr                      intarray;
 s_remote                     intarray;
 dist                  intarray;
 row_id                 rowidarray;
 cust_rowid             rowid;
 dist_name              varchar2(11);
 ware_name              varchar2(11);
 c_num                  pls_integer;
 PROCEDURE init_no(idxarr intarray);
 PROCEDURE init_del;
 PROCEDURE init_pay;
END inittpcc;
/
show errors;

CREATE OR REPLACE PACKAGE BODY inittpcc AS
  PROCEDURE init_no (idxarr  intarray)
  IS
  BEGIN
       -- initialize null date
   nulldate := TO_DATE('09-15-1811', 'MM-DD-YYYY');

               -- we found a savings of ~500 instructions when
               -- initializing the cr_date array on both the client
               -- and the server, instead of initializing on
               -- the client and passing it to the server. this cannot be done
       -- as we require the current system date
               -- cr_date := SYSDATE;

               -- initialize idx1arr on the client and store it here
               -- as a package variable
    idx1arr := idxarr;
  END init_no;

  PROCEDURE init_del
  IS
  BEGIN
    FOR i IN 1 .. 10 LOOP
      dist(i) := i;
    END LOOP;
  END init_del;

  PROCEDURE init_pay IS
  BEGIN
    NULL;
  END init_pay;

END inittpcc;
/
show errors

CREATE OR REPLACE PACKAGE tpcc
AS
 TYPE intarray IS TABLE OF INTEGER index by binary_integer;
 TYPE distarray IS TABLE OF VARCHAR(24) index by binary_integer;
 TYPE rowidarray IS TABLE OF ROWID INDEX BY
BINARY_INTEGER;
 TYPE chararray IS TABLE OF VARCHAR(1) index by binary_integer;
 TYPE numarray IS TABLE OF NUMBER index by binary_integer;
 TYPE datarray IS TABLE OF DATE INDEX BY BINARY_INTEGER;
 nulldate       DATE;
 s_dist                                  distarray;
 idx1arr                     intarray;
 s_remote                    intarray;
 dist                 intarray;
 row_id               rowidarray;
 cust_rowid            rowid;
 dist_name             varchar2(11);
 ware_name              varchar2(11);
 c_num                pls_integer;

PROCEDURE neworder (
  par_w_id BINARY_INTEGER,
  par_d_id BINARY_INTEGER,
  par_c_id BINARY_INTEGER,
  par_o_all_local BINARY_INTEGER,
  par_o_ol_cnt IN OUT BINARY_INTEGER,
  par_w_tax IN OUT BINARY_INTEGER,
  par_d_tax IN OUT BINARY_INTEGER,
  par_o_id IN OUT BINARY_INTEGER,
  par_c_discount IN OUT BINARY_INTEGER,
  par_c_credit IN OUT varchar2,
  par_c_last IN OUT varchar2,
  par_retry IN OUT BINARY_INTEGER,
  par_cr_date DATE,
  par_ol_i_id intarray,
  par_ol_supply_w_id intarray,
  par_i_price IN OUT numarray,
  par_i_name IN OUT distarray,
  par_s_quantity IN OUT intarray,
  par_brand_generic IN OUT chararray,
  par_ol_amount IN OUT intarray,
  par_s_remote intarray,
  par_ol_quantity intarray
);

PROCEDURE orderstatus (
      ware_id                 INTEGER,
      dist_id                 INTEGER,
      cust_id          IN OUT INTEGER,
      bylastname              INTEGER,
      cust_last        IN OUT VARCHAR2,
      cust_first          OUT VARCHAR2,
      cust_middle         OUT VARCHAR2,
      cust_balance        OUT NUMBER,
      ord_id           IN OUT INTEGER,
      ord_entry_d         OUT VARCHAR2,
      ord_carrier_id      OUT INTEGER,
      ord_ol_cnt          OUT INTEGER,
      oline_supply_w_id IN OUT intarray,
      oline_i_id        IN OUT intarray,
      oline_quantity    IN OUT intarray,
      oline_amount      IN OUT numarray,
      oline_delivery_d     OUT datarray
);

PROCEDURE delivery (
      ware_id                         IN          INTEGER,
      dist_id               IN OUT    intarray,
      order_id                        OUT         intarray,
      ordcnt                OUT       INTEGER,
      sums    OUT           intarray,
      del_date                        IN          DATE,
      carrier_id            IN        INTEGER,
      order_c_id            OUT       intarray,
      retry                   IN OUT    INTEGER
);

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

PROCEDURE stocklevel (
      ware_id      INTEGER,
      dist_id      INTEGER,
      threshold    INTEGER,
      low_stock OUT INTEGER
   );
END tpcc;
/
show errors;

CREATE OR REPLACE PACKAGE BODY tpcc AS
     rows_lost                   BINARY_INTEGER;
     max_index                   BINARY_INTEGER;
     temp_index                  BINARY_INTEGER;

      idx                   BINARY_INTEGER;
      dummy_local               BINARY_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 neworder (
  par_w_id BINARY_INTEGER,
  par_d_id BINARY_INTEGER,
  par_c_id BINARY_INTEGER,
  par_o_all_local BINARY_INTEGER,
  par_o_ol_cnt IN OUT BINARY_INTEGER,
  par_w_tax IN OUT BINARY_INTEGER,
  par_d_tax IN OUT BINARY_INTEGER,
  par_o_id IN OUT BINARY_INTEGER,
  par_c_discount IN OUT BINARY_INTEGER,
  par_c_credit IN OUT varchar2,
  par_c_last IN OUT varchar2,
  par_retry IN OUT BINARY_INTEGER,
  par_cr_date DATE,
  par_ol_i_id intarray,
  par_ol_supply_w_id intarray,
  par_i_price IN OUT numarray,
  par_i_name IN OUT distarray,
  par_s_quantity IN OUT intarray,
  par_brand_generic IN OUT chararray,
  par_ol_amount IN OUT intarray,
  par_s_remote intarray,
  par_ol_quantity intarray
  )
  IS
   BEGIN
      LOOP BEGIN
        UPDATE dist SET d_next_o_id = d_next_o_id + 1
         WHERE d_id = par_d_id AND  d_w_id = par_w_id
         RETURNING d_tax, d_next_o_id-1
         INTO par_d_tax, par_o_id;


        SELECT c_discount, c_credit, c_last
         INTO par_c_discount, par_c_credit, par_c_last
         FROM cust
         WHERE c_id = par_c_id AND c_d_id = par_d_id AND c_w_id =
par_w_id;

        SELECT w_tax
         INTO par_w_tax
         FROM ware
         WHERE w_id = par_w_id;


        INSERT INTO nord
         VALUES (par_w_id, par_d_id, par_o_id);
        INSERT INTO ordr
         VALUES (par_o_id, par_w_id, par_d_id, par_c_id, 11,
                par_o_ol_cnt, par_o_all_local, par_cr_date);

        -- copying par_d_id in local variable is important - lots of instr.
        dummy_local :=  par_d_id;

        CASE dummy_local
------------- u1 thru u10 --- BEGIN ---
WHEN 1 THEN
-- +++++++ u1
   BEGIN
        -- we found savings of ~900 SPARC instructions when using
        -- o_ol_cnt as a host bind instead of storing it in a variable.
       FORALL idx IN 1 .. par_o_ol_cnt
         UPDATE /*+ VECTOR_READ */ stock_item
         SET s_order_cnt = s_order_cnt + 1,
         s_ytd = s_ytd + par_ol_quantity(idx),
         s_remote_cnt = s_remote_cnt + par_s_remote(idx),
         s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                     THEN s_quantity +91
                     ELSE s_quantity
                  END) - par_ol_quantity(idx)
         WHERE i_id = par_ol_i_id(idx)
         AND s_w_id = par_ol_supply_w_id(idx)
         RETURNING i_price, i_name, s_quantity, s_dist_01,
                i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                   par_ol_amount,par_brand_generic;
   END;
-- +++++++ u1
WHEN 2 THEN
-- +++++++ u2
   BEGIN
        -- we found savings of ~900 SPARC instructions when using
        -- o_ol_cnt as a host bind instead of storing it in a variable.
       FORALL idx IN 1 .. par_o_ol_cnt
         UPDATE /*+ VECTOR_READ */ stock_item
         SET s_order_cnt = s_order_cnt + 1,
         s_ytd = s_ytd + par_ol_quantity(idx),
         s_remote_cnt = s_remote_cnt + par_s_remote(idx),
         s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                     THEN s_quantity +91
                     ELSE s_quantity
                  END) - par_ol_quantity(idx)
         WHERE i_id = par_ol_i_id(idx)
         AND s_w_id = par_ol_supply_w_id(idx)
         RETURNING i_price, i_name, s_quantity, s_dist_02,
                i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                   par_ol_amount,par_brand_generic;
   END;
-- +++++++ u2
WHEN 3 THEN
-- +++++++ u3
   BEGIN
        -- we found savings of ~900 SPARC instructions when using
        -- o_ol_cnt as a host bind instead of storing it in a variable.
       FORALL idx IN 1 .. par_o_ol_cnt
         UPDATE /*+ VECTOR_READ */ stock_item
         SET s_order_cnt = s_order_cnt + 1,
         s_ytd = s_ytd + par_ol_quantity(idx),
         s_remote_cnt = s_remote_cnt + par_s_remote(idx),
         s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                     THEN s_quantity +91
                     ELSE s_quantity
                  END) - par_ol_quantity(idx)
         WHERE i_id = par_ol_i_id(idx)
         AND s_w_id = par_ol_supply_w_id(idx)
         RETURNING i_price, i_name, s_quantity, s_dist_03,
                i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                   par_ol_amount,par_brand_generic;
   END;
-- +++++++ u3
```

```
WHEN 4 THEN
-- +++++++ u4
  BEGIN
       -- we found savings of ~900 SPARC instructions when using
       -- o_ol_cnt as a host bind instead of storing it in a variable.
      FORALL idx IN 1 .. par_o_ol_cnt
        UPDATE /*+ VECTOR_READ */ stock_item
        SET s_order_cnt = s_order_cnt + 1,
        s_ytd = s_ytd + par_ol_quantity(idx),
        s_remote_cnt = s_remote_cnt + par_s_remote(idx),
        s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                       THEN s_quantity +91
                       ELSE s_quantity
                  END) - par_ol_quantity(idx)
        WHERE i_id = par_ol_i_id(idx)
        AND s_w_id = par_ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_04,
                i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                     par_ol_amount,par_brand_generic;
  END;
-- +++++++ u4
WHEN 5 THEN
-- +++++++ u5
  BEGIN
       -- we found savings of ~900 SPARC instructions when using
       -- o_ol_cnt as a host bind instead of storing it in a variable.
      FORALL idx IN 1 .. par_o_ol_cnt
        UPDATE /*+ VECTOR_READ */ stock_item
        SET s_order_cnt = s_order_cnt + 1,
        s_ytd = s_ytd + par_ol_quantity(idx),
        s_remote_cnt = s_remote_cnt + par_s_remote(idx),
        s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                       THEN s_quantity +91
                       ELSE s_quantity
                  END) - par_ol_quantity(idx)
        WHERE i_id = par_ol_i_id(idx)
        AND s_w_id = par_ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_05,
                i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                     par_ol_amount,par_brand_generic;
  END;
-- +++++++ u5
WHEN 6 THEN
-- +++++++ u6
  BEGIN
       -- we found savings of ~900 SPARC instructions when using
       -- o_ol_cnt as a host bind instead of storing it in a variable.
      FORALL idx IN 1 .. par_o_ol_cnt
        UPDATE /*+ VECTOR_READ */ stock_item
        SET s_order_cnt = s_order_cnt + 1,
        s_ytd = s_ytd + par_ol_quantity(idx),
        s_remote_cnt = s_remote_cnt + par_s_remote(idx),
        s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                       THEN s_quantity +91
                       ELSE s_quantity
                  END) - par_ol_quantity(idx)
        WHERE i_id = par_ol_i_id(idx)
        AND s_w_id = par_ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_06,
                i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                     par_ol_amount,par_brand_generic;
  END;
-- +++++++ u6
WHEN 7 THEN
-- +++++++ u7
  BEGIN
       -- we found savings of ~900 SPARC instructions when using
       -- o_ol_cnt as a host bind instead of storing it in a variable.
      FORALL idx IN 1 .. par_o_ol_cnt
        UPDATE /*+ VECTOR_READ */ stock_item
        SET s_order_cnt = s_order_cnt + 1,
        s_ytd = s_ytd + par_ol_quantity(idx),
        s_remote_cnt = s_remote_cnt + par_s_remote(idx),
        s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                       THEN s_quantity +91
                       ELSE s_quantity
                  END) - par_ol_quantity(idx)
        WHERE i_id = par_ol_i_id(idx)
        AND s_w_id = par_ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_07,
                i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                     par_ol_amount,par_brand_generic;
  END;
-- +++++++ u7
WHEN 8 THEN
-- +++++++ u8
  BEGIN
       -- we found savings of ~900 SPARC instructions when using
       -- o_ol_cnt as a host bind instead of storing it in a variable.
      FORALL idx IN 1 .. par_o_ol_cnt
        UPDATE /*+ VECTOR_READ */ stock_item
        SET s_order_cnt = s_order_cnt + 1,
        s_ytd = s_ytd + par_ol_quantity(idx),
        s_remote_cnt = s_remote_cnt + par_s_remote(idx),
        s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                       THEN s_quantity +91
                       ELSE s_quantity
                  END) - par_ol_quantity(idx)
        WHERE i_id = par_ol_i_id(idx)
        AND s_w_id = par_ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_08,
                i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                  par_ol_amount,par_brand_generic;
  END;
-- +++++++ u8
WHEN 9 THEN
-- +++++++ u9
  BEGIN
      -- we found savings of ~900 SPARC instructions when using
      -- o_ol_cnt as a host bind instead of storing it in a variable.
     FORALL idx IN 1 .. par_o_ol_cnt
      UPDATE /*+ VECTOR_READ */ stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + par_ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + par_s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                   THEN s_quantity +91
                   ELSE s_quantity
                 END) - par_ol_quantity(idx)
      WHERE i_id = par_ol_i_id(idx)
      AND s_w_id = par_ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_09,
           i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                  par_ol_amount,par_brand_generic;
  END;
-- +++++++ u9
WHEN 10 THEN
-- +++++++ u10
  BEGIN
      -- we found savings of ~900 SPARC instructions when using
      -- o_ol_cnt as a host bind instead of storing it in a variable.
     FORALL idx IN 1 .. par_o_ol_cnt
      UPDATE /*+ VECTOR_READ */ stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + par_ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + par_s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < par_ol_quantity (idx) + 10
                   THEN s_quantity +91
                   ELSE s_quantity
                 END) - par_ol_quantity(idx)
      WHERE i_id = par_ol_i_id(idx)
      AND s_w_id = par_ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_10,
           i_price*par_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 par_i_price, par_i_name,
par_s_quantity, inittpcc.s_dist,
                  par_ol_amount,par_brand_generic;
  END;
-- +++++++ u10
------------- u1 thru u10 --- END ---
              ELSE
                EXIT;
          END CASE;

-- cache the no of rows processed
         dummy_local := sql%rowcount;


-- fix the rows if necessary
         IF (dummy_local != par_o_ol_cnt )  THEN
--   used to be PROCEDURE fix_items IS
   BEGIN
-- gotta shift price, name, s_quantity, brand_generic, s_dist, ol_amount
        idx := 1;
-- found 0 bad rows
        rows_lost := 0;
-- so many rows in out array to begin with
                  max_index := sql%rowcount;

        WHILE (max_index != par_o_ol_cnt) LOOP

-- find item where item ids dont match
                    WHILE (idx <= sql%rowcount AND
                                  sql%bulk_rowcount(idx + rows_lost) = 1)
          LOOP
                     idx := idx + 1;
          END LOOP;

-- shift the items please
          temp_index := max_index;
          WHILE (temp_index >= idx + rows_lost) LOOP
                       par_i_price(temp_index + 1)       :=
par_i_price(temp_index);
             par_i_name(temp_index + 1)        :=  par_i_name(temp_index);
                       par_s_quantity(temp_index + 1)    :=
par_s_quantity(temp_index);
             par_ol_amount(temp_index +1)      :=
par_ol_amount(temp_index);
             inittpcc.s_dist(temp_index + 1) :=  inittpcc.s_dist(temp_index);
                       par_brand_generic(temp_index + 1) :=
par_brand_generic(temp_index);
                       temp_index := temp_index - 1;
          END LOOP;

--  values for the non-existent items if not at end
           IF (idx + rows_lost <= par_o_ol_cnt) THEN
                      par_i_price(idx + rows_lost)     :=   0;
                      par_i_name(idx + rows_lost)      :=   'NO ITEM';
            par_ol_amount(idx + rows_lost)   :=   0;
            par_s_quantity(idx + rows_lost)  :=   0;
            inittpcc.s_dist(idx + rows_lost) := NULL;
            par_brand_generic(idx + rows_lost) := ' ';

--  one more bad row
            rows_lost := rows_lost + 1;
            max_index := max_index + 1;
                    END IF;

        END LOOP;
   END ;
-- end of procedure fix_items;

 END IF;

        FORALL idx IN 1..par_o_ol_cnt
-- doesnt hurt if we insert entries for invalid item too
        INSERT INTO ordl
         VALUES (par_w_id, par_d_id, par_o_id, inittpcc.idx1arr(idx),
par_ol_i_id(idx),
            inittpcc.nulldate, par_ol_amount(idx),
par_ol_supply_w_id(idx),
                  par_ol_quantity(idx),  inittpcc.s_dist(idx));

--If there are no errors, then just return without COMMITing
--The COMMIT is done on the driver side by OCI
-- If there are errors, then rollback and set o_ol_cnt to the processed value
-- note that this is an extra bind ### till we manage to get errors handled
-- properly
            IF (dummy_local != par_o_ol_cnt) THEN
         par_o_ol_cnt := dummy_local;
         ROLLBACK;
       END IF;

       EXIT;

       EXCEPTION
         WHEN not_serializable OR deadlock OR snapshot_too_old THEN
            ROLLBACK;
            par_retry := par_retry + 1;
        END;
     END LOOP;
  END neworder ;

PROCEDURE orderstatus (
    ware_id                INTEGER,
    dist_id                INTEGER,
    cust_id          IN OUT INTEGER,
    bylastname             INTEGER,
    cust_last        IN OUT VARCHAR2,
    cust_first          OUT VARCHAR2,
    cust_middle         OUT VARCHAR2,
    cust_balance        OUT NUMBER,
    ord_id           IN OUT INTEGER,
    ord_entry_d         OUT VARCHAR2,
    ord_carrier_id      OUT INTEGER,
    ord_ol_cnt          OUT INTEGER,
    oline_supply_w_id IN OUT intarray,
    oline_i_id       IN OUT intarray,
    oline_quantity   IN OUT intarray,
    oline_amount     IN OUT numarray,
    oline_delivery_d    OUT datarray
  )
  IS
    cust_rowid             ROWID;
    ol                     BINARY_INTEGER;
    c_num                  BINARY_INTEGER;
    row_id                 rowidarray;
    not_serializable       EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock               EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old       EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
    CURSOR o_cur IS
      SELECT ol_i_id, ol_supply_w_id, ol_quantity, ol_amount,
           nvl(ol_delivery_d,to_date('15-09-1911','DD-MM-YYYY'))
del_date
      FROM ordl
      WHERE ol_d_id = dist_id AND ol_w_id = ware_id AND ol_o_id =
ord_id;
    CURSOR c_cur IS
      SELECT rowid
      FROM cust
      WHERE c_d_id = dist_id AND c_w_id = ware_id AND c_last =
cust_last
      ORDER BY c_w_id, c_d_id, c_last, c_first;
  BEGIN

    LOOP BEGIN

      IF bylastname != 0 THEN

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

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

      ELSE

        SELECT c_balance, c_first, c_middle, c_last
          INTO cust_balance, cust_first, cust_middle, cust_last
          FROM cust
          WHERE c_id = cust_id AND c_d_id = dist_id AND c_w_id =
ware_id;

      END IF;

-- AVLIET added the rownum=1 clause to select only one ORDER
-- according to TPCC-spec (2.6.2.2) largest order_id must be selected
      SELECT o_id,
                   to_char(o_entry_d, 'DD-MM-YYYY.HH24:MI:SS'),
          nvl(o_carrier_id,0), o_ol_cnt
        INTO ord_id,
                   ord_entry_d,
                   ord_carrier_id, ord_ol_cnt
        FROM ordr
       WHERE o_d_id = dist_id AND o_w_id = ware_id AND o_c_id =
cust_id
            AND rownum = 1
       ORDER BY o_w_id, o_d_id, o_c_id, o_id DESC;


      ol := 0;
      FOR o_cur_rec IN o_cur LOOP
        ol := ol + 1;
        oline_i_id(ol) := o_cur_rec.ol_i_id;
        oline_supply_w_id(ol) := o_cur_rec.ol_supply_w_id;
        oline_quantity(ol) := o_cur_rec.ol_quantity;
        oline_amount(ol) := o_cur_rec.ol_amount;
        oline_delivery_d(ol) := o_cur_rec.del_date;
      END LOOP;

      COMMIT;
      EXIT;

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

  END orderstatus;

PROCEDURE delivery (
    ware_id                  IN          INTEGER,
    dist_id           IN OUT intarray,
    order_id                 OUT         intarray,
    ordcnt            OUT    INTEGER,
    sums   OUT        intarray,
    del_date                 IN          DATE,
    carrier_id        IN     INTEGER,
    order_c_id        OUT    intarray,
    retry                       IN OUT     INTEGER
) IS
BEGIN
  LOOP BEGIN
    FORALL d IN 1..10
      DELETE  /* index_asc (nord inord) */ FROM nord N
        WHERE no_d_id = inittpcc.dist(d)
          AND no_w_id = ware_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 dist_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 = dist_id(o)
        AND o_w_id = ware_id
      RETURNING o_c_id BULK COLLECT INTO order_c_id;

    FORALL o in 1.. ordcnt
      UPDATE ordl SET ol_delivery_d = del_date
      WHERE ol_w_id = ware_id
        AND ol_d_id = dist_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 = ware_id
         AND c_d_id = dist_id(c)
         AND c_id = order_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 delivery;

PROCEDURE payment (
      ware_id                 INTEGER,
      dist_id                 INTEGER,
      cust_w_id               INTEGER,
      cust_d_id               INTEGER,
      cust_id          IN OUT INTEGER,
      bylastname              INTEGER,
      hist_amount             NUMBER,
      cust_last        IN OUT VARCHAR2,
      ware_street_1       OUT VARCHAR2,
      ware_street_2       OUT VARCHAR2,
      ware_city           OUT VARCHAR2,
      ware_state          OUT VARCHAR2,
      ware_zip            OUT VARCHAR2,
      dist_street_1       OUT VARCHAR2,
      dist_street_2       OUT VARCHAR2,
      dist_city           OUT VARCHAR2,
      dist_state          OUT VARCHAR2,
      dist_zip            OUT VARCHAR2,
      cust_first          OUT VARCHAR2,
      cust_middle         OUT VARCHAR2,
      cust_street_1       OUT VARCHAR2,
      cust_street_2       OUT VARCHAR2,
      cust_city           OUT VARCHAR2,
      cust_state          OUT VARCHAR2,
      cust_zip            OUT VARCHAR2,
      cust_phone          OUT VARCHAR2,
      cust_since          OUT DATE,
      cust_credit      IN OUT VARCHAR2,
      cust_credit_lim     OUT NUMBER,
      cust_discount       OUT NUMBER,
      cust_balance     IN OUT NUMBER,
      cust_data           OUT VARCHAR2,
      cr_date          IN     DATE,
      retry            IN OUT INTEGER
    )
    IS
      TYPE rowidarray IS TABLE OF ROWID INDEX BY
BINARY_INTEGER;
      cust_rowid              ROWID;
      dist_name               VARCHAR2(11);
      ware_name               VARCHAR2(11);
      c_num                   BINARY_INTEGER;
      row_id                  rowidarray;
      not_serializable        EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
      deadlock                EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old        EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
      CURSOR c_cur IS
        SELECT rowid
        FROM cust
        WHERE c_d_id = cust_d_id AND c_w_id = cust_w_id AND c_last =
cust_last
        ORDER BY c_w_id, c_d_id, c_last, c_first;
    BEGIN
      LOOP BEGIN

        IF bylastname != 0 THEN

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

          UPDATE cust
            SET c_balance = c_balance - hist_amount,
                c_ytd_payment = c_ytd_payment + hist_amount,
                c_payment_cnt = c_payment_cnt + 1
           WHERE rowid = 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 cust_id, cust_first, cust_middle, cust_last, cust_street_1,
                cust_street_2, cust_city, cust_state, cust_zip, cust_phone,
                cust_since, cust_credit, cust_credit_lim, cust_discount,
                cust_balance;

        ELSE

          UPDATE cust
            SET c_balance = c_balance - hist_amount,
                c_ytd_payment = c_ytd_payment + hist_amount,
                c_payment_cnt = c_payment_cnt + 1
            WHERE c_id = cust_id AND c_d_id = cust_d_id AND
```

```
                c_w_id = cust_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 cust_rowid, cust_first, cust_middle, cust_last,
               cust_street_1, cust_street_2, cust_city, cust_state,
               cust_zip, cust_phone, cust_since, cust_credit,
               cust_credit_lim, cust_discount, cust_balance;
       END IF;

       IF cust_credit = 'BC' THEN

         UPDATE cust
           SET c_data = substr ((to_char (cust_id) || ' ' ||
                          to_char (cust_d_id) || ' ' ||
                          to_char (cust_w_id) || ' ' ||
                          to_char (dist_id) || ' ' ||
                          to_char (ware_id) || ' ' ||
                          to_char (hist_amount, '9999.99') || ' | ')
                         || c_data, 1, 500)
           WHERE rowid = cust_rowid
        RETURNING substr (c_data, 1, 200)
           INTO cust_data;
       ELSE
         cust_data := ' ';
       END IF;


       UPDATE dist
         SET d_ytd = d_ytd + hist_amount
        WHERE d_id = dist_id
         AND d_w_id = ware_id
     RETURNING d_name, d_street_1, d_street_2, d_city,d_state, d_zip
          INTO dist_name,dist_street_1,dist_street_2,dist_city,
              dist_state, dist_zip;

       UPDATE ware
         SET w_ytd = w_ytd + hist_amount
        WHERE w_id = ware_id
     RETURNING w_name, w_street_1, w_street_2, w_city, w_state,
w_zip
          INTO ware_name, ware_street_1, ware_street_2, ware_city,
              ware_state, ware_zip;


      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
         (cust_id, cust_d_id, cust_w_id, dist_id, ware_id, cr_date,
          hist_amount, ware_name || '    ' || dist_name);

      COMMIT;
      EXIT;

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

     END LOOP;
   END payment;

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

    LOOP BEGIN

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

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

END tpcc;
/
show errors


- - - - - - - - - - - - - - - - - -
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;
```

```
- - - - - - - - - - - - - - - - - -
tpcc.h
- - - - - - - - - - - - - - - - - -
/*
 * $Header: tk_perf/benchmark_kits/tpcc-new/benchrun/source/server/tpcc.h /main/1 2008/12/15 05:58:52 avliet Exp $ 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;


/* 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 tkvcninit ();
extern int tkvcpinit ();
extern int tkvcoinit ();
extern int tkvcdinit ();
extern int tkvcsinit ();

extern int tkvcn ();
extern int tkvcp ();
extern int tkvco ();
extern int tkvcd ();
extern int tkvcs ();

extern void tkvcndone ();
extern void tkvcpdone ();
extern void tkvcodone ();
extern void tkvcddone ();
extern void tkvcsdone ();

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(char *fname, int lineno,OCIError *errhp, sword status);
extern int sqlfile(char *fname, text *linebuf);

extern FILE *lfp;
extern FILE *fopen ();
extern int proc_no;
extern int doid[];

extern int execstatus;
extern int errcode;

extern OCIEnv *tpcenv;
extern OCIServer *tpcsrv;
extern OCIError *errhp;
extern OCISvcCtx *tpcsvc;
extern OCISession *tpcusr;
extern OCIStmt *curntest;
/* The bind and define handles for each transaction are
   included in their respective header files. */



/* for stock-level transaction */

extern int w_id;
extern int d_id;
extern int c_id;
#ifdef USE_IEEE_NUMBER
extern float threshold;
#else
extern int threshold;
#endif /* USE_IEEE_NUMBER */
extern int low_stock;

/* for delivery transaction */

extern int del_o_id[10];
extern int carrier_id;
extern int retries;

/* for order-status transaction */

extern int bylastname;
extern char c_last[17];
extern char c_first[17];
extern char c_middle[3];
extern double c_balance;
extern int o_id;
extern text o_entry_d[20];
extern int o_carrier_id;
extern int o_ol_cnt;
extern int ol_supply_w_id[15];
extern int ol_i_id[15];
#ifdef USE_IEEE_NUMBER
extern float ol_quantity[15];
extern float ol_amount[15];
#else
extern int ol_quantity[15];
extern int ol_amount[15];
#endif /* USE_IEEE_NUMBER */
ub4    ol_del_len[15];
extern text ol_delivery_d[15][11];
/* xnie - begin */
extern OCIRowid *o_rowid;
/* xnie - end */

/* for payment transaction */

extern int c_w_id;
extern int c_d_id;
#ifdef USE_IEEE_NUMBER
extern float h_amount;
#else
extern int h_amount;
#endif /* USE_IEEE_NUMBER */
extern char w_street_1[21];
extern char w_street_2[21];
extern char w_city[21];
extern char w_state[3];
extern char w_zip[10];
extern char d_street_1[21];
extern char d_street_2[21];
extern char d_city[21];
extern char d_state[3];
extern char d_zip[10];
extern char c_street_1[21];
extern char c_street_2[21];
extern char c_city[21];
extern char c_state[3];
extern char c_zip[10];
extern char c_phone[17];
extern text c_since_d[11];
extern char c_credit[3];
extern int c_credit_lim;
extern float c_discount;
extern char c_data[201];
extern text h_date[20];

/* for new order transaction */

extern int nol_i_id[15];
extern int nol_supply_w_id[15];
#ifdef USE_IEEE_NUMBER
extern float nol_quantity[15];
extern float nol_amount[15];
extern float s_quantity[15];
extern float i_price[15];
#else
extern int nol_quantity[15];
extern int nol_amount[15];
extern int s_quantity[15];
extern int i_price[15];
#endif /* USE_IEEE_NUMBER */
extern int nol_quanti10[15];
extern int nol_quanti91[15];
extern int nol_ytdqty[15];
extern int o_all_local;
extern float w_tax;
extern float d_tax;
extern float total_amount;
extern char i_name[15][25];
extern int i_name_strlen[15];
extern ub2 i_name_strlen_len[15];
extern ub2 i_name_strlen_rcode[15];
extern ub4 i_name_strlen_csize;
extern char brand_gen[15];
extern ub2 brand_gen_len[15];
extern ub2 brand_gen_rcode[15];
extern ub4 brand_gen_csize;
extern char brand_generic[15][1];
extern int status;
extern int tracelevel;

/* Miscellaneous */
extern OCIDate cr_date;
extern OCIDate c_since;
extern OCIDate o_entry_d_base;
extern OCIDate ol_d_base[15];

#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(__FILE__,__LINE__,(errp),(function));

#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
      ocierror(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
      ocierror(__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(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
      DISCARD ocierror(__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(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
      DISCARD ocierror(__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(__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(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
   DISCARD ocierror(__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(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0)); \
      DISCARD ocierror(__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(__FILE__,__LINE__, (errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0));\
      DISCARD ocierror(__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(__FILE__,__LINE__, (errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid*
*)0));\
      ocierror(__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(__FILE__,__LINE__,(errp), \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                                                  (dvoid**)0));\
      ocierror(__FILE__,__LINE__,(errp), \
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),
(progvl),(ftype),\
                          (indp),NULL,NULL, OCI_DYNAMIC_FETCH));\
      ocierror(__FILE__,__LINE__,(errp), \
      OCIDefineDynamic((dfnp),(errp),(ctxp),(cbf_data)));




/* New order */

struct newinstruct {
   int w_id;
   int d_id;
   int c_id;
   int ol_i_id[15];
   int ol_supply_w_id[15];
   int ol_quantity[15];
};

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];
   float total_amount;
   char i_name[15][25];
   int s_quantity[15];
   char brand_generic[15];
   float i_price[15];
   float ol_amount[15];
   char status[26];
   int retry;
};

struct newstruct {
   struct newinstruct newin;
   struct newoutstruct newout;
};


/* Payment */

struct payinstruct {
   int w_id;
   int d_id;
   int c_w_id;
   int c_d_id;
   int c_id;
   int bylastname;
   int h_amount;
   char c_last[17];
};

struct payoutstruct {
   int terror;
   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];
   char c_since[11];
   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 {
   struct payinstruct payin;
   struct payoutstruct payout;
};


/* Order status */

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;
   char o_entry_d[20];
   int o_carrier_id;
   int o_ol_cnt;
   int ol_supply_w_id[15];
   int ol_i_id[15];
   int ol_quantity[15];
   float ol_amount[15];
   char ol_delivery_d[15][11];
   int retry;
};

struct ordstruct {
   struct ordinstruct ordin;
   struct ordoutstruct ordout;
};


/* Delivery */

struct delinstruct {
   int w_id;
   int o_carrier_id;
   double qtime;
   int in_timing_int;
   int plsqlflag;
};

struct deloutstruct {
   int terror;
   int retry;
};

struct delstruct {
   struct delinstruct delin;
   struct deloutstruct delout;
};


/* Stock level */

struct stoinstruct {
   int w_id;
   int d_id;
   int threshold;
};

struct stooutstruct {
   int terror;
   int low_stock;
   int retry;
};

struct stostruct {
   struct stoinstruct stoin;
   struct stooutstruct stoout;
};

#endif
```

```
- - - - - - - - - - - - - - - - - -
tpccload.c
- - - - - - - - - - - - - - - - - -
#ifdef RCSID
static char *RCSid =
   "$Header: tk_perf/benchmark_kits/tpcc-
new/benchrun/source/server/tpccload.c /main/1 2008/12/15 05:58:52 avliet
Exp $ 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;
   long loopcount;
   int cid;
   int dwid;
   int cdid;
   int cwid;
   int sid;
   int swid;
   int olcnt;
   long nrows;
   long 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 = (long)eware - (long)bware + 1;

    fprintf (stderr, "Loading/generating warehouse: w%d - w%d (%ld
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.  %ld 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 = ((long)eware - (long)bware + 1) * DISTFAC;

    fprintf (stderr, "Loading/generating district: w%d - w%d (%ld 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.  %ld 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 = ((long)eware - (long)bware + 1) * CUSTFAC * DISTFAC;

    fprintf (stderr, "Loading/generating customer: w%d - w%d (%ld
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 = ((long)eware - (long)s_c_w_id + 1) * DISTFAC * CUSTFAC
- ((long)s_c_d_id - 1) * CUSTFAC - (long)s_c_id + 1;
    fprintf (stderr, "remaining rows: %ld\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 %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, " %ld rows committed\n    ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %ld 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 = ((long)ecid - (long)bcid + 1) * ((long)eware - (long)bware +1)
* DISTFAC;

      fprintf (stderr, "Loading/generating customer: c%d - c%d, w%d - w%d
(%ld 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 = ((long)ecid - (long)s_c_id + 1) * ((long)eware - (long)bware +
1) * DISTFAC - ((long)s_c_w_id - 1) * DISTFAC - (long)s_c_d_id + 1;
   fprintf (stderr, "remaining rows: %ld\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, " %ld rows committed\n   ", row);
   }

   end_time = gettime ();
   end_cpu = getcpu ();
   fprintf (stderr, "Done.  %ld 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: (%ld 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, " %ld rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %ld 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 = ((long)eware - (long)bware + 1) * STOCFAC;

    fprintf (stderr, "Loading/generating stock: w%d - w%d (%ld 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 = ((long)eware - (long)s_s_w_id + 1) * STOCFAC - (
(long)s_s_i_id - 1);
    fprintf (stderr, "remaining rows: %ld\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 %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, " %ld rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %ld 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 = ((long)eitem - (long)bitem + 1) * ((long)eware - (long)bware +
1);

    fprintf (stderr, "Loading/generating stock: i%d - i%d, w%d - w%d (%ld
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 = ((long)eitem - (long)s_s_i_id + 1) * ((long)eware -
(long)bware
+ 1) - ((long)s_s_w_id - (long)bware);
    fprintf (stderr, "remaining rows: %ld\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 %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, " %ld rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %ld 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 = ((long)eware - (long)bware + 1) * HISTFAC;

      fprintf (stderr, "Loading/generating history: w%d - w%d (%ld 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, " %ld rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %ld 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 = ((long)eware - (long)bware + 1) * ORDEFAC * DISTFAC;

      fprintf (stderr, "Loading/generating orders and order-line: w%d - w%d
(%ld ord, ~%ld 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,
                        cdid, 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,
                        cdid, 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, " %ld orders committed\n   ", row);
        }
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %ld 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 = ((long)eware - (long)bware + 1) * NEWOFAC * DISTFAC;
    fprintf (stderr, "Loading/generating new-order: w%d - w%d (%ld
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, " %ld rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %ld 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);
}
```

```
- - - - - - - - - - - - - - - - - -
views.sql
- - - - - - - - - - - - - - - - - -
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 /*+ leading(s) use_nl(i) */
 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;
```

- - - - - - - - - - - - - - - - - -
seed 70
- - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - -
switchlog.sh
- - - - - - - - - - - - - - - - - -

```
#!/bin/sh

sqlplus / as sysdba <<!
startup pfile=p_build.ora
quit;
!

(
sqlplus / as sysdba << !
ALTER DATABASE ADD LOGFILE group 3
('/home/oracle/disks/log_1_3') SIZE 2G reuse;
quit;
!
) &

(
sqlplus / as sysdba << !
ALTER DATABASE ADD LOGFILE group 4
('/home/oracle/disks/log_2_3') SIZE 2G reuse;
quit
!
) &

wait

./droplog.sh 1
./droplog.sh 2


(
sqlplus / as sysdba << !
ALTER DATABASE ADD LOGFILE group 1
('/home/oracle/disks/log_1_1', '/home/oracle/disks/log_2_1') SIZE 164G
reuse;
quit;
!
) &

(
sqlplus / as sysdba << !
ALTER DATABASE ADD LOGFILE group 2
('/home/oracle/disks/log_1_2', '/home/oracle/disks/log_2_2') SIZE 164G
reuse;
quit
!
) &

Wait
./droplog.sh 3
./droplog.sh 4

sqlplus / as sysdba <<!
shutdown immediate;
quit;
```

- - - - - - - - - - - - - - - - - -
droplog.sh
- - - - - - - - - - - - - - - - - -

```
rm -f /tmp/tkvswitchlog$1.out

sqlplus -s "/as sysdba" << !
set echo off
set term off
set head off
spool /tmp/tkvswitchlog$1.out
select 'STATUS',status from v\$log where group#=$1;
spool off
exit
!

s2=`grep '^STATUS' /tmp/tkvswitchlog$1.out | awk '{print $2}'`

c=""
if [ "$s2" = "CURRENT" -o "$s2" = "ACTIVE" -o "$s2" = "UNUSED" ]
then
  c="true";
fi


while test ! -z "$c"
do

  rm -f /tmp/tkvswitchlog$1.out
sqlplus "/as sysdba" << !
  alter system switch logfile;
  set echo off
  set term off
  set head off
  spool /tmp/tkvswitchlog$1.out
  select 'STATUS',status from v\$log where group#=$1;
  spool off
  exit
!
  s2=`grep '^STATUS' /tmp/tkvswitchlog$1.out | awk '{print $2}'`
  c=""
  if [ "$s2" = "CURRENT" -o "$s2" = "ACTIVE" -o "$s2" = "UNUSED" ]
  then
    c="true"
  fi

done
```

11 Appendix C: Tunable Parameters

```
- - - - - - - - - - - - - - - - - -
p_build.ora
- - - - - - - - - - - - - - - - - -
compatible = 11.2.0.0.1
control_files = (/dev/vol_one//control_001, /dev/vol_one//control_002)
db_16k_cache_size = 87381M
db_8k_cache_size = 32768M
db_block_size = 4096
db_cache_size = 87381M
db_files = 653
db_name = tpcc
log_buffer = 1048576
plsql_optimize_level=2
shared_pool_size = 16384M
statistics_level = basic
dml_locks = 500
transactions = 600
undo_management = auto
undo_retention = 2
processes = 600
recovery_parallelism = 40
sessions = 600
parallel_max_servers = 100
UNDO_TABLESPACE = undo_1
db_2k_cache_size = 20M


- - - - - - - - - - - - - - - - - -
p_create.ora
- - - - - - - - - - - - - - - - - -
compatible = 11.2.0.0.1
control_files = (/dev/vol_one//control_001, /dev/vol_one//control_002)
db_16k_cache_size = 87381M
db_8k_cache_size = 32768M
db_block_size = 4096
db_cache_size = 87381M
db_files = 653
db_name = tpcc
log_buffer = 1048576
plsql_optimize_level=2
shared_pool_size = 16384M
statistics_level = basic
undo_management = manual
db_2k_cache_size = 20M


- - - - - - - - - - - - - - - - - -
p_run.ora
- - - - - - - - - - - - - - - - - -
compatible = 11.2.0.0.1
control_files = (/dev/vol_one//control_001, /dev/vol_one//control_002)
db_cache_size = 12000m
db_16k_cache_size = 94668m
db_keep_cache_size = 255260m
db_recycle_cache_size = 6000m
db_8k_cache_size = 2048m
db_writer_processes=3
log_checkpoint_timeout = 0
log_checkpoints_to_alert = TRUE
log_checkpoint_interval = 0
log_buffer = 1048576
shared_pool_size = 8000m
db_block_size = 4096
db_files = 1016
db_name = tpcc
plsql_optimize_level=2
statistics_level = basic
dml_locks = 500
transactions = 600
undo_management = auto
undo_retention = 1
processes = 600
sessions = 600
recovery_parallelism = 0
parallel_max_servers = 0
undo_tablespace = undo_1
aq_tm_processes = 0
disk_asynch_io = true
job_queue_processes = 0
query_rewrite_enabled = false
replication_dependency_tracking = false
pga_aggregate_target = 0
java_pool_size = 0
transactions_per_rollback_segment = 1
fast_start_mttr_target = 0
db_block_checking = false
db_block_checksum = false
audit_trail = false
resource_manager_plan = ''
trace_enabled = false
result_cache_max_size = 0
timed_statistics=false


- - - - - - - - - - - - - - - - - -
rc.local
- - - - - - - - - - - - - - - - - -
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local

echo 190600 > /proc/sys/vm/nr_hugepages
echo "512 32000 512 128" > /proc/sys/kernel/sem
echo 412316860416 > /proc/sys/kernel/shmmax

- - - - - - - - - - - - - - - - - -
sysctl.conf
- - - - - - - - - - - - - - - - - -
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0
```

```
# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0

# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename.
# Useful for debugging multi-threaded applications.
kernel.core_uses_pid = 1

# Controls the use of TCP syncookies
net.ipv4.tcp_syncookies = 1

# Disable netfilter on bridges.
net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0

# Controls the maximum size of a message, in bytes
kernel.msgmnb = 65536

# Controls the default maxmimum size of a mesage queue
kernel.msgmax = 65536

# Controls the maximum shared segment size, in bytes
kernel.shmmax = 68719476736

# Controls the maximum number of shared memory segments, in pages
kernel.shmall = 4294967296


fs.aio-max-nr = 4194304
net.core.wmem_max  = 4194304
net.core.wmem_default  = 4194304
net.core.rmem_max  = 4194304
net.core.rmem_default  = 4194304

kernel.sem = 250        32000        128 128
fs.nr_open = 4194304


fs.file-max = 6815744
```

12 Appendix D: Price Quotations

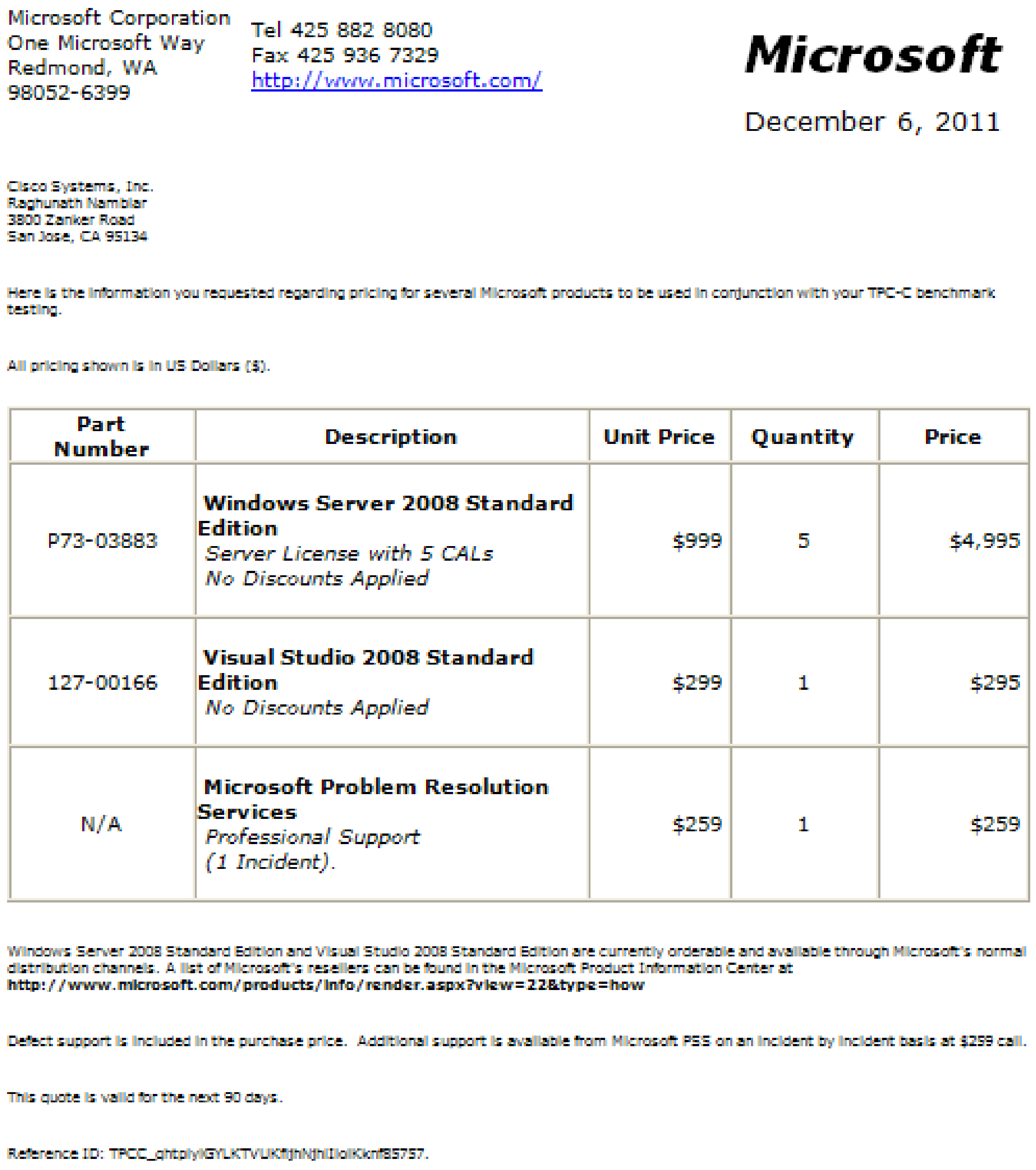
From: MaryBeth Pierantoni [mailto:mary.beth.pierantoni@oracle.com]

To: Raghunath Nambiar

Pricing:

Oracle Linux Basic Limited (3 year Support)	$1497	1	$1497
Oracle Database Standard Edition One, Per Processor for 3 years	$2900	2	$5800
Incident Server Support for 3 years	$2300	3	$6900
Total			$14197

Contact: MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com



Microsoft Corporation
One Microsoft Way
Redmond, WA
98052-6399

Tel 425 882 8080
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

December 6, 2011

Cisco Systems, Inc.
Raghunath Nambiar
3800 Zanker Road
San Jose, CA 95134

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 ($).

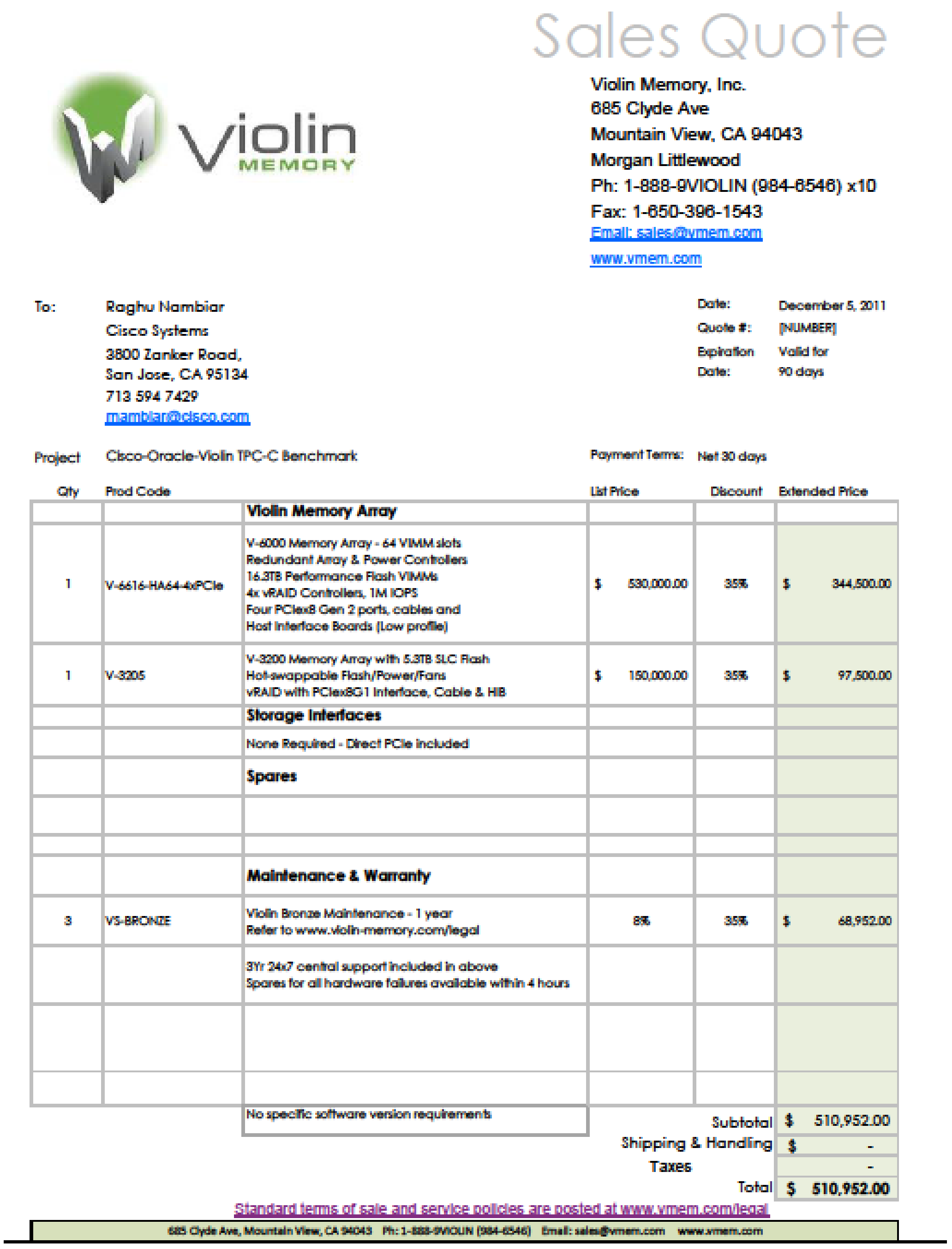
Part Number	Description	Unit Price	Quantity	Price
P73-03883	**Windows Server 2008 Standard Edition** *Server License with 5 CALs* *No Discounts Applied*	$999	5	$4,995
127-00166	**Visual Studio 2008 Standard Edition** *No Discounts Applied*	$299	1	$295
N/A	**Microsoft Problem Resolution Services** *Professional Support* *(1 Incident).*	$259	1	$259

Windows Server 2008 Standard Edition and Visual Studio 2008 Standard Edition are currently orderable and available through Microsoft's normal distribution channels. A list of Microsoft's resellers can be found in the Microsoft Product Information Center at **http://www.microsoft.com/products/info/render.aspx?view=22&type=how**

Defect support is included in the purchase price. Additional support is available from Microsoft PSS on an incident by incident basis at $259 call.

This quote is valid for the next 90 days.

Reference ID: TPCC_qhtplylGYLKTVUKfIjhNjhlIIolKknf85757.



Sales Quote

Violin Memory, Inc.
685 Clyde Ave
Mountain View, CA 94043
Morgan Littlewood
Ph: 1-888-9VIOLIN (984-6546) x10
Fax: 1-650-396-1543
Email: sales@vmem.com
www.vmem.com

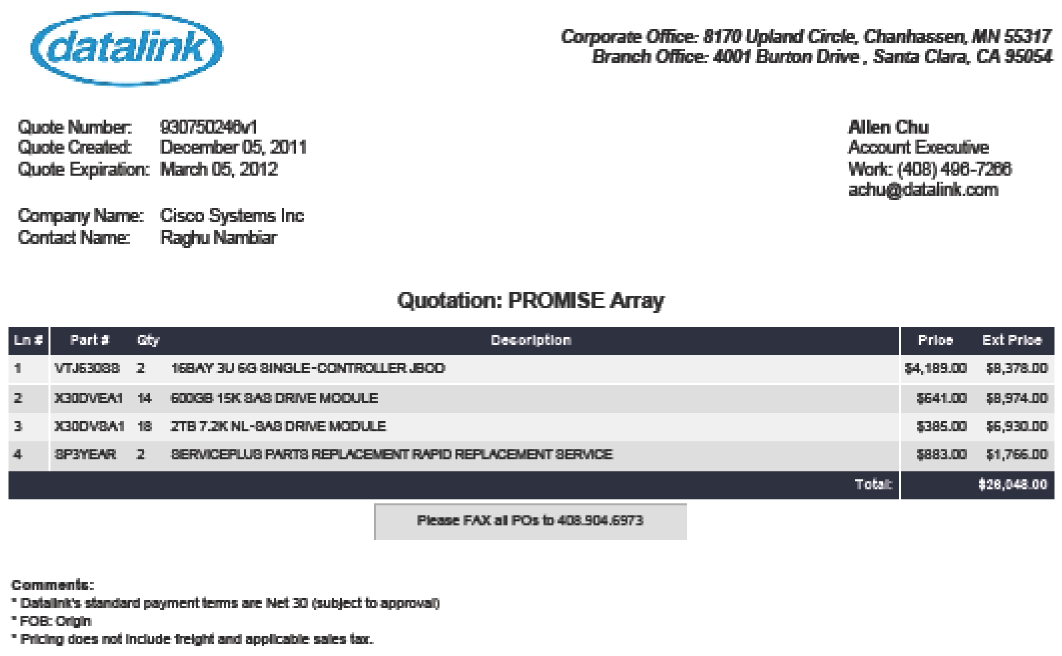
To:	Raghu Nambiar Cisco Systems 3800 Zanker Road, San Jose, CA 95134 713 594 7429 rnambiar@cisco.com	Date:	December 5, 2011
		Quote #:	[NUMBER]
		Expiration Date:	Valid for 90 days
Project	Cisco-Oracle-Violin TPC-C Benchmark	Payment Terms:	Net 30 days

Qty	Prod Code		List Price	Discount	Extended Price
		Violin Memory Array			
1	V-6616-HA64-4xPCIe	V-6000 Memory Array - 64 VIMM slots Redundant Array & Power Controllers 16.3TB Performance Flash VIMMs 4x vRAID Controllers, 1M IOPS Four PCIex8 Gen 2 ports, cables and Host Interface Boards (Low profile)	$ 530,000.00	35%	$ 344,500.00
1	V-3205	V-3200 Memory Array with 5.3TB SLC Flash Hot-swappable Flash/Power/Fans vRAID with PCIex8G1 Interface, Cable & HIB	$ 150,000.00	35%	$ 97,500.00
		Storage Interfaces			
		None Required - Direct PCIe included			
		Spares			
		Maintenance & Warranty			
3	VS-BRONZE	Violin Bronze Maintenance - 1 year Refer to www.violin-memory.com/legal	8%	35%	$ 68,952.00
		3Yr 24x7 central support included in above Spares for all hardware failures available within 4 hours			
		No specific software version requirements			

Subtotal	$ 510,952.00
Shipping & Handling	$ -
Taxes	-
Total	**$ 510,952.00**

Standard terms of sale and service policies are posted at www.vmem.com/legal

685 Clyde Ave, Mountain View, CA 94043 Ph: 1-888-9VIOLIN (984-6546) Email: sales@vmem.com www.vmem.com



Corporate Office: 8170 Upland Circle, Chanhassen, MN 55317
Branch Office: 4001 Burton Drive , Santa Clara, CA 95054

Quote Number:	930750246v1
Quote Created:	December 05, 2011
Quote Expiration:	March 05, 2012
Company Name:	Cisco Systems Inc
Contact Name:	Raghu Nambiar

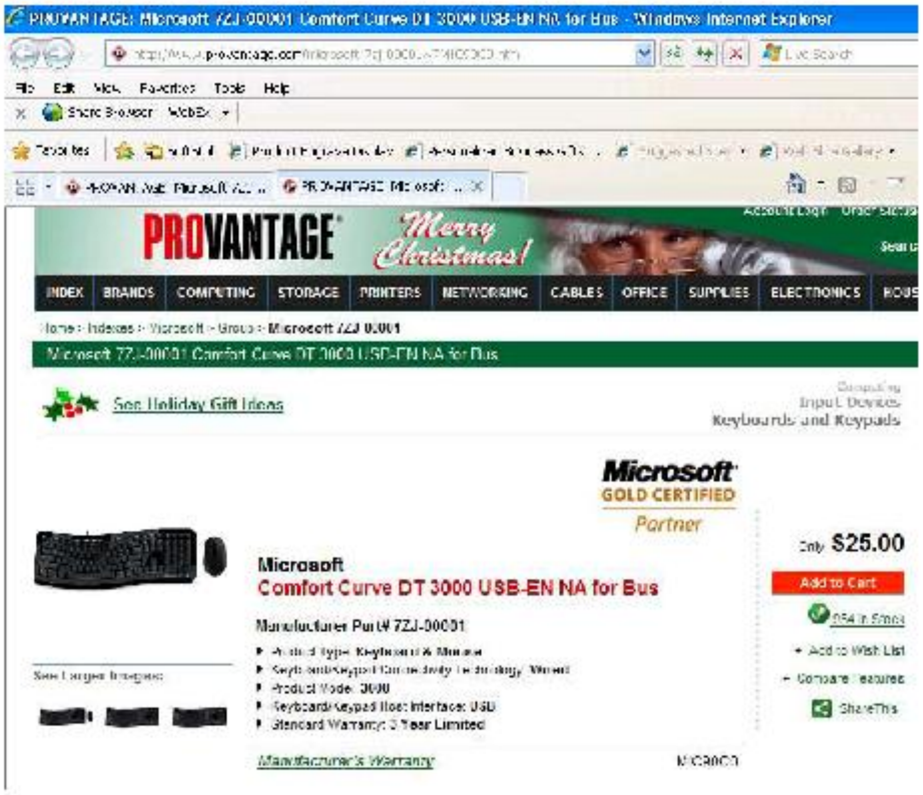
Allen Chu
Account Executive
Work: (408) 496-7266
achu@datalink.com

Quotation: PROMISE Array

Ln #	Part #	Qty	Description	Price	Ext Price
1	VTJ630SS	2	16BAY 3U 6G SINGLE-CONTROLLER JBOD	$4,189.00	$8,378.00
2	X30DVEA1	14	600GB 15K SAS DRIVE MODULE	$641.00	$8,974.00
3	X30DVSA1	18	2TB 7.2K NL-SAS DRIVE MODULE	$385.00	$6,930.00
4	SP3YEAR	2	SERVICEPLUS PARTS REPLACEMENT RAPID REPLACEMENT SERVICE	$883.00	$1,766.00
				Total:	$26,048.00

Please FAX all POs to 408.904.6973

Comments:
* Datalink's standard payment terms are Net 30 (subject to approval)
* FOB: Origin
* Pricing does not include freight and applicable sales tax.



PROVANTAGE: Microsoft 7ZJ-00001 Comfort Curve DT 3000 USB-EN NA for Bus - Windows Internet Explorer

Home > Indexes > Microsoft > Group > Microsoft 7ZJ-00001

Microsoft 7ZJ-00001 Comfort Curve DT 3000 USB-EN NA for Bus

See Holiday Gift Ideas

Computing
Input Devices
Keyboards and Keypads

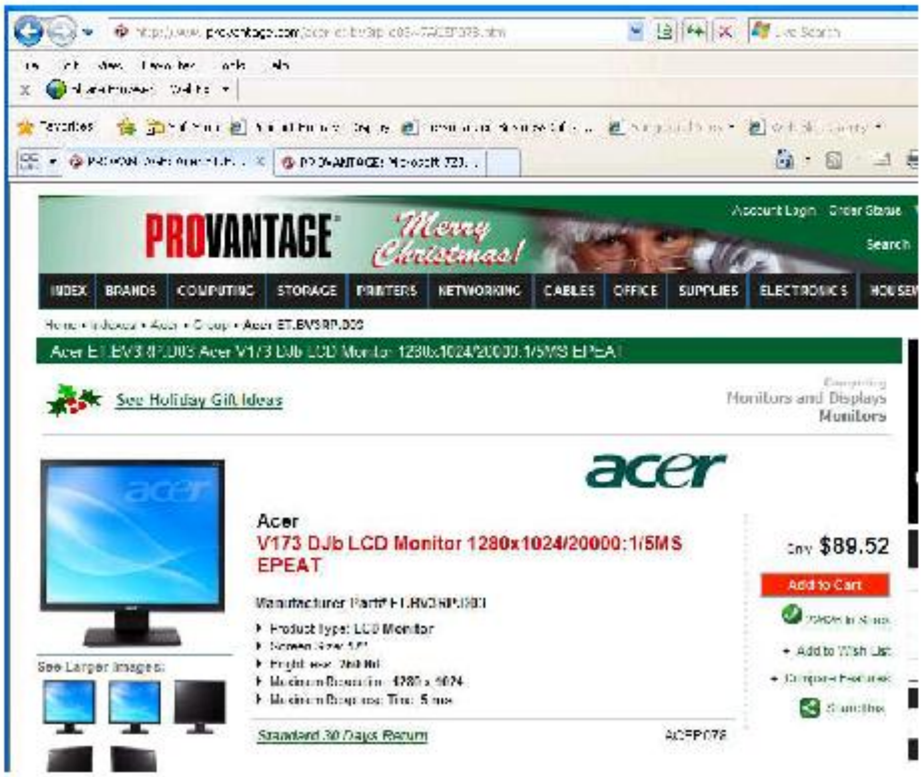
Microsoft
Comfort Curve DT 3000 USB-EN NA for Bus

Manufacturer Part# 7ZJ-00001

- Product Type: Keyboard & Mouse
- Keyboard/Keypad Connectivity Technology: Wired
- Product Model: 3000
- Keyboard/Keypad Host Interface: USB
- Standard Warranty: 3 Year Limited

Manufacturer's Warranty

Only $25.00



Home • Indexes • Acer • Group • Acer ET.BV3RP.003

Acer ET.BV3RP.003 Acer V173 DJb LCD Monitor 1280x1024/20000:1/5MS EPEAT

See Holiday Gift Ideas

Monitors and Displays
Monitors

Acer
V173 DJb LCD Monitor 1280x1024/20000:1/5MS EPEAT

Manufacturer Part# ET.BV3RP.003

- Product Type: LCD Monitor
- Screen Size: 17"
- Brightness: 250 Nit
- Maximum Resolution: 1280 x 1024
- Maximum Response Time: 5 ms

Standard 30 Days Return

ACEP078

$89.52