Hewlett-Packard Company

TPC Benchmark™ C
Full Disclosure Report
for

HP ProLiant ML350 G6

Using
Oracle Database 11g Release 2 Standard Edition One and
Oracle Enterprise Linux

First Edition
August 2010

First Edition – August 16, 2010

Hewlett Packard Company (HP) believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. HP assumes no responsibility for any errors that may appear in this document. The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, HP provides no warranty of the pricing information in this document.

Benchmark results are highly dependent upon workload, specific application requirements, and system design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC Benchmark 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.

All performance data contained in this report were obtained in a rigorously controlled environment. Results obtained in other operating environments may vary significantly. HP does not warrant or represent that a user can or will achieve similar performance expressed in transactions per minute (tpmC) or normalized price/performance ($/tpmC). No warranty of system performance or price/performance is expressed or implied in this report.

Printed in U.S.A., 2010

Parallel Database Cluster Model PDC and ProLiant are registered trademarks of Hewlett Packard Company.

ORACLE 11g, Pro*C, PL/SQL, SQL*Net, SQL*Plus are registered trademarks of Oracle Corporation.

TPC Benchmark is a trademark of the Transaction Processing Performance Council.

All other brand or product names mentioned herein must be considered trademarks or registered trademarks of their respective owners.

Table of Contents

Preface

The TPC Benchmark C was developed by the Transaction Processing Performance Council (TPC). The TPC was founded to define transaction processing benchmarks and to disseminate objective, verifiable performance data to the industry. This full disclosure report is based on the TPC Benchmark C Standard Specifications Version 5.10.1, released February 2009.

TPC Benchmark C Overview

The TPC describes this benchmark in Clause 0.1 of the specifications as follows:

TPC Benchmark C is an On Line Transaction Processing (OLTP) workload. It is 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.

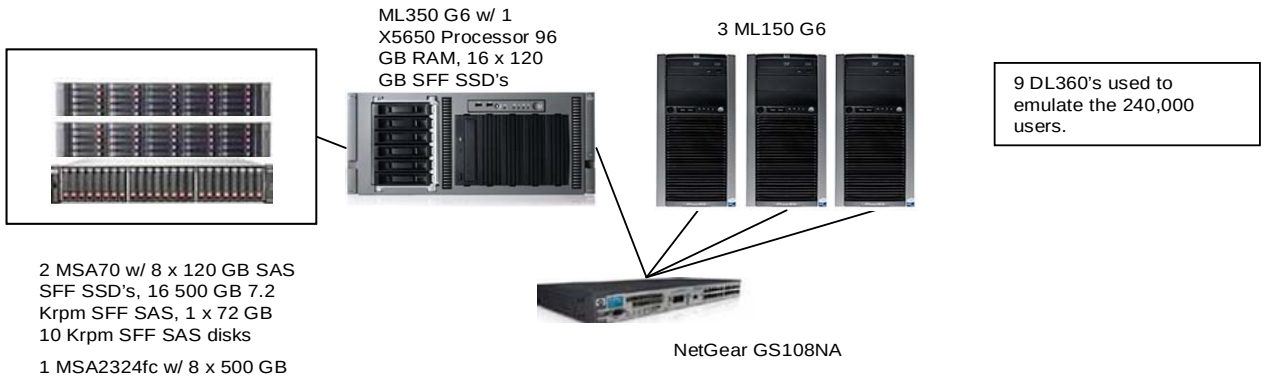
Despite the fact that this benchmark offers a rich environment that emulates many OLTP applications, this benchmark does not reflect the entire range of OLTP requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C 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.

	HP ProLiant ML350 G6 2.67 GHz 12MB L2 **C/S with 3 ProLiant ML150 G6**	**TPC-C Version 5.11** **TPC-Pricing 1.5.0** **TPC-Energy 1.1.1** **Report Date: August 16, 2010**

Total System Cost	TPC-CThroughput	Price/Performance	Availability Date	TPC-Energy Metric
$113,012 USD	**290,040 tpmC**	**$0.39 USD/tpmC**	**August 6, 2010**	**4.22 Watts/KtpmC**

Processors	Database Manager	Operating System	Other Software	Number of Users
1/6/12 Intel Xeon 2.67 GHz 12MB L2 cache	Oracle Database 11g Release 2 Standard Edition One	Oracle Enterprise Linux	Microsoft COM+	**240,000**



	Server		Each Client	
System Components	Quantity	Description	Quantity	Description
Processor	1/6/12	1/6/12 Intel Xeon 2.67 GHz 12MB L2 cache	1/4/4	2.13 GHz Intel Xeon w/ 4MB L2 Cache
Memory	96 GB	6x16GB	2.0	2x1024 MB
Disk Controllers	1 3 2	HP P410i FBWC HP P410 FBWC HP P411 FBWC	1	Smart Array B110i SATA
Disk Drives	24 24 1	120 GB SSD's 500 GB 7.2 Krpm SFF SAS drives (data) 72 GB 10K SFF SAS drives (OS)	1	250 GB NHP SATA
Total Storage	13,928.3			

Hewlett-Packard Company	HP ProLiant ML350 G6 2.67 GHz w/ 12 MB L3 C/S with 3 ProLiant ML150 G6	TPC-C Rev. 5.11 TPC-Pricing 1.5.0 TPC-Energy 1.1.1 **Report Date: August 16, 2010**

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date	TPC-Energy Metric
USD $113,012	**290 KtpmC**	**USD $0.39**	**August 6, 2010**	**4.22 watts/ KtpmC**

Numerical Quanties For Reported Energy Configuration:

REC Idle Power: 1,141.5 watts

Average Power of REC: 1,223.94 watts

Lowest ambient temperature at air inlet: 20.88°C

Items in Priced Configuration not in the Reported Energy Configuration:

None

Items in Reported Energy Configuration not in the Measured Energy Configuration:

4 HP LE1851w 18.5-Inch wide Monitor Part Number NK033AA#ABA

Hewlett-Packard Company	HP ProLiant ML350G6	TPC-C Rev. 5.11	
		Report Date	16-Aug-10

Description	Part Number	Pricing	Unit Price	Qty	Extended Price	3 yr. Maint. Price
Server Hardware						
HP ProLiant ML350G6 SFF SAS/SATA Tower CTO Chassis	483447-B21	1	607	1	607	
HP X5650 ML350 G6 Kit	601240-B21	1	1,399	1	1,399	
HP 16GB 1x16GB PC3-8500 Registered CAS 7 Quad Rank x4 DRAM Memory Kit	500666-B21	1	1,549	6	9,294	
HP 512MB Flash Backed Write Cache	534916-B21	1	429	1	429	
HP ML350/370 G6 8 Small Form Factor (SFF) 2nd Drive Cage Kit	507803-B21	1	100	1	100	
HP Half-Height SATA DVD-ROM Optical Drive	447326-B21	1	69	1	69	
HP P411 with 512MB Flash Backed Cache Controller	578229-B21	1	699	2	1,398	
HP P410 with 512MB Flash Backed Cache Controller	578230-B21	1	699	3	2,097	
HP LE1851w 18.5-Inch wide Monitor	NK033AA#ABA	1	159	1	159	
HP 5642 Pallet Unassembled Rack	358254-B21	1	865	1	865	
HP 750W Common Slot High Efficiency Power Supply Kit	512327-B21	1	200	1	200	
HP 72GB 3G SAS 15K rpm SFF (2.5-inch) Dual Port Enterprise 3yr Warranty Hard Drive	418371-B21	1	379	1	379	
HP 500GB 6G SAS 7.2K rpm SFF (2.5-inch) Dual Port Midline 1yr Warranty Hard Drive	507610-B21	1	419	24	10,056	
HP 120GB 3G SATA SFF (2.5-inch) Midline Solid State Drive	572073-B21	1	2,709	24	65,016	
D2700 Disk Enclosure	AJ941A	1	3,399	2	6,798	
MSA2324fc	AJ797A	1	8,900	1	8,900	
HP StorageWorks FC2142SR 4Gb PCIe Host Bus Adapter	A8002A	1	1,140	1	1,140	
HP 3y 4h 24x7 MSA2000 Array HWSupp	UJ675E	1	1,513	1		1,513
HP 3y SupportPlus24 w/DMR D2000 Encl SVC,D2000 Enclosures,4h 24x7 onsite response	UQ105E	1	2,147	2		4,294
HP 3y 4h 24x7 ProLiant ML350 HW Support	U4513E	1	595	1		595
				Subtotal	**108,906**	**6,402**
Server Software						
Oracle Enterprise Linux		6	0	1	0	
Oracle Unbreakable Linux Support: Enterprise Linux Basic Limited for 3 years		2	1,497	1		1,497
Oracle Database 11g Release 2 Standard Edition One, Unlimited Users , 3 years		2	2,900	1	2,900	
Oracle Premium Support for 3 years		2	1,276	3		3,828
				Subtotal	**2,900**	**5,325**
Client Hardware						
HP ProLiant ML150 G6 Smart Buy Intel Xeon E5506 Non-hot Plug SATA Tower Server	518175-005	1	839	3	2,517	
HP NC110T PCI-E Gigabit Server Adapter	434905-B21	1	79	2	158	
HP NC364T Dual Port PCI-E Gigabit Server Adapter	412648-B21	1	229	1	229	
HP LE1851w 18.5-Inch wide Monitor	NK033AA#ABA	1	159	3	477	
HP 3y 4h 24x7 ProLiant ML150 HW Support ,Proliant Server ML150,3 years of hardware.	U8193E	1	500	3		1,500
				Subtotal	**3,381**	**1,500**
Client Software						
Microsoft Problem Resolution Services	N/A	3	245	1		245
Windows Server 2008 R2 Standard Edition	P73-03883	3	1,029	3	3,087	Incl.
Visual Studio Standard 2005	127-00012	3	250	1	250	
				Subtotal	**3,337**	**245**
User Connectivity						
Netgear 8-Port Gigabit Switch, incl. spares	GS108NA	5	76	3	228	
7ft Green Cat 6 Patch Cable, Molded, incl. spares	CB242-7GN	4	2	10	18	
				Subtotal	**246**	**0**
Large Purchase and Net 30 discount (See Note 1)	16.0%	1			**($17,966)**	**($1,264)**
				Total	**$100,804**	**$12,208**

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

Three-Year Cost of Ownership: USD	**$113,012**
tpmC Rating:	**290,040**
$ / tpmC: USD	**$0.39**

Pricing: 1=HP Direct 800-203-6748 2=Oracle 3= Microsoft 4= www.deepsurplus.com 5= www.serversdirect.com
Note 1 = Discount based on HP Direct guidence applies to all lines where pricing = 1
Note 6=Oracle Enterprise Linux is a free download
The benchmark results were audited by Lorna Livingtree of Performance Metrics

Numerical Quantities Summary

MQTH, Computed Maximum Qualified Throughput 290,040 tpmC

Response Times (in seconds)	Average	90%	Maximum
Menu	0.62	1.02	4.94
New-Order	0.70	1.12	5.30
Payment	0.66	1.06	5.25
Order-Status	0.68	1.09	3.91
Delivery (interactive portion)	0.61	1.02	3.33
Delivery (deferred portion)	0.05	0.11	0.80
Stock-Level	1.01	1.50	4.76

Transaction Mix, in percent of total transaction	
New-Order	44.996%
Payment	43.002%
Order-Status	4.001%
Delivery	4.001%
Stock-Level	4.001%

Emulation Delay (in seconds)	Resp.Time	Menu
New-Order	0.10	0.10
Payment	0.10	0.10
Order-Status	0.10	0.10
Delivery (interactive)	0.10	0.10
Stock-Level	0.10	0.10

Keying/Think Times (in seconds)	Min.	Average	Max.
New-Order	18.01/0.00	18.01/12.05	18.01/120.49
Payment	3.01/0.00	3.01/12.01	3.02/120.10
Order-Status	2.01/0.00	2.01/10.01	2.01/100.08
Delivery (interactive)	2.01/0.00	2.01/5.02	2.02/50.19
Stock-Level	2.01/0.00	2.01/5.01	2.02/50.09

Test Duration	
Ramp-up time	73.6 minutes
Measurement interval	120.0 minutes
Transactions (all types) completed during measurement interval	77,351,351
Ramp down time	253.9 minutes

Checkpointing	
Number of checkpoints in measurement interval	4
Checkpoint interval (average)	26:18

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark C test conducted on the hp ProLiant ML350 G6. The operating system used for the benchmark was Oracle Enterprise Linux. The DBMS used was Oracle Database 11g Standard Edition One.

TPC Benchmark C Metrics

The standard TPC Benchmark C metrics, tpmC (transactions per minute), price per tpmC (three year capital cost per measured tpmC), and the availability date are reported as:

290,040 tpmC
$0.39 USD per tpmC
4.22 watts/KtpmC
Available as of August 16, 2010.

Standard and Executive Summary Statements

The following pages contain an executive summary of results for this benchmark.

Auditor

The benchmark configuration, environment and methodology were audited by Lorna Livingtree of Performance Metrics Inc. to verify compliance with the relevant TPC specifications.

General Items

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 all source code implemented in this benchmark.

Test Sponsor

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

This benchmark was sponsored by Hewlett Packard Company. The benchmark was developed and engineered by Hewlett Packard Company and Oracle Corporation. Testing took place at HP Performance Engineering Laboratory in Houston, Texas.

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.

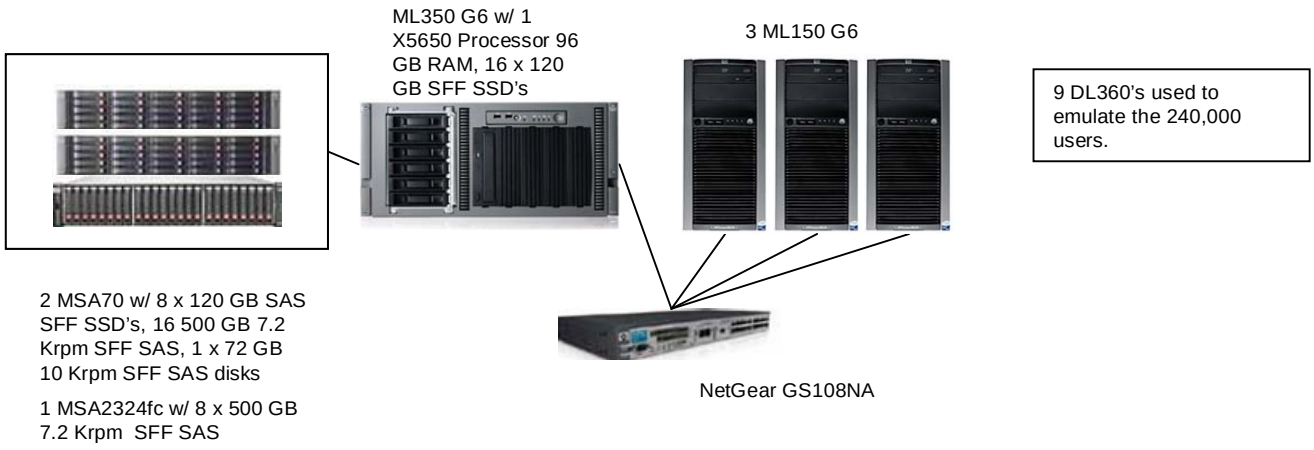
Appendix C contains the tunable parameters for the database, the operating system, and the transaction monitor.

Configuration Items

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 are the same and included below.

Figure 1. Benchmarked and Priced Configuration



Clause 1 Related Items

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.

Physical Organization of Database

The physical organization of tables and indices within the database must be disclosed.
24 Solid State Disks (SSD's) used in the benchmark had a capacity of 111.8 GB, and 24 disks used in the benchmark had a capacity of 146.8 GB 10K rpm.

Controller	Slot	#disks	Objects
P410i	0	4 120GB SFF SSD's	database tables/indexes
P410	1	4 120GB SFF SSD's	database tables/indexes
P410	2	4 120GB SFF SSD's	database tables/indexes
P410	3	4 120 GB SFF SSD's	database tables/indexes
P411	4	4 120 GB SFF SSD's	database tables/indexes
P411	4	16 500 GB SFF SAS 7.2K rpm	backup
P411	5	4 120 GB SFF SSD's	database tables/indexes
P411	5	1 72 GB SFF SAS 10K rpm	O/S
FC2142SR	6	8 500 GB SFF SAS 7.2K rpm	DB logs

Priced Configuration:

All hardware and software remained the same between the benchmarked and priced configurations.

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.

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.

None.

Replication, Duplication or Additions

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.

Clause 2 Related Items

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.

Input/Output Screen Layout

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

All screen layouts followed the specifications exactly.

Priced Terminal Feature Verification

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.

Presentation Manager or Intelligent Terminal

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

Application code running on the client machines implemented the TPC-C user interface. No presentation manager software or intelligent terminal features were used. The source code for the forms applications is listed in Appendix A.

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	85.01%
	Remote warehouse	14.99%
	Accessed by last name	60.00%
Order Status	Accessed by last name	59.99%
Delivery	Skipped transactions	none
Transaction Mix	New Order	44.996%
	Payment	43.002%
	Order status	4.001%
	Delivery	4.001%
	Stock level	4.001%

Queuing Mechanism

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

Microsoft COM+ on each client system served as the queuing mechanism to the database. Each delivery request was submitted to Microsoft COM+ asynchronously with control being returned to the client process immediately and the deferred delivery part completing asynchronously.

Clause 3 Related Items

Transaction System Properties (ACID)

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 were successful. The executions are described below.

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.

Completed Transactions

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.

Aborted Transactions

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.

Consistency

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

Consistency conditions one through four were tested using a shell script to issue queries to the database. The results of the queries verified that the database was consistent for all four tests.

A run was executed under full load over two hours with checkpoints.

The shell script was executed again. The result of the same queries verified that the database remained consistent after the run.

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.

Isolation tests one through nine were executed using shell scripts to issue queries to the database. Each included timestamps to demonstrate the concurrency of operations. The results of the queries were captured to files. The captured files were verified by the auditor 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.

Durability

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

Durable Media Failure

The durable media failure was demonstrated on the full configuration but using only 125,000 active users.

Loss of Log and Data

To demonstrate recovery from a permanent failure of durable medium containing TPC-C tables, the following steps were executed:

1. All partitions on a controller were backed up.
2. The total number of New Orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving the beginning count. Consistency check 3 was verified before run.
3. The RTE was started with 240,000 users
4. The test was allowed to run for a minimum of 5 minutes.
5. A disk was removed from the MSA2324fc that was a log disk.
6. The system continued running due to the fact that the logs are on raid10 devices.
7. The run was allowed to continue for 5 minues.
8. A disk was removed from the array of disks that was backed up.
9. Oracle recorded I/O errors and crashed. The database and the RTE were then shut down.
10. New drives replaced the removed ones, and reconfigured.
11. The database partitions which were backed up in Step 1 were restored.
12. The database was then started. The database was opened and Oracle performed instance recovery.
13. Consistency conditions were executed and verified.
14. Step 2 was repeated and the difference between the first and second counts was noted.
15. An RTE report was generated for the entire run time giving the number of NEW-ORDERS successfully returned to the RTE.
16. The counts in step 14 and 15 were compared and the results verified that all committed transactions had been successfully recovered.
17. Samples were taken from the RTE files and used to query the database to demonstrate successful transactions had corresponding rows in the ORDER table.

Instantaneous Interruption, Loss of Memory

Because 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 of 8000 warehouses under a full load of 80000 users. 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.
2. The RTE was started with 240,000 users.
3. The test was allowed to run for a minimum of 5 minutes.
4. A system crash and loss of memory were induced by pulling the power plugs out of the computer. No battery backup or Uninterruptible Power Supply (UPS) were used to preserve the contents of memory.
5. The RTE was shutdown.
6. Power was restored and the system restarted.
7. Oracle10g was restarted and performed an automatic recovery.
8. Consistency conditions were executed and verified.
9. Step 1 was repeated and the difference between the first and second counts was noted.
10. An RTE report was generated for the entire run time giving the number of NEW-ORDERS successfully returned to the RTE.
11. The counts in step 9 and 10 were compared and the results verified that all committed transactions had been successfully recovered.
12. Samples were taken from the RTE files and used to query the database to demonstrate successful transactions had corresponding rows in the ORDER table.

Clause 4 Related Items

Initial 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 Number of Rows for Server

Table	Occurrences
Warehouse	24,000
District	240,000
Customer	720,000,000
History	720,000,000
Order	720,000,000
New Order	216,000,000
Order Line	7,200,182,048
Stock	2,400,000,000
Item	100,000
Unused Warehouses	0

Database Layout

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

The database datafiles which contain table and index information are stored on the following hardware

- Four P410 SMART Array SAS RAID Controllers, each of which was attached to 4 x 120 GB SFF Solid State Disks.
- Two P411 SMART Array SAS RAID Controllers, each attached to a D2700 containing four 120 GB SFF Solid State Disks

The database logfiles were stored on eight 500 GB 7.2K rpm SFF SAS disk drives in the MSA2324fc attached to the fibre channel controller in slot 6.
The system O/S was located in the D2700 attached to the controller in slot 5. It consisted of a single 72 GB 10K rpm SFF SAS disk drive for the O/S.

The P411 Smart Array Controller accelerator caches were enabled for 100% write. The MSA2324FC, which contained the log disks, cache configuration was set to fault tolerant active-active. The database logfiles were configured as a single RAID-10 volume.

Section 1.2 of this report details the distribution of database tables and logs across all disks. The code that creates the database and tables are included in Appendix B.

Type of Database

A statement must be provided that describes:

1. *The data model implemented by DBMS used (e.g. relational, network, hierarchical).*
2. *The database interface (e.g. embedded, call level) and access language (e.g. SQL, DL/1, COBOL read/write used to implement the TPC-C 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 10g Edition One is a relational DBMS.

Anonymous block PL/SQL and stored procedures were accessed through the ORACLE Call Interface. Application code is included in Appendix A.

Database Mapping

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

The database was not replicated. The tables were not partitioned.

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.

SEGMENT	BLOCKS	BLOCK_SIZE	REQUIRED	STATIC	DYNAMIC	OVERSIZE
CUSTCLUSTER	267426240	2048	264890214	264890214	0	2536026
DB_STAT	1048576	2048	1048576	1048576	0	0
DISTCLUSTER	522240	2048	270178	270178	0	252062
HIST	32931840	2048	26547462	0	22221648	6384378
ICUST1	1139840	16384	1136150	1136150	0	3690
ICUST2	21719040	2048	21240605	21240605	0	478435
IDIST	66560	2048	63538	63538	0	3022
IITEM	10240	2048	5914	5914	0	4326
IORDR2	19481600	2048	15598317	15598317	0	3883283
ISTOK	3362560	16384	3362150	3362150	0	410
ITEMCLUSTER	10240	2048	8868	8868	0	1372
IWARE	20480	2048	16288	16288	0	4192
NORDCLUSTER_QUEUE	4300800	2048	2928089	2928089	0	1372711
ORDRCLUSTER_QUEUE	56394240	16384	43813447	0	36674202	12580793
STOKCLUSTER	361485810	2048	360360394	360360394	0	1125416
SYSAUX	61440	2048	61440	61440	0	0
SYSTEM	204800	2048	204800	204800	0	0
SYS_IQ0000011854$$	56394240	16384	162767	162767	0	56231473
SYS_IQ0000011858$$	4300800	2048	225238	225238	0	4075562
WARECLUSTER	30720	2048	27502	27502	0	3218

STATIC	DYNAMIC	OVERSIZE	DAILY_GROW	DAILY_SPREAD	SPACE60
1408476994	631230528	1141309862	122879548	0	8781249874

Avg Log Switch Interval	#logfiles Required	Logfile Size (MB)	Log Space Req'd (MB)
1,578	19	43,945	1,669,910

Storage Type	# Disks	Size of Disk (GB)	Total Space GB)	Space Required (GB)
Log	8	465.7	3725.6	1,630.8
Data	16	465.7	7451.2	
Data	24	111.8	2683.2	8374.5

Clause 5 Related Items

Throughput

Measured tpmC must be reported

Measured tpmC 290,040 tpmC
Price per tpmC $0.39 USD per tpmC
Watts per KtpmC 4.11 watts/KtpmC

Response Times

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

Table 5.1: Response Times

Type	Average	Maximum	90th %
New-Order	0.70	5.30	1.12
Payment	0.66	5.25	1.06
Order-Status	0.68	3.91	1.09
Interactive Delivery	0.61	3.33	1.02
Deferred Delivery	0.05	0.80	0.11
Stock-Level	1.01	4.76	1.50
Menu	0.62	4.94	1.02

Keying and Think Times

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

Table 5.2: Keying Times/Think Times

Type	Minimum	Average	Maximum
New-Order	18.01/0.00	18.01/12.05	18.10/120.49
Payment	3.01/0.00	3.01/12.01	3.02/120.10
Order-Status	2.01/0.00	2.01/10.01	2.02/100.08
Interactive Delivery	2.01/0.00	2.01/5.02	2.02/50.19
Stock-Level	2.01/0.00	2.01/5.01	2.02/50.09

Response Time Frequency Distribution Curves and Other Graphs

Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type.
The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction.
Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type.
Keying Time frequency distribution curves (see Clause 5.6.4) must be reported for each transaction type.
A graph of throughput versus elapsed time (see Clause 5.6.5) must be reported for the New-Order transaction.

Figure 5.1: Response Times Frequency Distribution for New Order Transactions

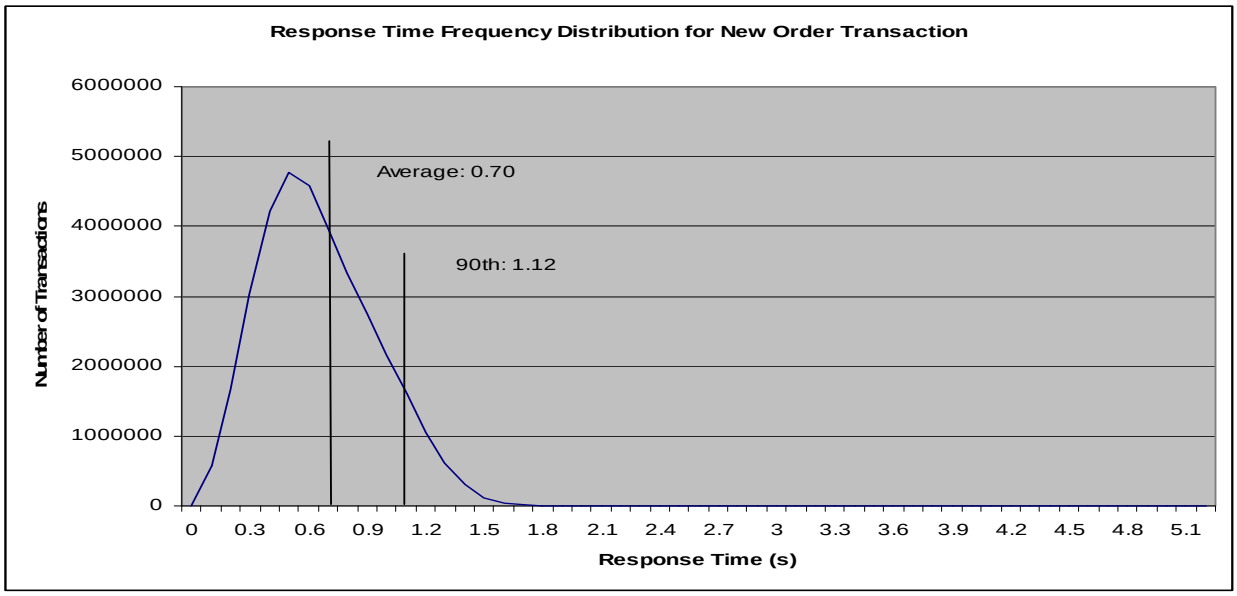


Figure 5.2: Response Times Frequency Distribution for Payment Transactions

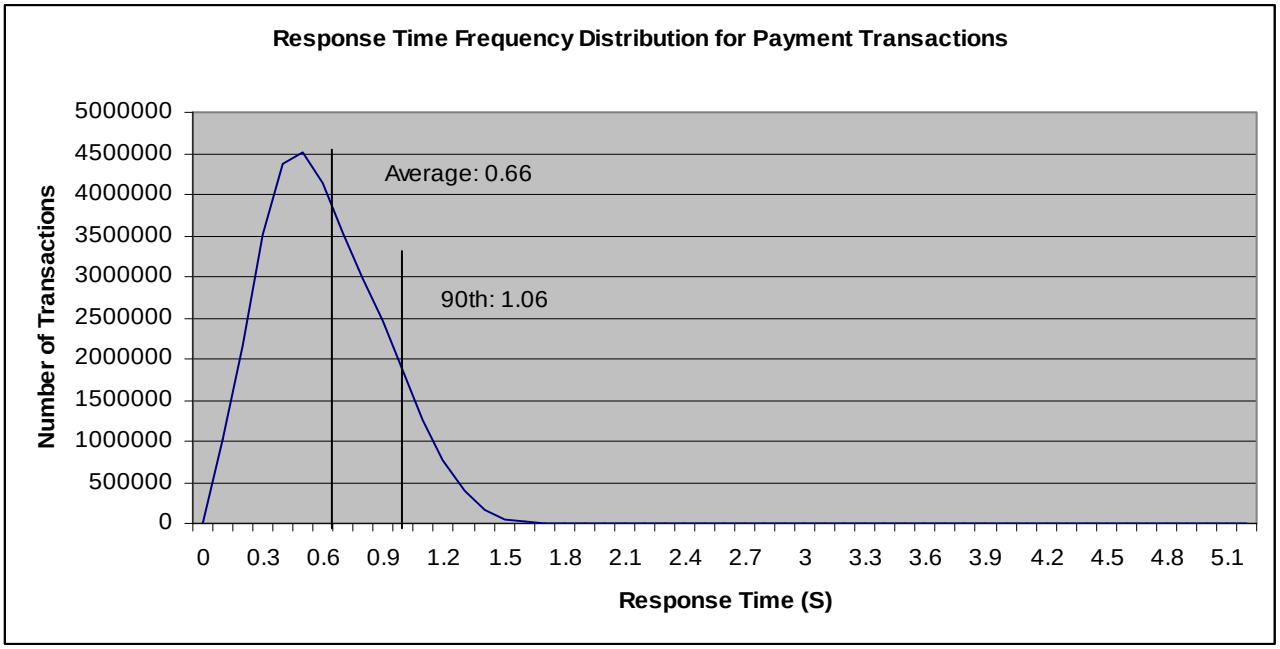


Figure 5.3: Response Times Frequency Distribution for Order Status Transactions

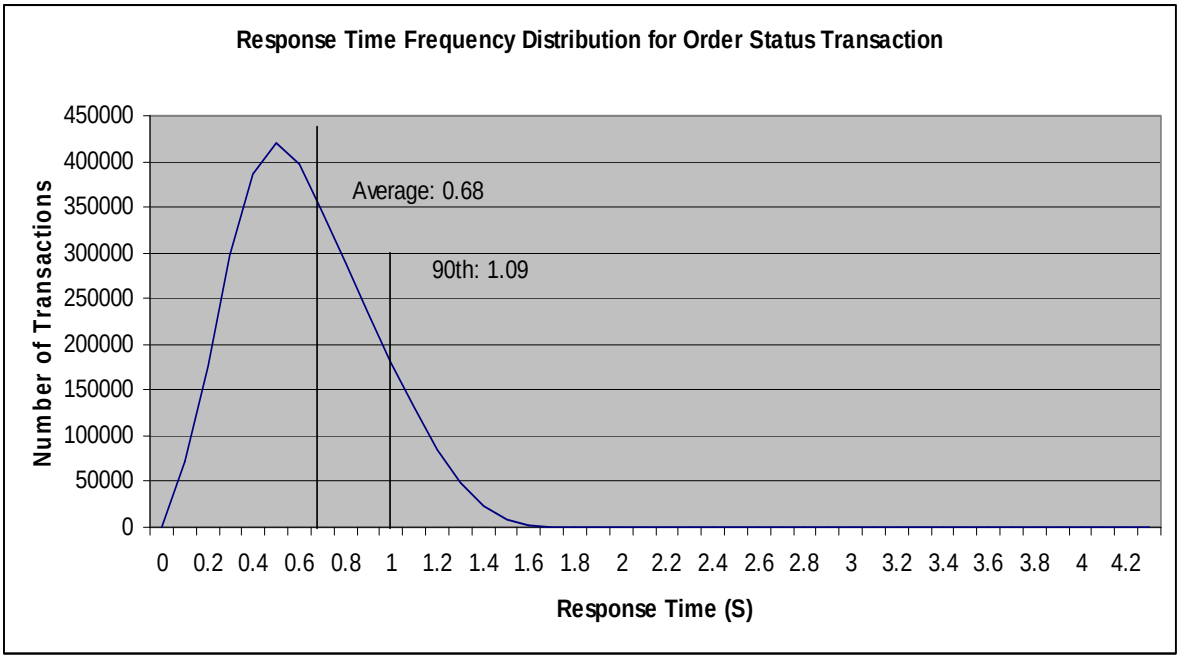


Figure 5.4: Response Times Frequency Distribution for Delivery Transactions

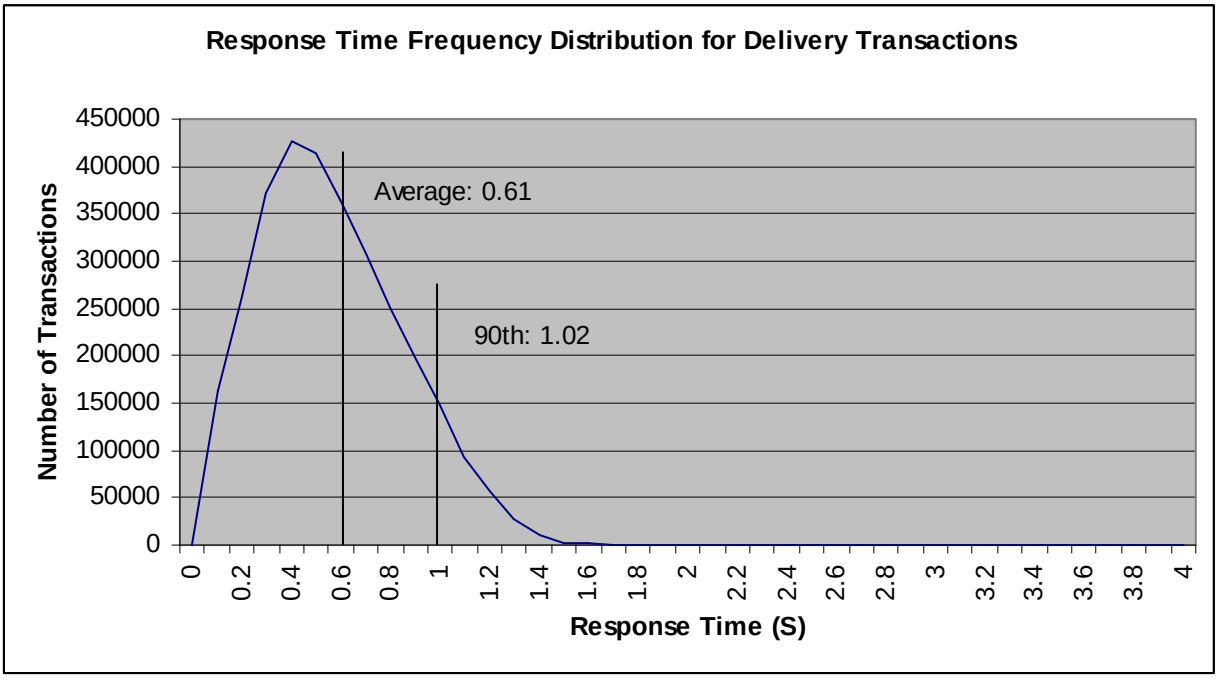


Figure 5.5: Response Times Frequency Distribution for Stock Level Transactions

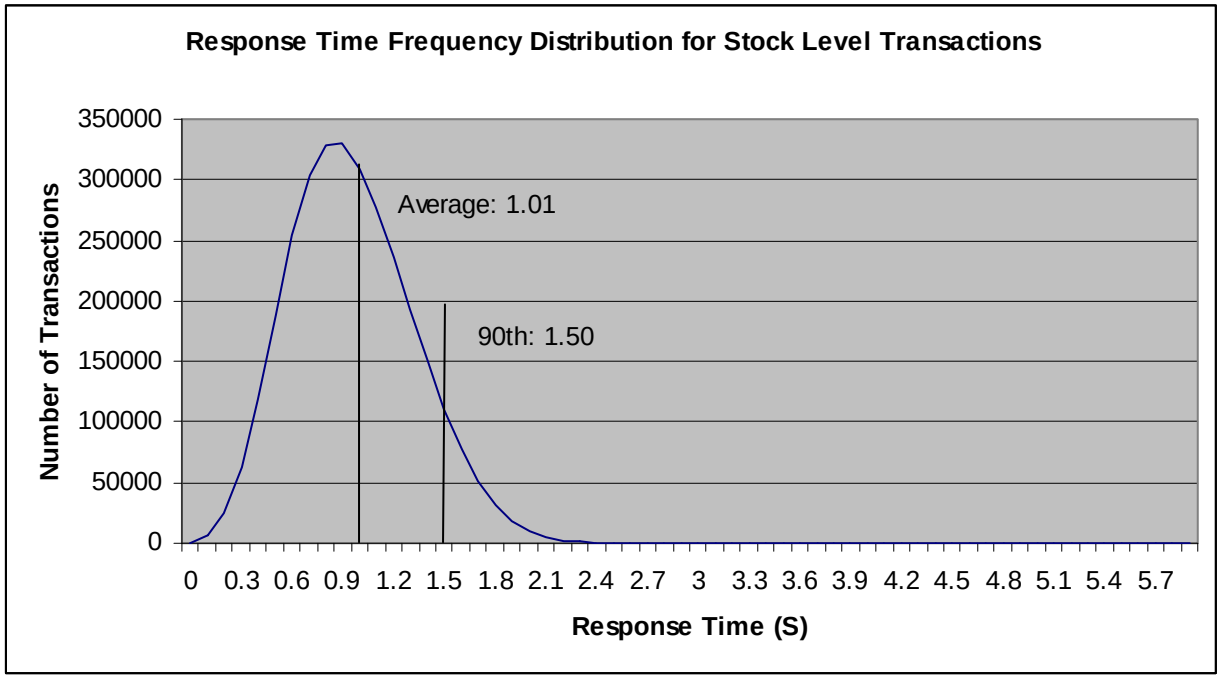


Figure 5.6: Response Time versus Throughput

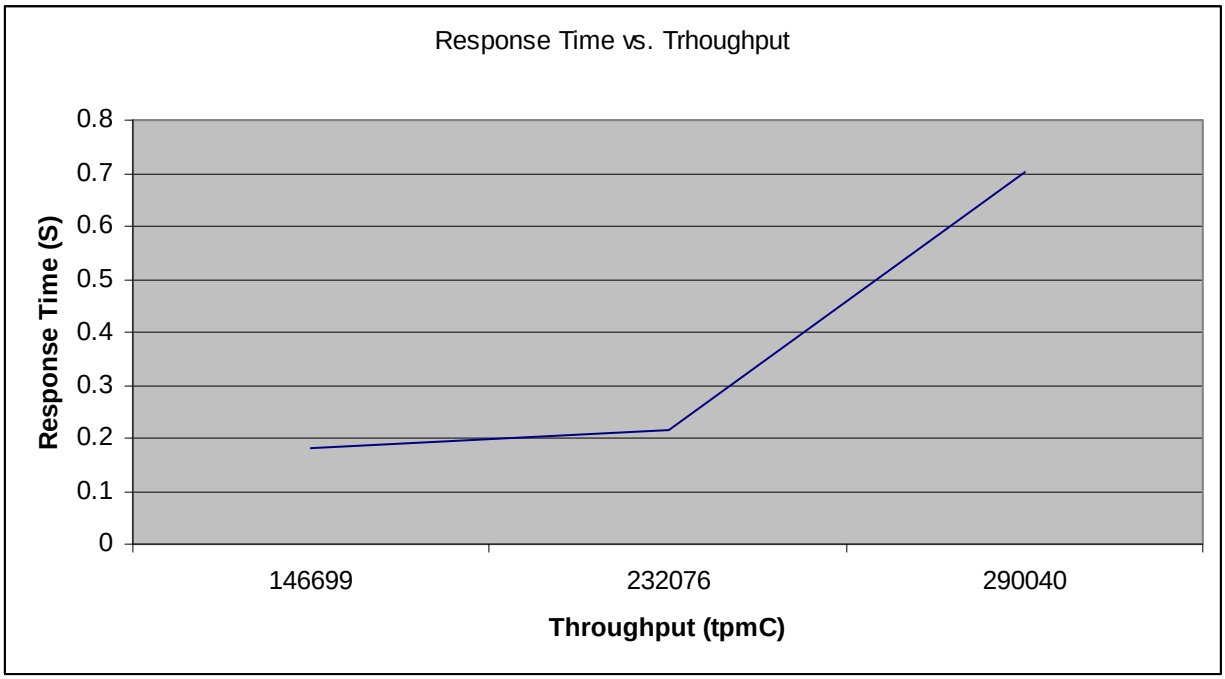


Figure 5.7: Think Times distribution for New Order Transactions

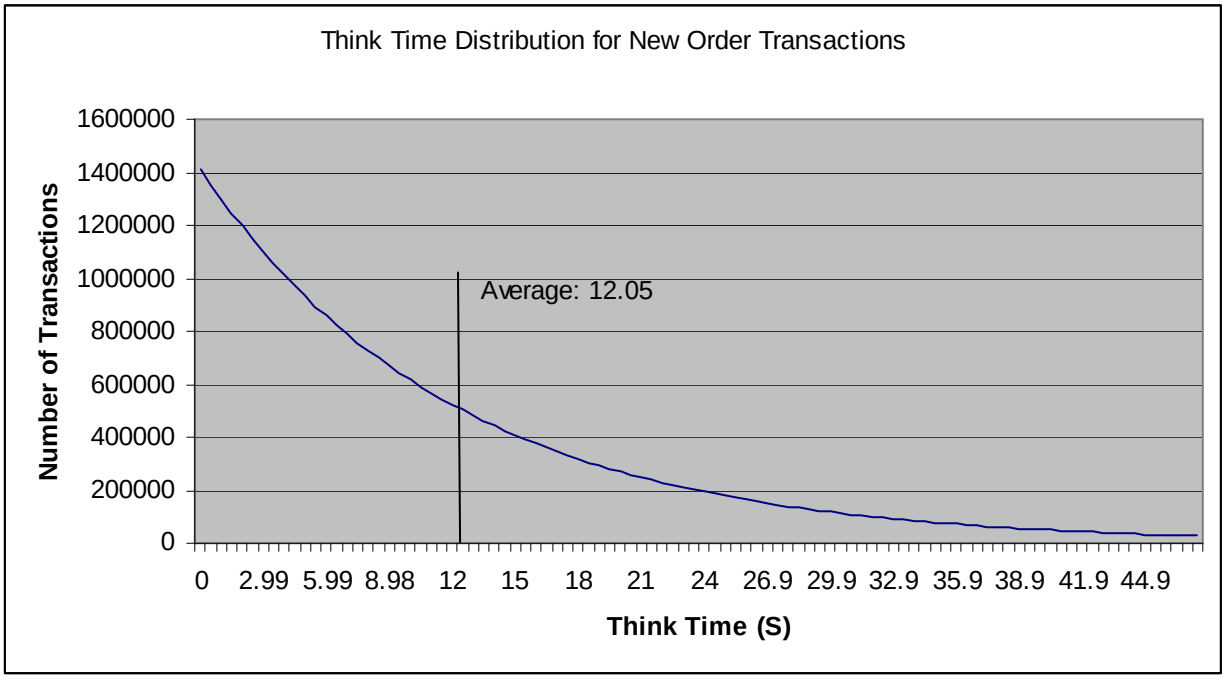
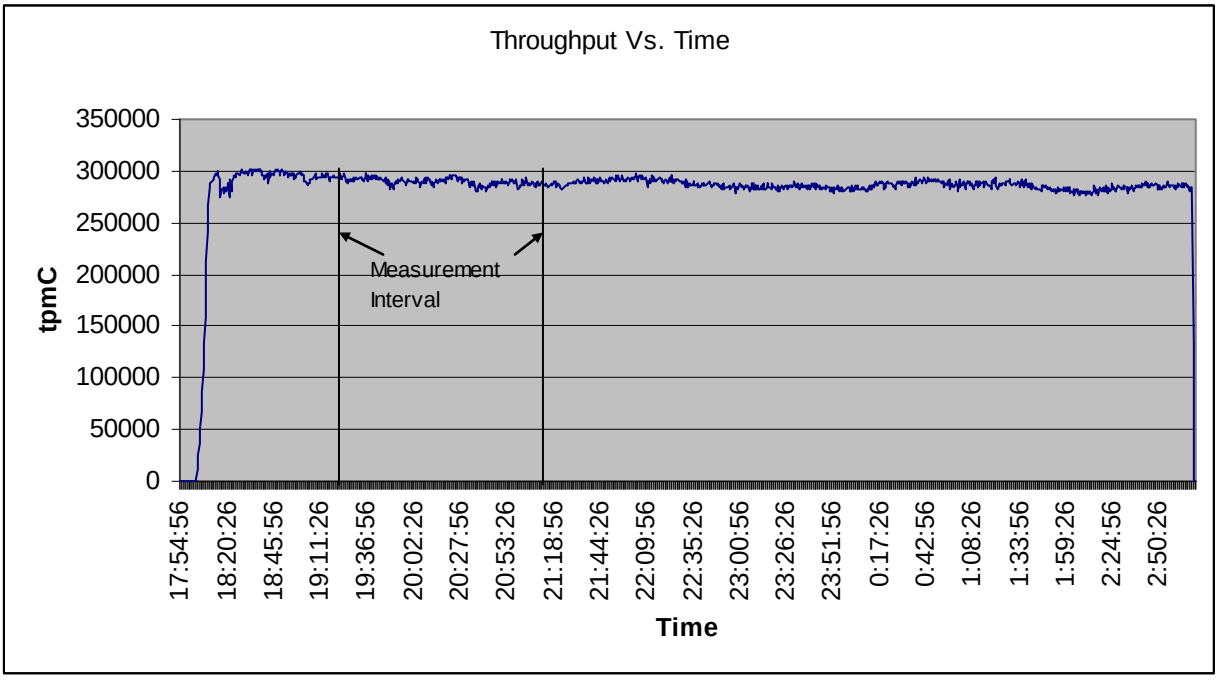


Figure 5.8: Throughput versus Time



Steady State Determination

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

Steady state was determined using real time monitor utilities from both the operating system and the RTE. Steady state was further confirmed by performing a run for 8 hours. The measurement interval is a part of the 8 hour run. The throughput data collected during the run is graphed in Figure 5.8.

Work Performed During Steady State

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

For each of the TPC Benchmark C transaction types, the following steps are executed. Each emulated user starts an Internet browser and asks to attach to the application on the desired client. The application formats the menus, input forms and data output using HTML (HyperText Markup Language). The HTML strings are transmitted over TCP/IP back to the client, where they can be displayed by any Web Browser software. The application on the client is run under the control of the Microsoft IIS.

Transactions are submitted by the RTE in accordance with the rules of the TPC-C benchmark. The emulated user chooses a transaction from the menu. The RTE records the time it takes from selecting the menu item to receiving the requested form. Data is generated for input to the form, then the user waits the specified keying time. The submit is sent and the RTE records the time it takes for the transaction to be processed and all the output data to be returned. The user then waits for the randomly generated think time before starting the process over again. All timings taken by the RTE generate a start and end timestamp. Keying and think times are calculated as the difference between end-time of a timing to the start of the next.

The database records transactions in the database tables and the transaction log. Writes to the database may stay in Oracle's in-memory data cache for a while before being written to disk. Checkpoints are initiated once the log files were filled and allowed to roll over.

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.

Regulation of Transaction Mix

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

The RTE was given a weighted random distribution, which could not be adjusted during the run.

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 5.3: 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	85.01%
	Remote warehouse	14.99%
	Accessed by last name	60.00%
Order Status	Accessed by last name	59.99%
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.996%
	Payment	43.002%
	Order status	4.001%
	Delivery	4.001%
	Stock level	4.001%

Checkpoint

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.

Oracle database was performed checkpoints at log switches. The database log files were sized such that a log switch would occur approximately every 26 minutes at the desired throughput. Two complete checkpoints occurred during the warm-up period. There were four complete checkpoint occurred during the measurement period, the first of which started 19 minutes and 23 seconds into the measurement interval..

Clause 6 Related Items

RTE Descriptions

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.

PRTE Software was used to simulate terminal users, generate random data and record response times. This package ran on systems that are distinct from the system under test. PRTE command file used is included in Appendix A.

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.

Due to the large number of PCs and associated hardware that would be required to run these tests, Remote Terminal Emulator was used to emulate the connected PCs and LAN. As configured for this test, the driver software emulates the traffic that would be observed from the users ' PCs connected by Ethernet to the front-end clients using HTTP (HyperText Transfer Protocol)over TCP/IP.

The driver system consisted of 10 ProLiant DL580 servers, 1 master RTE and 9 slaves.

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 1 shows the tested and priced benchmark configurations.

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 eight clients are connected to the server via 1000Base T Ethernet switch. The server has 2 connections into the switch, while each client has only 1 connection into the switch.

The drivers systems and client systems were connected using another 1000BaseT Ethernet switch.

Operator Intervention

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

This configuration does not require any operator intervention to sustain eight hours of the reported throughput.

Clause 7 Related Items

System 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 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 details of the hardware and software are reported in the front of this report as part of the executive summary. All third party quotations are included at the end of this report as Appendix D.

Availability, Throughput, and Price Performance

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

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

- **Maximum Qualified Throughput 290,040 tpmC**
- **Price per tpmC $0.39 per tpmC**
- **Watts per KtpmC 4.22 watts/KtpmC**
- **Available August 16, 2010**

All components are available now.

Country Specific Pricing

Additional Clause 7 related items may be included in the Full Disclosure Report for each country specific priced configuration. Country specific pricing is subject to Clause 7.1.7

This system is being priced for the United States of America.

Usage Pricing

For any usage pricing, the sponsor must disclose:

- *Usage level at which the component was priced.*
- *A statement of the company policy allowing such pricing.*

The component pricing based on usage is shown below:

- Oracle Database 11g R2 Standard Edition One
- Oracle Enterprise Linux
- Windows Server 2008 R2 Standard Edition

Clause 9 Related Items

Auditor's Report

The auditor's name, address, phone number, and a copy of the auditor's attestation letter indicating compliance must be included in the Full Disclosure Report.

This implementation of the TPC Benchmark C was audited by Lorna Livingtree of Performance Metrics Inc.

Performance Metrics, Inc. P.O. Box 984 Klamath, CA 95538
phone: 707-482-0523 fax: 707-482-0575 email: lornal@perfmetrics.com

August 13, 2010

Mr. Bryon Georgson
Database Performance Engineer
Hewlett-Packard Company
20555 SH 249
Houston, TX 77070

I have verified by remote the TPC Benchmark™ C for the following configuration:
Platform: HP ProLiant ML350 G6

Database Manager: Oracle Database 11g Release 2 Standard Edition

Operating System: Oracle Enterprise Linux

Transaction Monitor: Microsoft COM+

System Under Test:				
CPU's	Memory	Disks (total)	90% Response	TpmC
1 Intel Xeon 6 core @ 2.67 Ghz	Main: 96 GB	24 @ 120 GB SSD 24 @ 500 GB 1 @ 72 GB (OS)	1.11	**290,040**
Clients: 3 ML150 G6				
1 Intel quad core	2 GB	1 @ 250 GB	NA	**NA**

@ 2.13 Ghz				

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark. The following attributes of the benchmark were given special attention:

- The transactions were correctly implemented.
- The database files were properly sized.
- The database was properly scaled with 24,000 warehouses, all of which were active during the measured interval.
- The ACID properties were successfully demonstrated
- Input data was generated according to the specified percentages.
- Eight hours of mirrored log space was present on the tested system.
- Eight hours of growth space for the dynamic tables was present on the tested system.
- The data for the 60 days space calculation was verified.
- The steady state portion of the test was 120 minutes.
- There was one complete checkpoint in steady state before the measured interval.
- There were 4 checkpoints started and completed inside the measured interval.
- The system pricing was checked for major components and maintenance.
- Third party quotes were verified for compliance.

Auditor Notes: None.

Sincerely,

Lorna Livingtree

Appendix A: Source Code

```
-------------------
---- buf.cpp
-------------------

/*
**
** File:
**
**  buf.c double buffering code to emulate c runtime file reading
**
** Author:
**
**  Bill Carr
**
** Revisions:
**
**   10/04/95 WCarr
**  - Original
**
**
*/

/*#ifdef HISTORY

  02-Jun-97 WCarr Removed use of Mutex objects in favor of critical
       sections.  These prove to be at least one order of
       magnitude faster.

  31-Oct-97 WCarr Fixed a buffer wrap problem where if the data
were
       to end exactly on the buffer end, the code would
       not wrap to the beginning.

       Also added code that causes a writer to block if
       there is no room to write in data.  Fixed the
       timeout code so that a non-blocking read or write
       can function.

  06-Nov-97 WCarr Modified APIs to allow the read and write
operaions
       to supply the timeout values.

       Fixed a bug where the critical section was not
       released when a buffer read or write operation
       timed out.
*/
#include "stdafx.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

#include <crtdbg.h>
#include <windows.h>

#include "buf.h"

int
bufopen(size_t bufsize, BUFPTR *bufptr)
{
  BUFPTR buf;

  *bufptr = NULL;
  if( NULL == (buf = (BUFPTR) malloc( (sizeof( BUF ) - BUF_MINSIZE)
+ bufsize )))
    return (BUF_MALLOCFAIL);
  buf->freestart = (uchar *) buf->buf;
  buf->storedstart = (uchar *) buf->buf;
  buf->size = bufsize;
  buf->maxplus1 = (uchar *) buf->buf + bufsize;
  buf->full = FALSE;
  buf->blockedreadercount = 0;
  buf->blockedwritercount = 0;
  InitializeCriticalSection( &buf->control );
  if((HANDLE)NULL == (buf->dataready = CreateEvent(NULL, FALSE,
FALSE, NULL))) {
    free(buf);
    return (BUF_CREEVENT);
  }
  if((HANDLE)NULL == (buf->spacefreed = CreateEvent(NULL, FALSE,
FALSE, NULL))){
    free(buf);
    return (BUF_CREEVENT);
  }

  *bufptr = buf;
  return(BUF_SUCCESS);
}

static void
calcstoredsize(BUFPTR buf, size_t *storedsize, size_t
*storedsizehigh)
{
  if( buf->storedstart < buf->freestart ) {
    *storedsizehigh = *storedsize = buf->freestart - buf-
>storedstart;
  }
  else if( buf->full || buf->storedstart > buf->freestart ) {
    *storedsizehigh = ((uchar *) buf->buf + buf->size) - buf-
>storedstart;
    *storedsize = *storedsizehigh + (buf->freestart - (uchar *)buf-
>buf);
  }
  else {
    *storedsizehigh = *storedsize = 0;
  }
  return;
}

/* bufread.
 * Current implementation mixes two paradigms.  API implies
 * partial read of buffer is possible however implementation
 * insists on a complete read.
 * Possible fixes include:
 *    Add a bufread_complete routine which only returns with
 *   full data.
 *    Return partial data from read and have the caller verify
 *   that they got the data they requested.
 *
 */
int
bufread(void *rbuf, size_t btr, size_t *br, uint timeout, BUFPTR
buf)
{
  size_t storedsize, storedsizehigh;
  DWORD status, last_error;

  if( btr <= 0 )
    return BUF_SUCCESS;

  if( btr > buf->size )
    return BUF_READWAYTOOBIG;

  EnterCriticalSection( &buf->control );
  while( 1 ) {

    /* see if we have enough data to get from the buffer */
    calcstoredsize( buf, &storedsize, &storedsizehigh );
    if( btr <= storedsize )
      break;

    /* not enough data. if no wait, return else block until data
available. */
    if( 0 == timeout ) {
      LeaveCriticalSection( &buf->control );
      return BUF_READTIMEOUT;
    }
    buf->blockedreadercount++;
    LeaveCriticalSection( &buf->control );
    status = WaitForSingleObjectEx( buf->dataready, timeout, TRUE
);
    EnterCriticalSection( &buf->control );
    buf->blockedreadercount--;
    if( WAIT_OBJECT_0 == status )
      continue;
    LeaveCriticalSection( &buf->control );
    if( WAIT_TIMEOUT == status )
      return BUF_READTIMEOUT;
    else if( WAIT_IO_COMPLETION == status )
      return BUF_IOCOMPLETE;
    else {
      last_error = GetLastError( );
      return BUF_READWAITFAILED;
    }
  }

  if( btr <= storedsizehigh ) {
    CopyMemory( rbuf, buf->storedstart, btr );
    buf->storedstart += btr;
    if( buf->storedstart == buf->maxplus1 )
      buf->storedstart = (uchar *) buf->buf;
    else if( buf->storedstart > buf->maxplus1 )
      /* error */
      _ASSERT( FALSE );
  }
  else {
    CopyMemory( rbuf, buf->storedstart, storedsizehigh );
    CopyMemory( (uchar *)rbuf + storedsizehigh, buf->buf, btr -
storedsizehigh);
    buf->storedstart = (uchar *)buf->buf + (btr - storedsizehigh);
  }
  storedsize -= btr;

  if( buf->freestart == buf->storedstart ) {
#ifdef NDEBUG  /* keep messages in the buffer as long a possible
for debugging*/
    buf->freestart = buf->storedstart = (uchar *)buf->buf;
#endif
  }
  buf->full = FALSE;
```

```
  /* if data is in the buffer and a reader is blocked, unblock it
*/
  if(( 0 < storedsize ) && ( 0 != buf->blockedreadercount )) {
    SetEvent( buf->dataready );
  }

  /* see if a writer is blocked and unblock one */
  if( 0 != buf->blockedwritercount ) {
    SetEvent( buf->spacefreed );
  }

  LeaveCriticalSection( &buf->control );

  *br = btr;

  return BUF_SUCCESS;
}

static void calcfreesize(BUFPTR buf, size_t *freesize, size_t
*freesizehigh)
{
  if( buf->storedstart > buf->freestart ) {
    *freesizehigh = *freesize = buf->storedstart - buf->freestart;
  }
  else if( !buf->full && buf->storedstart <= buf->freestart ) {
    *freesizehigh = ((uchar *)buf->buf + buf->size) - buf-
>freestart;
    *freesize = *freesizehigh + (buf->storedstart - (uchar *)buf-
>buf);
  }
  else {
    *freesizehigh = *freesize = 0;
  }
  return;
}

int
bufwrite(const void *wbuf, size_t btw, size_t *bw, uint timeout,
BUFPTR buf)
{
  size_t freesize, freesizehigh;
  DWORD status, last_error;

  if( btw <= 0 )
    return BUF_SUCCESS;

  if( btw > buf->size )
    return BUF_WRITEWAYTOOBIG;

  EnterCriticalSection( &buf->control );
  while( 1 ) {

    /* see if we have enough room to put all data in the buffer */
    calcfreesize( buf, &freesize, &freesizehigh );
    if( !buf->full && btw <= freesize )
      break;

    /* not enough room. if no wait, return else block until space
available. */
    if( 0 == timeout ) {
      LeaveCriticalSection( &buf->control );
      return BUF_WRITETIMEOUT;
    }
    buf->blockedwritercount++;
    LeaveCriticalSection( &buf->control );
    status = WaitForSingleObject( buf->spacefreed, timeout );
    EnterCriticalSection( &buf->control );
    buf->blockedwritercount--;
    if( WAIT_OBJECT_0 == status )
      continue;
    LeaveCriticalSection( &buf->control );
    if( WAIT_TIMEOUT == status )
      return BUF_WRITETIMEOUT;
    else {
      last_error = GetLastError( );
      return BUF_WRITEWAITFAILED;
    }
  }

  if( btw <= freesizehigh ) {
    CopyMemory( buf->freestart, wbuf, btw );
    buf->freestart += btw;
    if( buf->freestart == buf->maxplus1 )
      buf->freestart = (uchar *)buf->buf;
    else if( buf->freestart > buf->maxplus1 )
      /* error */
      _ASSERT( FALSE );
  }
  else {
    CopyMemory( buf->freestart, wbuf, freesizehigh );
    CopyMemory( buf->buf, (uchar *)wbuf + freesizehigh, btw -
freesizehigh );
    buf->freestart = (uchar *)buf->buf + ( btw - freesizehigh );
  }
  freesize -= btw;

  if( buf->freestart == buf->storedstart )
    buf->full = TRUE;

  /* see if a reader is blocked and unblock one */
  if( 0 != buf->blockedreadercount ) {
    SetEvent( buf->dataready );
  }

  /* if space is available and a writer is blocked, unblock it */
  if(( 0 < freesize ) && ( 0 != buf->blockedwritercount )) {
    SetEvent( buf->spacefreed );
  }

  LeaveCriticalSection( &buf->control );

  *bw = btw;

  return BUF_SUCCESS;
}

void __cdecl bufclose(BUFPTR buf)
{
  DeleteCriticalSection( &buf->control );
  CloseHandle( buf->dataready );
  CloseHandle( buf->spacefreed );
  free( buf );
}
-------------------
---- buf.h
-------------------

/*
**
** File:
**
**  buf.h double buffering code to emulate c runtime file reading
**
** Author:
**
**  Bill Carr
**
** Revisions:
**
**   10/04/95 WCarr
**  - Original
**
**
*/

/*#ifdef HISTORY

  02-Jun-97 WCarr Removed use of Mutex objects in favor of critical
      sections.  These prove to be at least one order of
      magnitude faster.

*/

#ifndef _buf_h_
#define _buf_h_

#ifndef WIN32_LEAN_AND_MEAN
# define WIN32_LEAN_AND_MEAN
#endif
#include <windows.h>

#define BUF_INFINITE INFINITE

#define BUF_SUCCESS 0
#define BUF_READFAIL 1    /* Read thread exited unexpectedly */
#define BUF_CREEVENT 2    /* internal error Failure to create event
*/
#define BUF_READTIMEOUT 3 /* Reading thread timed out */
#define BUF_WRITETIMEOUT 4  /* Writing thread timed out */
#define BUF_MALLOCFAIL 5  /* failed to allocate needed worksapce */
#define BUF_READWAYTOOBIG 6 /* request larger than whole buffer */
#define BUF_WRITEWAYTOOBIG 7  /* request larger than whole buffer
*/
#define BUF_WRITETOOBIG 8 /* request larger than available space */
#define BUF_READWAITFAILED 9  /* internal error while waiting for
data */
#define BUF_WRITEWAITFAILED 10  /* internal error while waiting to
store */
#define BUF_IOCOMPLETE 11 /* an external async I/O operation
completed */

#define BUF_MINSIZE 4

typedef unsigned int uint;
typedef unsigned char uchar;

struct _buf
{
  uchar *freestart;
  uchar *storedstart;
  size_t size;
  uchar *maxplus1;
  BOOL full;
  int blockedreadercount;
  int blockedwritercount;
  CRITICAL_SECTION control;
  HANDLE dataready;
  HANDLE spacefreed;
```

```
  char  buf[BUF_MINSIZE]; /* MUST BE AT END for malloc to succeed
*/
};
typedef struct _buf BUF, *BUFPTR;

int bufopen(size_t bufsize, BUFPTR *buf);
int bufread(void *rbuf, size_t btr, size_t *br, uint timeout,
BUFPTR buf);
int bufwrite(const void *wbuf, size_t btw, size_t *bw, uint
timeout,BUFPTR buf);
void __cdecl bufclose(BUFPTR);

#endif
-------------------
---- DBConnection.cpp
-------------------

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

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

//#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)*DBConnections);
  for (i=0; i<DBConnections; i++)
    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]);

    }

    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;
    output->delta_time = GetTickCount();

#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();
  output->delta_time=GetTickCount() - output->delta_time;
  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
#ifdef USE_DELIVERY_LOG
  write_delivery_log(output, id);
#endif

  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) {
        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

            if (DBExecution_pool[current].current_status == IDLE) {
              DBExecution_pool[current].current_status = IN_USE;
#ifndef CONNECTION_MUTEX
              ReleaseMutex(DBExecution_lock[current]);
#else
              mutexptr=&(DBExecution_lock[current]);
#endif
              TlsSetValue(TlsPtr, (void *)
DBExecution_pool[current].pointer);
              return &(DBExecution_pool[current]);
            }
            else {
              ReleaseMutex(DBExecution_lock[current]);
#ifdef DEBUG
              userlog("get connection mutex, but current_status is
not IDLE\n");
#endif
            }
            break;

          case WAIT_TIMEOUT:
            break;

          default:
            userlog("Error on WaitForSingleObject, DBExecution\n");
            return NULL;
        }

      }
```

```
      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*)FULLDATE,
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),(text*)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(FULLDATE),(
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[jj] = 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)
{
  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;

  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)
{
  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)
{
  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)
{
  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;
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;
        }
    #ifdef USE_DELIVERY_LOG
        initDelLog(DeliveryThreads);
    #endif
        modtpcc_ready=1;
      }
      else if (ul_reason_for_call == DLL_PROCESS_DETACH) {
    #ifdef USE_DELIVERY_LOG
        endDelLog(DeliveryThreads);
    #endif
        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_spinlock))
    ;

    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;
  }
  bufclose (deliveryoutput);
  return TRUE;
}



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

  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);
  }
  deliverybufsize=(DeliveryQueues+DeliveryThreads)*sizeof(pT_delive
ry_data);
  if (BUF_SUCCESS != bufopen(deliverybufsize, &deliveryoutput)){
    userlog("Error opening delivery output buffer pipe\n");
```

```
    return FALSE;
  }

  return TRUE;
}



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

  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);

  buf_status=bufwrite(&queue_info,sizeof(pDelQueue_info),&bw,INFINI
TE,deliveryoutput);
        if (BUF_SUCCESS != buf_status)
          userlog("Error writing the delivery informatino to
delivery output buffer\n");

//        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_length
= 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_lengt
h = 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;
      int ii, buf_status;
      size_t br;
      pDelQueue_info CompletedDeliveries[DELIVERY_RESPONSE_COUNT];

      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);
      for (ii=0;ii<DELIVERY_RESPONSE_COUNT;ii++) {
     buf_status=bufread(&CompletedDeliveries[ii],sizeof(pDelQueue_info
   ),&br,0,deliveryoutput);
     if (BUF_READTIMEOUT == buf_status)
       CompletedDeliveries[ii]=NULL;
     else if (BUF_SUCCESS != buf_status)
       userlog ("Error reading delivery response buffer:
   %d\n",status);
      }
      if (!SetEvent(waitDelWork)) {
        userlog("Error on SetEvent(waitDelWork)\n");
        ret=sendform_deliveryoutput(status, queue_info->pdata,
   CompletedDeliveries);
        ret=FALSE;
      }
      else ret=sendform_deliveryoutput(status, queue_info->pdata,
   CompletedDeliveries);
      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)) {
       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)) {
       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)) {
              return NEWORDER_ITEMID_RANGE;
           }
           if ((pdata->o_orderline[items].ol_quantity >= 100) ||
               (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)) {
       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)) {
          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)) {
       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)) {
       return DELIVERY_CARRIER_ID_RANGE;
   }

   if ((ptr = value_ptr[DE_INPUT_QTIME].value) == NULL) {
     GetLocalTime(&(pdata->enqueue_date_time));
       pdata->enqueue_time = GetTickCount();
     time (&pdata->enqueue_time2);
       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)) {
       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)) {
          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)) {
       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,
pDelQueue_info CompletedDeliveries[DELIVERY_RESPONSE_COUNT])
{
  EXTENSION_CONTROL_BLOCK *pECB;
    int w_id, ld_id;
    char *form;
    int index = -1, ret;
    form_template_pool *pool;

  int ii, index2, jj;
  pT_delivery_data pCompletedDelivery;
  T_delivery_data blankDelivery = { 0 };

    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);

  index2 = D_QUEUE1;
  for (jj=0;jj<DELIVERY_RESPONSE_COUNT;jj++) {
    if (NULL==CompletedDeliveries[jj])
      pCompletedDelivery = &blankDelivery;
    else
      pCompletedDelivery =      CompletedDeliveries[jj]->pdata;
    fill_number (form, pCompletedDelivery-
>enqueue_time2,form_index[SUBI][index2].index,
      form_index[SUBI][index2].length);
    index2++;
    fill_number (form, pCompletedDelivery-
>delta_time,form_index[SUBI][index2].index,
      form_index[SUBI][index2].length);
    index2++;
    fill_number (form, pCompletedDelivery-
>w_id,form_index[SUBI][index2].index,
      form_index[SUBI][index2].length);
    index2++;
    fill_number (form, pCompletedDelivery-
>o_carrier_id,form_index[SUBI][index2].index,
      form_index[SUBI][index2].length);
    index2++;
    for (ii=0;ii<10;ii++) {
      fill_number (form, pCompletedDelivery-
>o_id[ii],form_index[SUBI][index2].index,
        form_index[SUBI][index2].length);
      index2++;
    }
    if (NULL != CompletedDeliveries[jj]){
      addFreeDelQueue(CompletedDeliveries[jj]);
    }
  }

    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
#define D_QUEUE1 DE_CARID+1
#define D_DELTA1 D_QUEUE+1
#define D_wID1 D_DELTA1+1
#define D_CAR1 D_WID1 + 1
#define D_OID10 D_CAR1 + 1
#define D_OID11 D_OID10 + 1
#define D_OID12 D_OID11 + 1
#define D_OID13 D_OID12 + 1
#define D_OID14 D_OID13 + 1
#define D_OID15 D_OID14 + 1
#define D_OID16 D_OID15 + 1
#define D_OID17 D_OID16 + 1
#define D_OID18 D_OID17 + 1
#define D_OID19 D_OID18 + 1
#define D_QUEUE2 D_OID19 + 1
#define D_DELTA2 D_QUEUE2 + 1
#define D_WID2 D_DELTA2 + 1
#define D_CAR2 D_WID2 + 1
#define D_OID20 D_CAR2 + 1
#define D_OID21 D_OID20 + 1
#define D_OID22 D_OID21 + 1
#define D_OID23 D_OID22 + 1
#define D_OID24 D_OID23 + 1
#define D_OID25 D_OID24 + 1
#define D_OID26 D_OID25 + 1
#define D_OID27 D_OID26 + 1
```

```
#define D_OID28 D_OID27 + 1
#define D_OID29 D_OID28 + 1
#define DE_FORMINDEX_SIZE    D_OID29+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>"
"Previous Deliveries:<BR>"
"########## ######## ##### ## #### #### #### #### #### #### ####
#### #### ####"
"<BR>"
"########## ######## ##### ## #### #### #### #### #### #### ####
#### #### ####"
"<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>"
"</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>
#include "buf.cpp"

#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"
#define DELIVERY_RESPONSE_COUNT 2

typedef struct _DelQueue_info {
  _DelQueue_info *Next;
  T_delivery_data *pdata;
  HANDLE queue_lock;
} DelQueue_info;
typedef struct _DelQueue_info DelQueue_info, *pDelQueue_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,
pDelQueue_info CompletedDeliveries[DELIVERY_RESPONSE_COUNT]);
```

```
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;
BUFPTR deliveryoutput;

static form_index_entry
form_index[POOL_TYPE_TXN_MAX][TXN_TYPE_MAX][MAX_FORM_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
//  modtpcc.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__FBB80AB0_1068_4095_8E53_EEA38B5CF47B__INCLUD
ED_)
#define
AFX_STDAFX_H__FBB80AB0_1068_4095_8E53_EEA38B5CF47B__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>
#include <io.h>
#include <time.h>
#include <process.h>
#include <sys/stat.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__FBB80AB0_1068_4095_8E53_EEA38B5CF47B__INCLUD
ED_)
-------------------
---- 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-mon-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,c
bf_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_DEFAU
LT));


#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;
   time_t      enqueue_time2;
   DWORD           complete_time;
   DWORD       delta_time;
   int             o_carrier_id;
   int             o_id[10];
};
typedef struct T_delivery_data T_delivery_data, *pT_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;
};

struct stooutstruct {
   int terror;
   int low_stock;
   int retry;
};

struct stostruct {
   struct stoinstruct stoin;
   struct stooutstruct stoout;
};
```

Appendix B: Database Design

```
---------------
--- alterordr.sql
--------------

alter database datafile '/home/oracle/tpcc_disks/ordr_0_0' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_7' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_3' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_4' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_5' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_2' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_6' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_8' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_1' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_11' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_10' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_12' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_13' resize
51503153152;
alter database datafile '/home/oracle/tpcc_disks/ordr_0_9' resize
51503153152;
---------------
--- createdb.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreatedb.sh Mon Mar 29 19:09:14
CDT 2010 */
spool createdb.log

set echo on

shutdown abort

startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxinstances 1
  datafile
   '/home/oracle/tpcc_disks//system_1' size 400M reuse
  logfile '/home/oracle/tpcc_disks//log_1_1' size 43945M reuse,
          '/home/oracle/tpcc_disks//log_1_2' size 43945M reuse
  sysaux datafile '/home/oracle/tpcc_disks//tpccaux' size 120M
reuse ;



create undo tablespace undo_1 datafile
  '/home/oracle/tpcc_disks//roll1' size 8096M reuse blocksize 8K;

set echo off
exit sql.sqlcode
---------------
--- createindex_icust1.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:23 CDT 2010 */
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 4
  compute statistics
  tablespace icust1_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
---------------
--- createindex_icust2.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:24 CDT 2010 */
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 16
  compute statistics
  tablespace icust2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
---------------
--- createindex_idist.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:24 CDT 2010 */
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/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:25 CDT 2010 */
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/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:27 CDT 2010 */
set timing on
  exit 0;
---------------
--- createindex_iordl.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:26 CDT 2010 */
set timing on
  exit 0;
---------------
--- createindex_iordr1.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:25 CDT 2010 */
set timing on
  exit 0;
---------------
--- createindex_iordr2.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:26 CDT 2010 */
set timing on
    set sqlblanklines on
```

```
    spool createindex_iordr2.log ;
    set echo on ;
    drop index iordr2 ;
      create unique index iordr2 on ordr ( o_c_id
, o_d_id
, o_w_id
, o_id )
  pctfree 25  initrans 4
  storage ( buffer_pool default )
  parallel 16
  compute statistics
  tablespace iordr2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
---------------
--- createindex_istok.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:25 CDT 2010 */
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 16
  compute statistics
  tablespace istok_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
---------------
--- createindex_iware.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreateindex.sh Mon Mar 29
19:09:23 CDT 2010 */
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 292000 24000
@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/tpcc24000/scripts/buildcreatetable.sh Mon Mar 29
19:09:15 CDT 2010 */
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 720000000
  hash is ( (c_id * ( 24000 * 10 ) + c_w_id * 10 + c_d_id) )
  size 180
  pctfree 0  initrans 3
  storage ( buffer_pool recycle ) parallel ( degree 4 )
  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/tpcc24000/scripts/buildcreatetable.sh Mon Mar 29
19:09:16 CDT 2010 */
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 240000
  hash is ( ((d_w_id * 10) + d_id) )
  size 1448
    initrans 4
  storage ( buffer_pool default )
  tablespace dist_0;

create table dist (
  d_id number
, d_w_id number
, d_ytd number
, d_next_o_id number
, d_tax number
, d_name varchar2(10)
, d_street_1 varchar2(20)
, d_street_2 varchar2(20)
, d_city varchar2(20)
, d_state char(2)
, d_zip char(9)
)
cluster distcluster (
  d_id
, d_w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
---------------
--- createtable_hist.sql
--------------

/* created automatically by
/home/oracle/tpcc24000/scripts/buildcreatetable.sh Mon Mar 29
19:09:17 CDT 2010 */
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/tpcc24000/scripts/buildcreatetable.sh Mon Mar 29
19:09:19 CDT 2010 */
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/tpcc24000/scripts/buildcreatetable.sh Mon Mar 29
19:09:21 CDT 2010 */
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 240000
    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/tpcc24000/scripts/buildcreatetable.sh Mon Mar 29
19:09:21 CDT 2010 */
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/tpcc24000/scripts/buildcreatetable.sh Mon Mar 29
19:09:20 CDT 2010 */
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 240000
    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/tpcc24000/scripts/buildcreatetable.sh Mon Mar 29
19:09:18 CDT 2010 */
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 2400000000
  hash is ( (s_i_id * 24000 + s_w_id) )
  size 256
  pctfree 0  initrans 2 maxtrans 2
  storage ( buffer_pool keep ) parallel ( degree 4 )
  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/tpcc24000/scripts/buildcreatetable.sh Mon Mar 29
19:09:14 CDT 2010 */
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 24000
  hash is ( (w_id - 1) )
  size 1448
    initrans 2
  storage ( buffer_pool default )
  tablespace ware_0;

create table ware (
  w_id number
, w_ytd number
, w_tax number
, w_name varchar2(10)
, w_street_1 varchar2(20)
, w_street_2 varchar2(20)
, w_city varchar2(20)
, w_state char(2)
, w_zip char(9)
)
cluster warecluster (
  w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
---------------
--- createts.sh
--------------

#created automatically by
/home/oracle/tpcc24000/scripts/buildcreatets.sh Mon Mar 29 19:09:05
CDT 2010

# Tablespace ware, ts size 60M (61440K)
# each file 60M (61440K)
# extents 52384K (52384K)
# 1 files

$tpcc_createts ware 1 1       60M 52384K unix 0       0 4 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for ware failed.  Exiting.
      exit 0
    fi

# Tablespace cust, ts size 644000M (659456000K)
# each file 8050M (8243200K)
# extents 132812K (132812K)
# 80 files

$tpcc_createts cust 80 1       8050M 132812K unix 0       1 4 auto
t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for cust failed.  Exiting.
      exit 0
    fi

# Tablespace dist, ts size 510M (522240K)
# each file 510M (522240K)
# extents 514624K (514624K)
# 1 files

$tpcc_createts dist 1 1       510M 514624K unix 0       81 4 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for dist failed.  Exiting.
      exit 0
    fi

# Tablespace hist, ts size 64320M (65863680K)
# each file 8040M (8232960K)
# extents 102878K (102878K)
# 8 files

$tpcc_createts hist 8 1       8040M 102878K unix 0       82 4 auto
t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for hist failed.  Exiting.
      exit 0
    fi

# Tablespace stok, ts size 721800M (739123200K)
# each file 8020M (8212480K)
# extents 149250K (149250K)
# 90 files

$tpcc_createts stok 90 1       8020M 149250K unix 0       90 4 auto
t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for stok failed.  Exiting.
      exit 0
    fi

# Tablespace item, ts size 20M (20480K)
# each file 20M (20480K)
# extents 16892K (16892K)
# 1 files

$tpcc_createts item 1 1       20M 16892K unix 0       180 4 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for item failed.  Exiting.
      exit 0
    fi

# Tablespace ordr, ts size 887460M (908759040K)
# each file 63390M (64911360K)
# extents 103344K (103344K)
# 14 files

$tpcc_createts ordr 14 1       63390M 103344K unix 0       181 4
16K t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for ordr failed.  Exiting.
      exit 0
    fi

# Tablespace nord, ts size 8400M (8601600K)
# each file 4200M (4300800K)
# extents 429024K (429024K)
# 2 files

$tpcc_createts nord 2 1       4200M 429024K unix 0       195 4 auto
t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for nord failed.  Exiting.
      exit 0
    fi

# Tablespace iware, ts size 40M (40960K)
# each file 40M (40960K)
# extents 31024K (31024K)
# 1 files

$tpcc_createts iware 1 1       40M 31024K unix 0       197 4 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for iware failed.  Exiting.
      exit 0
    fi

# Tablespace icust1, ts size 16910M (17315840K)
# each file 16910M (17315840K)
# extents 541024K (541024K)
# 1 files

$tpcc_createts icust1 1 1       16910M 541024K unix 0       198 4
16K t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for icust1 failed.  Exiting.
      exit 0
    fi

# Tablespace icust2, ts size 42420M (43438080K)
# each file 7070M (7239680K)
# extents 226024K (226024K)
# 6 files

$tpcc_createts icust2 6 1       7070M 226024K unix 0       199 4
auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for icust2 failed.  Exiting.
      exit 0
    fi

# Tablespace idist, ts size 130M (133120K)
# each file 130M (133120K)
# extents 121024K (121024K)
# 1 files

$tpcc_createts idist 1 1       130M 121024K unix 0       205 4 auto
t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for idist failed.  Exiting.
      exit 0
    fi

# Tablespace istok, ts size 50040M (51240960K)
# each file 50040M (51240960K)
# extents 1601024K (1601024K)
```

```
# 1 files

$tpcc_createts istok 1 1       50040M 1601024K unix 0       206 4
16K t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for istok failed.  Exiting.
      exit 0
    fi

# Tablespace iitem, ts size 20M (20480K)
# each file 20M (20480K)
# extents 11264K (11264K)
# 1 files

$tpcc_createts iitem 1 1       20M 11264K unix 0       207 4 auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for iitem failed.  Exiting.
      exit 0
    fi

# Tablespace iordr2, ts size 38050M (38963200K)
# each file 7610M (7792640K)
# extents 102452K (102452K)
# 5 files

$tpcc_createts iordr2 5 1       7610M 102452K unix 0       208 4
auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for iordr2 failed.  Exiting.
      exit 0
    fi

# Tablespace temp, ts size 125760M (128778240K)
# each file 7860M (8048640K)
# extents 201024K (201024K)
# 16 files

$tpcc_createts temp 16 1       7860M 201024K unix 1       213 4
auto t
    if expr $? != 0 > /dev/null; then
      echo Creating tablespace for temp failed.  Exiting.
      exit 0
    fi
---------------
--- loadcust.sh
--------------

#created automatically by
/home/oracle/tpcc24000/scripts/evenload.sh Mon Mar 29 19:09:22 CDT
2010
rm -f loadcust*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 24000 -C  -l 1 -m 375 >> loadcust0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -C  -l 376 -m 750 >> loadcust1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -C  -l 751 -m 1125 >> loadcust2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -C  -l 1126 -m 1500 >> loadcust3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -C  -l 1501 -m 1875 >> loadcust4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -C  -l 1876 -m 2250 >> loadcust5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -C  -l 2251 -m 2625 >> loadcust6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -C  -l 2626 -m 3000 >> loadcust7.log 2>&1 &
allprocs="$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/tpcc24000/scripts/evenload.sh Mon Mar 29 19:09:22 CDT
2010
rm -f loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 24000 -h  -b 1 -e 3000 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -h  -b 3001 -e 6000 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -h  -b 6001 -e 9000 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -h  -b 9001 -e 12000 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -h  -b 12001 -e 15000 >> loadhist4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -h  -b 15001 -e 18000 >> loadhist5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -h  -b 18001 -e 21000 >> loadhist6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -h  -b 21001 -e 24000 >> loadhist7.log 2>&1 &
allprocs="$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/tpcc24000/scripts/evenload.sh Mon Mar 29 19:09:22 CDT
2010
rm -f loadnord*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 24000 -n  -b 1 -e 24000 >> loadnord0.log 2>&1 &
allprocs="$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/tpcc24000/scripts/evenload.sh Mon Mar 29 19:09:22 CDT
2010
rm -f loadordrordl*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 24000 -o ${tpcc_disks_location}dummy0.dat -b 1 -e
3000 >> loadordrordl0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -o ${tpcc_disks_location}dummy1.dat -b 3001 -e
6000 >> loadordrordl1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -o ${tpcc_disks_location}dummy2.dat -b 6001 -e
9000 >> loadordrordl2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -o ${tpcc_disks_location}dummy3.dat -b 9001 -e
12000 >> loadordrordl3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -o ${tpcc_disks_location}dummy4.dat -b 12001 -e
15000 >> loadordrordl4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -o ${tpcc_disks_location}dummy5.dat -b 15001 -e
18000 >> loadordrordl5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -o ${tpcc_disks_location}dummy6.dat -b 18001 -e
21000 >> loadordrordl6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -o ${tpcc_disks_location}dummy7.dat -b 21001 -e
24000 >> loadordrordl7.log 2>&1 &
allprocs="$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/tpcc24000/scripts/evenload.sh Mon Mar 29 19:09:22 CDT
2010
rm -f loadstok*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 24000 -S  -j 1 -k 12500 >> loadstok0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -S  -j 12501 -k 25000 >> loadstok1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -S  -j 25001 -k 37500 >> loadstok2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -S  -j 37501 -k 50000 >> loadstok3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -S  -j 50001 -k 62500 >> loadstok4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -S  -j 62501 -k 75000 >> loadstok5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -S  -j 75001 -k 87500 >> loadstok6.log 2>&1 &
```

```
allprocs="$allprocs ${!}"
$tpcc_load -M 24000 -S  -j 87501 -k 100000 >> loadstok7.log 2>&1 &
allprocs="$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
---------------
--- space_get.sql
--------------

REM=============================================================
==+
REM         Copyright (c) 1995  Oracle Corp, Redwood Shores, CA
|
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  Oracle Corp, Redwood Shores, CA
|
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
--------------

-- 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;
 nulldate       DATE;
 TYPE rowidarray IS TABLE OF ROWID INDEX BY PLS_INTEGER;
 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('01-01-1811', 'MM-DD-YYYY');
    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
exit
---------------
--- defaultopts.sh
--------------

# configurable vars
tpcc_os=unix
tpcc_version=ttt
tpcc_ldrive=1
tpcc_scale=10
tpcc_np=1
tpcc_cpu=1
#megabytes
tpcc_memsize=512
#minutes
tpcc_runlen=24

tpcc_temp_imp=temp
tpcc_temp_size=10M
tpcc_temp_ext=calc
tpcc_temp_nf=calc
tpcc_temp_bs=2K

tpcc_ware_imp=iot
tpcc_ware_size=10M
tpcc_ware_ext=calc
tpcc_ware_nf=calc
tpcc_ware_bs=auto

tpcc_ware_used=-1
tpcc_ware_free=-1
tpcc_ware_trans=-1

tpcc_ware_indices=1-

tpcc_ware_autospace=t
tpcc_ware_flg=30
tpcc_ware_fl=22

tpcc_iware_imp=none
tpcc_iware_size=1M
tpcc_iware_ext=calc
tpcc_iware_nf=calc
tpcc_iware_bs=2K

tpcc_iware_used=-1
tpcc_iware_free=-1
tpcc_iware_trans=-1

tpcc_iware_autospace=t
tpcc_iware_flg=30
tpcc_iware_fl=22

tpcc_iware_indices=1-



tpcc_dist_imp=cluster
tpcc_dist_size=10M
```

```
tpcc_dist_ext=calc
tpcc_dist_nf=calc
tpcc_dist_bs=auto

tpcc_dist_used=-1
tpcc_dist_free=-1
tpcc_dist_trans=-1

tpcc_dist_indices=2-1-

tpcc_dist_autospace=t
tpcc_dist_flg=30
tpcc_dist_fl=22

tpcc_idist_imp=index
tpcc_idist_size=1M
tpcc_idist_ext=calc
tpcc_idist_nf=calc
tpcc_idist_bs=2K

tpcc_idist_used=-1
tpcc_idist_free=-1
tpcc_idist_trans=-1

tpcc_idist_autospace=t
tpcc_idist_flg=30
tpcc_idist_fl=22

tpcc_idist_indices=2-1-


tpcc_item_imp=cluster
tpcc_item_size=15M
tpcc_item_ext=calc
tpcc_item_nf=calc
tpcc_item_bs=auto

tpcc_item_used=-1
tpcc_item_free=-1
tpcc_item_trans=-1

tpcc_item_indices=1-

tpcc_item_autospace=t
tpcc_item_flg=30
tpcc_item_fl=22

tpcc_iitem_imp=index
tpcc_iitem_size=1M
tpcc_iitem_ext=calc
tpcc_iitem_nf=calc
tpcc_iitem_bs=2K

tpcc_iitem_used=-1
tpcc_iitem_free=-1
tpcc_iitem_trans=-1

tpcc_iitem_autospace=t
tpcc_iitem_flg=30
tpcc_iitem_fl=22

tpcc_iitem_indices=1-


tpcc_nord_imp=iot
tpcc_nord_size=10M
tpcc_nord_ext=calc
tpcc_nord_nf=calc
tpcc_nord_bs=auto

tpcc_nord_used=-1
tpcc_nord_free=-1
tpcc_nord_trans=-1

tpcc_nord_indices=1-2-3-

tpcc_nord_autospace=t
tpcc_nord_flg=30
tpcc_nord_fl=22

tpcc_inord_imp=none
tpcc_inord_size=1M
tpcc_inord_ext=calc
tpcc_inord_nf=calc
tpcc_inord_bs=2K

tpcc_inord_used=-1
tpcc_inord_free=-1
tpcc_inord_trans=-1

tpcc_inord_autospace=t
tpcc_inord_flg=30
tpcc_inord_fl=22

tpcc_inord_indices=1-2-3-


tpcc_ordl_imp=iot
tpcc_ordl_size=10M
tpcc_ordl_ext=calc
tpcc_ordl_nf=calc
tpcc_ordl_bs=auto

tpcc_ordl_used=-1
tpcc_ordl_free=-1
tpcc_ordl_trans=-1

tpcc_ordl_indices=1-2-3-4-

tpcc_ordl_autospace=t
tpcc_ordl_flg=30
tpcc_ordl_fl=22

tpcc_iordl_imp=none
tpcc_iordl_size=1M
tpcc_iordl_ext=calc
tpcc_iordl_nf=calc
tpcc_iordl_bs=2K

tpcc_iordl_used=-1
tpcc_iordl_free=-1
tpcc_iordl_trans=-1

tpcc_iordl_autospace=t
tpcc_iordl_flg=30
tpcc_iordl_fl=22

tpcc_iordl_indices=1-2-3-4-


tpcc_ordr_imp=table
tpcc_ordr_size=10M
tpcc_ordr_ext=calc
tpcc_ordr_nf=calc
tpcc_ordr_bs=auto

tpcc_ordr_used=-1
tpcc_ordr_free=-1
tpcc_ordr_trans=-1

tpcc_ordr_indices=2-3-1

tpcc_ordr_autospace=t
tpcc_ordr_flg=30
tpcc_ordr_fl=22

tpcc_iordr1_imp=index
tpcc_iordr1_size=1M
tpcc_iordr1_ext=calc
tpcc_iordr1_nf=calc
tpcc_iordr1_bs=2K

tpcc_iordr1_used=-1
tpcc_iordr1_free=-1
tpcc_iordr1_trans=-1

tpcc_iordr1_autospace=t
tpcc_iordr1_flg=30
tpcc_iordr1_fl=22

tpcc_iordr1_indices=2-3-1

tpcc_iordr2_imp=index
tpcc_iordr2_size=1M
tpcc_iordr2_ext=calc
tpcc_iordr2_nf=calc
tpcc_iordr2_bs=2K

tpcc_iordr2_used=-1
tpcc_iordr2_free=-1
tpcc_iordr2_trans=-1

tpcc_iordr2_autospace=t
tpcc_iordr2_flg=30
tpcc_iordr2_fl=22

tpcc_iordr2_indices=2-3-4-1


tpcc_stok_imp=cluster
tpcc_stok_size=35M
tpcc_stok_ext=calc
tpcc_stok_nf=calc
tpcc_stok_bs=auto

tpcc_stok_used=-1
tpcc_stok_free=-1
tpcc_stok_trans=-1

tpcc_stok_indices=1-2-

tpcc_stok_autospace=t
tpcc_stok_flg=30
tpcc_stok_fl=22

tpcc_istok_imp=index
tpcc_istok_size=1M
tpcc_istok_ext=calc
tpcc_istok_nf=calc
```

```
tpcc_istok_bs=2K

tpcc_istok_used=-1
tpcc_istok_free=-1
tpcc_istok_trans=-1

tpcc_istok_autospace=t
tpcc_istok_flg=30
tpcc_istok_fl=22

tpcc_istok_indices=1-2-


tpcc_cust_imp=cluster
tpcc_cust_size=25M
tpcc_cust_ext=calc
tpcc_cust_nf=calc
#bs
tpcc_cust_bs=2K

tpcc_cust_used=-1
tpcc_cust_free=-1
tpcc_cust_trans=-1

tpcc_cust_indices=1-2-3-

tpcc_cust_autospace=t
tpcc_cust_flg=30
tpcc_cust_fl=22


tpcc_icust1_imp=index
tpcc_icust1_size=1M
tpcc_icust1_ext=calc
tpcc_icust1_nf=calc
tpcc_icust1_bs=2K

tpcc_icust1_used=-1
tpcc_icust1_free=-1
tpcc_icust1_trans=-1

tpcc_icust1_autospace=t
tpcc_icust1_flg=30
tpcc_icust1_fl=22

tpcc_icust1_indices=1-2-3-

tpcc_icust2_imp=index
tpcc_icust2_size=1M
tpcc_icust2_ext=calc
tpcc_icust2_nf=calc
tpcc_icust2_bs=2K

tpcc_icust2_used=-1
tpcc_icust2_free=-1
tpcc_icust2_trans=-1

tpcc_icust2_autospace=t
tpcc_icust2_flg=30
tpcc_icust2_fl=22

tpcc_icust2_indices=6-7-1-2-3-

tpcc_hist_imp=table
tpcc_hist_size=10M
tpcc_hist_ext=calc
tpcc_hist_nf=calc
tpcc_hist_bs=auto

tpcc_hist_used=-1
tpcc_hist_free=-1
tpcc_hist_trans=-1

tpcc_hist_indices=no

tpcc_hist_autospace=t
tpcc_hist_flg=30
---------------
--- driver.sh
--------------

#!/bin/sh

. ./stepenv.sh

if expr $# \< 1 > /dev/null; then
  echo "$0 <starting stepname> <optional: only>"
  echo OR use:
  echo "$0 buildcreate  - to build the database creation scripts"
  echo "$0 create       - to create the database (after
buildcreate)"
  echo "$0 steps        - to list individual steps"
  exit 1
fi

if expr x$1 = xsteps > /dev/null; then
  echo stepnames are from creation scripts: $tpcc_create_steps
  echo
  echo or running steps: $tpcc_steps
  echo "use the 'only' option to only do that step (otherwise all
steps after will also be executed.)"
  echo "   (e.g.  $0 listfiles only)"
  echo "use the 'through' option to do a sequence of steps
(inclusively.)"
  echo "   (e.g.  $0 shutdowndb through startupdb-p_build)"
  exit 1
fi

startstep=$1
controlcmd=$2
endstep=$3

# Aliases for special steps
if test $startstep = buildcreate; then
  startstep=`echo $tpcc_create_steps | cut -d' ' -f1`
fi

if test $startstep = create; then
  startstep=`echo $tpcc_steps | cut -d' ' -f1`
fi

if test "x$controlcmd" = x; then
  endstep=
  # Since endstep is null it won't match any other steps, so we
keep going.
elif test "x$controlcmd" = xonly; then
  controlcmd=only
  # this is allowed
elif test "x$controlcmd" = xthrough; then
  actualstep=f
  for step in $tpcc_create_steps $tpcc_steps ; do
    if test "x$step" = "x$endstep"; then
      actualstep=t
    fi
  done
  if test $actualstep = f; then
    echo "Invalid step $endstep.  Use $0 steps  to show steps."
    exit 1
  fi
else
  echo "Invalid syntax.  Use $0 by itself for help."
  exit 1
fi

echo Starting from step: $startstep

dostep=f
for step in $tpcc_create_steps $tpcc_steps ; do
  if expr $step = $startstep > /dev/null; then
    dostep=t
  fi

  if expr $dostep = t > /dev/null; then
    echo STEP: $step
    cd $tpcc_bench
    $tpcc_scripts/`echo $step | cut -d- -f1`.sh `echo $step | sed -
e's/-*$/-/' | cut -d- -f2- | sed -e's/-/ /g'`
    lasterror=$?
    cd $tpcc_bench
    if test -n "`find $tpcc_bench/scripts -name '*.log'`"; then
  mv -f *.log `find $tpcc_bench/scripts -name '*.log'`
$tpcc_bench/log/
    else
      if test -n "`find $tpcc_bench/ -name '*.log'`"; then
        mv -f *.log $tpcc_bench/log/
      fi
    fi

    if expr $lasterror != 0 > /dev/null; then
      if expr $lasterror != 99 > /dev/null; then
        echo Step $step failed.  Stopping driver.
        exit 1
      else
        echo Step $step has completed and requested stop.  Stopping
driver.
        exit 0
      fi
    fi
    if test "x$controlcmd" = xonly; then
      exit 0
    fi
    if test "x$endstep" = "x$step"; then
      echo The driver reached the last desired step.  Stopping
driver.
      exit 0
    fi
  fi
done

if expr $dostep = f > /dev/null; then
  echo No such step: $1
fi



---------------
--- localoptions.sh
--------------
```

```
#LOCAL OPTION FILE- You must fill these in
# before the driver will work.


#oracle sid to use for the run
ORACLE_SID=tpcc


#folder location of the database files (or links to raw partitions)
tpcc_disks_location=/home/oracle/tpcc_disks/

#FOR NT
#tpcc_disks_location=\\\\.\\



#FOR RAC

#node id
#tpcc_rac_id=1


# How many createts_node*.sh will be run in this node, started from
tpcc_rac_id
# eq. if tpcc_rac_id is 3 and tpcc_rac_createts_count is 2
# createts_node3.sh and createts_node4.sh will be executed

#tpcc_rac_createts_count=1


#locations of various files used in the generation scripts.
#(you can usually leave these alone.)
tpcc_sql_dir=${tpcc_bench}/scripts/sql
tpcc_log_dir=${tpcc_bench}/log
tpcc_genscripts_dir=${tpcc_bench}/scripts/generated


#Once you have filled all the options, comment
#out or delete this line.
---------------
--- options.sh
--------------

tpcc_os='unix'
tpcc_version='ttt'
tpcc_ldrive='1'
tpcc_scale='24000'
tpcc_np='1'
tpcc_cpu='4'
tpcc_memsize='262144'
#tpcc_memsize='73728'
tpcc_runlen='8'
tpcc_compress='t'
tpcc_overflow='t'
tpcc_defbs='2'
tpcc_ieee_number='f'
tpcc_numfiles='0'

tpcc_cust_imp='cluster'
tpcc_cust_size='calc'
tpcc_cust_ext='calc'
tpcc_cust_nf='calc'
tpcc_cust_bs='auto'
tpcc_cust_used='-1'
tpcc_cust_free='0'
tpcc_cust_trans='3'
tpcc_cust_autospace='t'
tpcc_cust_flg='43'
tpcc_cust_fl='22'
tpcc_cust_rsize='auto'
tpcc_cust_hkey='auto'
tpcc_cust_hash='auto'
tpcc_cust_bpool='recycle'
tpcc_cust_indices=3-2-1-

tpcc_dist_imp='cluster'
tpcc_dist_size='calc'
tpcc_dist_ext='calc'
tpcc_dist_nf='calc'
tpcc_dist_bs='auto'
tpcc_dist_used='-1'
tpcc_dist_free='-1'
tpcc_dist_trans='4'
tpcc_dist_autospace='t'
tpcc_dist_flg='43'
tpcc_dist_fl='22'
tpcc_dist_rsize='auto'
tpcc_dist_hkey='auto'
tpcc_dist_hash='auto'
tpcc_dist_bpool='default'
tpcc_dist_indices=2-1-

tpcc_hist_imp='table'
tpcc_hist_size='1791'
tpcc_hist_ext='calc'
tpcc_hist_nf='calc'
tpcc_hist_bs='auto'
tpcc_hist_used='-1'
tpcc_hist_free='5'
tpcc_hist_trans='4'
tpcc_hist_autospace='t'
tpcc_hist_flg='43'
tpcc_hist_fl='22'
tpcc_hist_rsize='auto'
tpcc_hist_hkey='auto'
tpcc_hist_hash='auto'
tpcc_hist_bpool='recycle'
tpcc_hist_indices=no

tpcc_item_imp='cluster'
tpcc_item_size='calc'
tpcc_item_ext='calc'
tpcc_item_nf='calc'
tpcc_item_bs='auto'
tpcc_item_used='-1'
tpcc_item_free='0'
tpcc_item_trans='3'
tpcc_item_autospace='t'
tpcc_item_flg='43'
tpcc_item_fl='22'
tpcc_item_rsize='auto'
tpcc_item_hkey='auto'
tpcc_item_hash='auto'
tpcc_item_bpool='keep'
tpcc_item_indices=1-

tpcc_nord_imp='queue'
tpcc_nord_size='178'
tpcc_nord_ext='calc'
tpcc_nord_nf='calc'
tpcc_nord_bs='auto'
tpcc_nord_used='-1'
tpcc_nord_free='5'
tpcc_nord_trans='4'
tpcc_nord_autospace='t'
tpcc_nord_flg='43'
tpcc_nord_fl='22'
tpcc_nord_rsize='auto'
tpcc_nord_hkey='auto'
tpcc_nord_hash='auto'
tpcc_nord_bpool='default'
tpcc_nord_indices=1-2-3-

tpcc_ordl_imp='queue'
tpcc_ordl_size='21775'
tpcc_ordl_ext='calc'
tpcc_ordl_nf='calc'
tpcc_ordl_bs='16K'
tpcc_ordl_used='-1'
tpcc_ordl_free='5'
tpcc_ordl_trans='4'
tpcc_ordl_autospace='t'
tpcc_ordl_flg='43'
tpcc_ordl_fl='22'
tpcc_ordl_rsize='auto'
tpcc_ordl_hkey='auto'
tpcc_ordl_hash='auto'
tpcc_ordl_bpool='default'
tpcc_ordl_indices=1-2-3-4-

tpcc_ordr_imp='queue'
tpcc_ordr_size='1206'
tpcc_ordr_ext='calc'
tpcc_ordr_nf='calc'
tpcc_ordr_bs='16K'
tpcc_ordr_used='-1'
tpcc_ordr_free='5'
tpcc_ordr_trans='4'
tpcc_ordr_autospace='t'
tpcc_ordr_flg='43'
tpcc_ordr_fl='22'
tpcc_ordr_rsize='auto'
tpcc_ordr_hkey='auto'
tpcc_ordr_hash='auto'
tpcc_ordr_bpool='default'
tpcc_ordr_indices=2-3-1-

tpcc_stok_imp='cluster'
tpcc_stok_size='calc'
tpcc_stok_ext='calc'
tpcc_stok_nf='calc'
tpcc_stok_bs='auto'
tpcc_stok_used='-1'
tpcc_stok_free='0'
tpcc_stok_trans='2'
tpcc_stok_autospace='t'
tpcc_stok_flg='43'
tpcc_stok_fl='22'
tpcc_stok_rsize='auto'
tpcc_stok_hkey='auto'
tpcc_stok_hash='auto'
tpcc_stok_bpool='keep'
tpcc_stok_indices=1-2-

tpcc_ware_imp='cluster'
tpcc_ware_size='calc'
tpcc_ware_ext='calc'
tpcc_ware_nf='calc'
tpcc_ware_bs='auto'
tpcc_ware_used='-1'
```

```
tpcc_ware_free='-1'
tpcc_ware_trans='2'
tpcc_ware_autospace='t'
tpcc_ware_flg='43'
tpcc_ware_fl='22'
tpcc_ware_rsize='auto'
tpcc_ware_hkey='auto'
tpcc_ware_hash='auto'
tpcc_ware_bpool='default'
tpcc_ware_indices=1-

tpcc_icust1_imp='index'
tpcc_icust1_size='736'
tpcc_icust1_ext='calc'
tpcc_icust1_nf='calc'
tpcc_icust1_bs='16K'
tpcc_icust1_used='-1'
tpcc_icust1_free='1'
tpcc_icust1_trans='3'
tpcc_icust1_autospace='t'
tpcc_icust1_flg='43'
tpcc_icust1_fl='22'
tpcc_icust1_rsize='auto'
tpcc_icust1_hkey='auto'
tpcc_icust1_hash='auto'
tpcc_icust1_bpool='default'
tpcc_icust1_indices=3-2-1-

tpcc_icust2_imp='index'
tpcc_icust2_size='4591'
tpcc_icust2_ext='calc'
tpcc_icust2_nf='calc'
tpcc_icust2_bs='auto'
tpcc_icust2_used='-1'
tpcc_icust2_free='1'
tpcc_icust2_trans='3'
tpcc_icust2_autospace='t'
tpcc_icust2_flg='43'
tpcc_icust2_fl='22'
tpcc_icust2_rsize='auto'
tpcc_icust2_hkey='auto'
tpcc_icust2_hash='auto'
tpcc_icust2_bpool='default'
tpcc_icust2_indices=6-3-2-7-1-

tpcc_idist_imp='index'
tpcc_idist_size='4'
tpcc_idist_ext='calc'
tpcc_idist_nf='calc'
tpcc_idist_bs='auto'
tpcc_idist_used='-1'
tpcc_idist_free='5'
tpcc_idist_trans='3'
tpcc_idist_autospace='t'
tpcc_idist_flg='43'
tpcc_idist_fl='22'
tpcc_idist_rsize='auto'
tpcc_idist_hkey='auto'
tpcc_idist_hash='auto'
tpcc_idist_bpool='default'
tpcc_idist_indices=2-1-

tpcc_iitem_imp='index'
tpcc_iitem_size='2048'
tpcc_iitem_ext='calc'
tpcc_iitem_nf='calc'
tpcc_iitem_bs='auto'
tpcc_iitem_used='-1'
tpcc_iitem_free='5'
tpcc_iitem_trans='4'
tpcc_iitem_autospace='t'
tpcc_iitem_flg='43'
tpcc_iitem_fl='22'
tpcc_iitem_rsize='auto'
tpcc_iitem_hkey='auto'
tpcc_iitem_hash='auto'
tpcc_iitem_bpool='default'
tpcc_iitem_indices=1-

tpcc_inord_imp='none'
tpcc_inord_size='229'
tpcc_inord_ext='calc'
tpcc_inord_nf='calc'
tpcc_inord_bs='auto'
tpcc_inord_used='-1'
tpcc_inord_free='5'
tpcc_inord_trans='4'
tpcc_inord_autospace='t'
tpcc_inord_flg='43'
tpcc_inord_fl='22'
tpcc_inord_rsize='auto'
tpcc_inord_hkey='auto'
tpcc_inord_hash='auto'
tpcc_inord_bpool='default'
tpcc_inord_indices=1-2-3-

tpcc_iordl_imp='none'
tpcc_iordl_size='8072'
tpcc_iordl_ext='calc'
tpcc_iordl_nf='calc'
tpcc_iordl_bs='auto'
tpcc_iordl_used='-1'
tpcc_iordl_free='5'
tpcc_iordl_trans='4'
tpcc_iordl_autospace='t'
tpcc_iordl_flg='43'
tpcc_iordl_fl='22'
tpcc_iordl_rsize='auto'
tpcc_iordl_hkey='auto'
tpcc_iordl_hash='auto'
tpcc_iordl_bpool='default'
tpcc_iordl_indices=1-2-3-4-

tpcc_iordr1_imp='none'
tpcc_iordr1_size='703'
tpcc_iordr1_ext='calc'
tpcc_iordr1_nf='calc'
tpcc_iordr1_bs='auto'
tpcc_iordr1_used='-1'
tpcc_iordr1_free='1'
tpcc_iordr1_trans='3'
tpcc_iordr1_autospace='t'
tpcc_iordr1_flg='43'
tpcc_iordr1_fl='22'
tpcc_iordr1_rsize='auto'
tpcc_iordr1_hkey='auto'
tpcc_iordr1_hash='auto'
tpcc_iordr1_bpool='default'
tpcc_iordr1_indices=2-3-1-

tpcc_iordr2_imp='index'
tpcc_iordr2_size='1135'
tpcc_iordr2_ext='calc'
tpcc_iordr2_nf='calc'
tpcc_iordr2_bs='auto'
tpcc_iordr2_used='-1'
tpcc_iordr2_free='25'
tpcc_iordr2_trans='4'
tpcc_iordr2_autospace='t'
tpcc_iordr2_flg='43'
tpcc_iordr2_fl='22'
tpcc_iordr2_rsize='auto'
tpcc_iordr2_hkey='auto'
tpcc_iordr2_hash='auto'
tpcc_iordr2_bpool='default'
tpcc_iordr2_indices=2-3-4-1-

tpcc_istok_imp='index'
tpcc_istok_size='2090'
tpcc_istok_ext='calc'
tpcc_istok_nf='calc'
tpcc_istok_bs='16K'
tpcc_istok_used='-1'
tpcc_istok_free='1'
tpcc_istok_trans='3'
tpcc_istok_autospace='t'
tpcc_istok_flg='43'
tpcc_istok_fl='22'
tpcc_istok_rsize='auto'
tpcc_istok_hkey='auto'
tpcc_istok_hash='auto'
tpcc_istok_bpool='default'
tpcc_istok_indices=1-2-

tpcc_iware_imp='index'
tpcc_iware_size='1'
tpcc_iware_ext='calc'
tpcc_iware_nf='calc'
tpcc_iware_bs='auto'
tpcc_iware_used='-1'
tpcc_iware_free='1'
tpcc_iware_trans='3'
tpcc_iware_autospace='t'
tpcc_iware_flg='43'
tpcc_iware_fl='22'
tpcc_iware_rsize='auto'
tpcc_iware_hkey='auto'
tpcc_iware_hash='auto'
tpcc_iware_bpool='default'
tpcc_iware_indices=1-

tpcc_temp_imp='temp'
tpcc_temp_size='16145'
tpcc_temp_ext='calc'
tpcc_temp_nf='calc'
tpcc_temp_bs='auto'
tpcc_temp_used='-1'
tpcc_temp_free='0'
tpcc_temp_trans='3'
tpcc_temp_autospace='t'
tpcc_temp_flg='43'
tpcc_temp_fl='22'
tpcc_temp_rsize='auto'
tpcc_temp_hkey='auto'
tpcc_temp_hash='auto'
tpcc_temp_bpool='default'
tpcc_temp_indices=no

---------------
--- shutdown.sh
```

```
--------------

#!/bin/sh

sqlplus / as sysdba <<!
shutdown immediate
quit
!

---------------
--- start.sh
--------------

#!/bin/sh

sqlplus / as sysdba <<!
startup pfile=test.ora
quit
!

---------------
--- stepenv.sh
--------------

# forces any env variables we set to be exported
set -a
tpcc_kit=t
tpcc_bench=$PWD
tpcc_scripts=$tpcc_bench/scripts
tpcc_require=$tpcc_scripts/require_vars.sh
tpcc_lcm=$tpcc_scripts/lcm.sh
tpcc_tokilobytes=$tpcc_scripts/tokilobytes.sh
tpcc_fromkilobytes=$tpcc_scripts/fromkilobytes.sh
tpcc_estsize=$tpcc_scripts/estsize.sh
tpcc_notneg=$tpcc_scripts/notneg.sh
tpcc_isneg=$tpcc_scripts/isneg.sh


# need a better way to check for bc, may
# resort to checking each directory in path
# if this doesn't work
#11/7/02 - alex.ni this is causing too many problems
#because systems have bc in some odd place.  typically
#mangled cygwin installs w/ mksnt/cygwin mixes
#if test -x /usr/bin/bc -o -x /bin/bc; then
tpcc_bcexpr=$tpcc_scripts/bcexpr.sh
#else
#tpcc_bcexpr=expr
#fi


# the ksh version is a bit faster, so we want
# to use it if we have ksh.  Otherwise we have
# a compatible version.
#if test -x /bin/ksh; then
#tpcc_createts=$tpcc_scripts/createts.ksh
#else
tpcc_createts=$tpcc_scripts/createts.sh
#fi

tpcc_tabledata=$tpcc_scripts/tabledata.sh
tpcc_load=$tpcc_bench/benchrun/bin/tpccload.exe
tpcc_createtablespaces=$tpcc_scripts/createtablespaces.sh

##
tpcc_sqlplus=cat
tpcc_sqlplus_args='/nolog'
tpcc_internal_connect='connect / as sysdba'
tpcc_user_pass='tpcc/tpcc'
tpcc_dba_user_pass='system/manager'
oracle_dba=system
oracle_dba_password=manager
tpcc_sqlplus=sqlplus

# import options generated by gui
. ${tpcc_bench}/options.sh

#8gb oracle filesize limit (in k)
tpcc_fsize_limit_k=8243200
#2gb - 1k oracle extent limit (in k)
tpcc_extent_limit_k=2048000
#file number limit: 1024
tpcc_file_number_limit=1024

# Runlen calculations should be in hours, but
# this was the old calculation, which assumed
# minutes, and also 8 times:
# tpcc_runlen=`$tpcc_bcexpr 8 \* 60 \* $tpcc_runlen`
# we just want to keep the value as it is.

tpcc_system_size=400M
tpcc_kilo_bytes=1024
#tpcc_logfile_size=`$tpcc_bcexpr 20 + \( $tpcc_scale \)`

if test $tpcc_np -gt 1 ; then
  # 4.69k per commit * 2.1 commit per TPMC ~ 9.85K
  # 9.85k * 30 minutes * 12.5 TPMC per Warehouse = 3693
  tpcc_logfile_size=`$tpcc_bcexpr \( $tpcc_scale \* 3693 \) /
$tpcc_kilo_bytes`
else
  # 2.4k per commit * 2.1 commit per TPMC ~ 5k
  # 5k * 30 minutes * 12.5 TPMC per Warehouse = 1875
  tpcc_logfile_size=`$tpcc_bcexpr \( $tpcc_scale \* 1875 \) /
$tpcc_kilo_bytes`
fi

if test $tpcc_logfile_size -lt 1024; then
  tpcc_logfile_size=1024
fi
tpcc_logfile_size="${tpcc_logfile_size}M"

tpcc_undo_size=`$tpcc_bcexpr 2 \* $tpcc_scale`
if test $tpcc_undo_size -gt 8096; then
  tpcc_undo_size=8096
fi
if test $tpcc_undo_size -lt 512; then
  tpcc_undo_size=512
fi
tpcc_undo_size="${tpcc_undo_size}M"

tpcc_undo_bs=8K

tpcc_statspack_size=`$tpcc_bcexpr 1 \* $tpcc_scale`
if test $tpcc_statspack_size -gt 2048; then
  tpcc_statspack_size=2048
fi
if test $tpcc_statspack_size -lt 300; then
  tpcc_statspack_size=300
fi
tpcc_statspack_size="${tpcc_statspack_size}M"

tpcc_sysaux_size=120M

# fixed table params

#table list (note temp is always at the end since it may use
numbers from other tables, and it's not included in these lists)
tpcc_table_list='ware cust dist hist stok item ordr ordl nord'
tpcc_index_list='iware icust1 icust2 idist istok iitem iordr1
iordr2 iordl inord'
#for these I use average row length, calculated from multi-
blocksize stats.
#we figure out how many new rows we will gain in a run (in
createtablesspaces.sh)
#and add that much to the base tablespace size.
tpcc_hist_growth=51
tpcc_ordr_growth=35
tpcc_nord_growth=regular
#tpcc_ordl_growth=660
tpcc_ordl_growth=900

#i started indices at 1/10th... need an exact figure
tpcc_iordr1_growth=20
tpcc_iordr2_growth=20
tpcc_iordl_growth=66
tpcc_inord_growth=2

tpcc_item_growth=0
tpcc_iitem_growth=0
tpcc_temp_growth=0

tpcc_cust_growth=regular
tpcc_icust1_growth=regular
tpcc_icust2_growth=regular

tpcc_stok_growth=regular
tpcc_istok_growth=regular

tpcc_ware_growth=regular
tpcc_iware_growth=regular

tpcc_dist_growth=regular
tpcc_idist_growth=regular

# minimum size of temp tablespace
tpcc_tempts_min=10240

# for Linux, set appropriate tablespace heuristics
# to set high io tables to have 64 files, and minimize
# others.
if expr $tpcc_os = linux > /dev/null; then
#  for table in $tpcc_table_list $tpcc_index_list temp; do
#    eval "tpcc_${table}_tsfileinc=1"
#  done
  if test $tpcc_numfiles = 0 ; then
    tpcc_numfiles=256
  fi
  tpcc_os=unix

#  tpcc_stok_tsfileinc=64
#  tpcc_cust_tsfileinc=64
#  tpcc_iordl2_tsfileinc=16
#  tpcc_icust2_tsfileinc=16
#  tpcc_iordl_tsfileinc=16
else
#in case someone changes out of linux, and the shell is stuck
  for table in $tpcc_table_list $tpcc_index_list temp; do
    eval "tpcc_${table}_tsfileinc="
  done
fi
```

```
  tpcc_stok_tsfileinc=
  tpcc_cust_tsfileinc=
  tpcc_iordl2_tsfileinc=
  tpcc_icust2_tsfileinc=
  tpcc_iordl_tsfileinc=
#fi


# import local options
. ${tpcc_bench}/localoptions.sh

if expr `echo x$tpcc_no_options` = xt > /dev/null; then
  echo Please modify ${tpcc_bench}/localoptions.sh to configure the
generator.
  exit 1
fi

tpcc_fixordrordl=${tpcc_genscripts_dir}/loadfixordrordl.sh
tpcc_updateordrordl=${tpcc_scripts}/updateordrordl.sh

#tp- get table param.  (that is, $tpcc_tablename_tableparam)
tp(){
  eval echo \""\$tpcc_$1_$2"\"
}

# automatically generated variables
if expr `echo $tpcc_version | cut -b1` = t > /dev/null; then
  tpcc_auto_undo=t
else
  tpcc_auto_undo=f
fi
if expr `echo $tpcc_version | cut -b2` = t > /dev/null; then
  tpcc_autospace_avail=t
else
  tpcc_autospace_avail=f
fi
if expr `echo $tpcc_version | cut -b3` = t > /dev/null; then
  tpcc_queue_avail=t
  tpcc_use_sysaux=t
else
  tpcc_queue_avail=f
  tpcc_use_sysaux=f
fi

# for NT, ORACLE does not like $variables in sql scripts, so we
must
# hardcode these things for it.
if test x$tpcc_os = xnt; then
  tpcc_hardcode=t
else
  tpcc_hardcode=f
fi

# if this is unset we need to make sure it's something anyway
if test x$tpcc_defbs = x; then
  tpcc_defbs=2
fi

# used for loading program
if test x$tpcc_hash_overflow = xt; then
  tpcc_hash_overflow=t
else
  unset tpcc_hash_overflow
fi
if test x$tpcc_overflow = xt; then
  tpcc_hash_overflow=t
else
  unset tpcc_hash_overflow
fi

tpcc_create_steps="buildtpccflags buildcreatets buildcreatedb \
buildcreatetable-ware buildcreatetable-cust buildcreatetable-dist
buildcreatetable-hist buildcreatetable-stok buildcreatetable-item
buildcreatetable-ordr buildcreatetable-ordl buildcreatetable-nord \
buildloadware buildloaddist buildloaditem buildloadhist
buildloadnord buildloadordrordl buildloadcust buildloadstok \
buildcreateindex-iware buildcreateindex-icust1 buildcreateindex-
icust2 buildcreateindex-idist buildcreateindex-istok
buildcreateindex-iitem buildcreateindex-iordr1 buildcreateindex-
iordr2 buildcreateindex-iordl buildcreateindex-inord \
buildstoreprocsql buildspacestats listfiles
"

# remove runscript-loadfixordrordl - shuang, 030626

tpcc_steps="runsqllocal-createdb shutdowndb startupdb-p_build
createuser ddview runscript-createts assigntemp \
  runsql-createtable_ware runsql-createtable_cust runsql-
createtable_dist runsql-createtable_hist runsql-createtable_stok
runsql-createtable_item runsql-createtable_ordr runsql-
createtable_ordl runsql-createtable_nord \
runscript-loadware runscript-loaddist runscript-loaditem runscript-
loadhist runscript-loadnord runscript-loadordrordl runscript-
loadcust runscript-loadstok \
analyze runsql-createindex_iware runsql-createindex_icust1 runsql-
createindex_icust2 runsql-createindex_idist runsql-
createindex_istok runsql-createindex_iitem runsql-
createindex_iordr1 runsql-createindex_iordr2 runsql-
createindex_iordl runsql-createindex_inord \
createstats createstoredprocs createspacestats createmisc"

tpcc_total_files=524

# no longer automatically exports env variables
set +a

# check for problems with configuration
badconf=
for table in $tpcc_table_list; do
  if expr `tp $table imp` = queue > /dev/null; then
    if expr $tpcc_queue_avail = f > /dev/null; then
  echo Table $table may not be a queue, since queues are
  echo are unavailable in the selected Oracle version.
  badconf=t
    fi
  fi
  if expr $tpcc_autospace_avail = f \& `tp $table autospace` = t >
/dev/null; then
    echo Table $table may not use bitmapped space management
    echo since it is not available in the selected Oracle version.
    badconf=t
  fi
done

if test -n "$badconf"; then
  exit 1
fi

# make sure we have everything
if $tpcc_require ORACLE_SID \
 tpcc_tokilobytes tpcc_createts tpcc_lcm\
 tpcc_sqlplus tpcc_internal_connect\
 tpcc_np tpcc_cpu tpcc_os tpcc_runlen tpcc_ldrive tpcc_scale
tpcc_disks_location tpcc_auto_undo tpcc_tempts_min\
 tpcc_system_size tpcc_logfile_size\
 tpcc_undo_size tpcc_undo_bs\
 oracle_dba oracle_dba_password tpcc_dba_user_pass
then exit 1; fi

if test x$tpcc_hardcode != xt; then
  tpcc_disks_location=${tpcc_disks_location}/
#  tpcc_sql_dir='$tpcc_sql_dir'
#  tpcc_statspack_size='$tpcc_statspack_size'
#  tpcc_genscripts_dir='$tpcc_genscripts_dir'
fi

---------------
--- stop.sh
--------------

#!/bin/sh

sqlplus / as sysdba <<!
shutdown immediate;
quit
!

---------------
--- p_build.ora
--------------


compatible = 10.1.0.0.0
db_name = tpcc
control_files =
(/home/oracle/tpcc_disks//control_001,/home/oracle/tpcc_disks//cont
rol_002)
parallel_max_servers = 100
recovery_parallelism = 40
db_files = 329
db_cache_size   = 14576M
db_8k_cache_size  = 3216M
db_16k_cache_size  = 14576M
dml_locks = 500
statistics_level = basic
log_buffer = 1048576
processes = 150
sessions = 150
transactions = 150
shared_pool_size = 4608M
cursor_space_for_time = TRUE
db_block_size = 2048
undo_management = auto
undo_retention = 2
plsql_optimize_level=2


UNDO_TABLESPACE = undo_1
db_4k_cache_size = 20M
filesystemio_options=setall
---------------
--- p_create.ora
--------------

compatible = 10.1.0.0.0
db_name = tpcc
control_files = (/home/oracle/tpcc_disks//control_001,
/home/oracle/tpcc_disks//control_002)
db_block_size = 2048
```

```
db_cache_size = 24576M
db_8k_cache_size = 9216M
log_buffer = 1048576
db_16k_cache_size = 24576M
undo_management = manual
statistics_level = basic
shared_pool_size = 4608M
plsql_optimize_level=2
db_4k_cache_size = 20M

------------------
--- c0d0.in
------------------
# partition table of /dev/cciss/c0d0
unit: sectors

/dev/cciss/c0d0p1 : start=        8, size=128903511, Id=83
/dev/cciss/c0d0p2 : start=128903520, size= 15536640, Id=83
/dev/cciss/c0d0p3 : start=144440160, size= 15536640, Id=83
/dev/cciss/c0d0p4 : start=159976800, size= 74394720, Id= 5
/dev/cciss/c0d0p5 : start=159976808, size= 15536639, Id=83
/dev/cciss/c0d0p6 : start=175513448, size= 15536639, Id=83
/dev/cciss/c0d0p7 : start=191050088, size= 12770399, Id=83
/dev/cciss/c0d0p8 : start=203820488, size= 12770399, Id=83
/dev/cciss/c0d0p9 : start=216590888, size= 12770399, Id=83

------------------
--- c0d2.in
------------------
# partition table of /dev/cciss/c0d2
unit: sectors

/dev/cciss/c0d2p1 : start=        8, size=128903511, Id=83
/dev/cciss/c0d2p2 : start=128903520, size= 15536640, Id=83
/dev/cciss/c0d2p3 : start=144440160, size= 15536640, Id=83
/dev/cciss/c0d2p4 : start=159976800, size= 74394720, Id= 5
/dev/cciss/c0d2p5 : start=159976808, size= 15536639, Id=83
/dev/cciss/c0d2p6 : start=175513448, size= 15536639, Id=83
/dev/cciss/c0d2p7 : start=191050088, size= 12770399, Id=83
/dev/cciss/c0d2p8 : start=203820488, size= 12770399, Id=83
/dev/cciss/c0d2p9 : start=216590888, size=   130559, Id=83
/dev/cciss/c0d2p10: start=216721448, size=  1052639, Id=83
/dev/cciss/c0d2p11: start=217774088, size=   824159, Id=83
/dev/cciss/c0d2p12: start=218598248, size=    48959, Id=83

------------------
--- c0d2.new.in
------------------
# partition table of /dev/cciss/c0d2
unit: sectors

/dev/cciss/c0d2p1 : start=        8, size=128903511, Id=83
/dev/cciss/c0d2p2 : start=128903520, size= 15536640, Id=83
/dev/cciss/c0d2p3 : start=144440160, size= 15536640, Id=83
/dev/cciss/c0d2p4 : start=159976800, size= 74394720, Id= 5
/dev/cciss/c0d2p5 : start=159976808, size= 15536639, Id=83
/dev/cciss/c0d2p6 : start=175513448, size= 15536639, Id=83
/dev/cciss/c0d2p7 : start=191050088, size= 12770399, Id=83
/dev/cciss/c0d2p8 : start=203820488, size= 12770399, Id=83
/dev/cciss/c0d2p9 : start=216590888, size=   130559, Id=83
/dev/cciss/c0d2p10: start=216721448, size=  1080719, Id=83
/dev/cciss/c0d2p11: start=217802168, size=   824159, Id=83
/dev/cciss/c0d2p12: start=218626328, size=    48959, Id=83

------------------
--- c0d3.in
------------------
# partition table of /dev/cciss/c0d3
unit: sectors

/dev/cciss/c0d3p1 : start=        8, size= 16475032, Id=83
/dev/cciss/c0d3p2 : start= 16475040, size= 15593760, Id=83
/dev/cciss/c0d3p3 : start= 32068800, size= 15536640, Id=83
/dev/cciss/c0d3p4 : start= 47605440, size=186766080, Id= 5
/dev/cciss/c0d3p5 : start= 47605448, size= 15536639, Id=83
/dev/cciss/c0d3p6 : start= 63142088, size= 15536639, Id=83
/dev/cciss/c0d3p7 : start= 78678728, size= 14483999, Id=83
/dev/cciss/c0d3p8 : start= 93162728, size= 12770399, Id=83
/dev/cciss/c0d3p9 : start=105933128, size= 12770399, Id=83
/dev/cciss/c0d3p10: start=118703528, size= 12770399, Id=83
/dev/cciss/c0d3p11: start=131473928, size= 12770399, Id=83
/dev/cciss/c0d3p12: start=144244328, size= 16099679, Id=83
/dev/cciss/c0d3p13: start=160344008, size= 16099679, Id=83

------------------
--- c1d6.in
------------------
# partition table of /dev/cciss/c1d6
unit: sectors

/dev/cciss/c1d6p1 : start=        8, size= 34639191, Id=83
/dev/cciss/c1d6p2 : start= 34639200, size= 15536640, Id=83
/dev/cciss/c1d6p3 : start= 50175840, size= 15536640, Id=83
/dev/cciss/c1d6p4 : start= 65712480, size=168659040, Id= 5
/dev/cciss/c1d6p5 : start= 65712488, size= 15536639, Id=83
/dev/cciss/c1d6p6 : start= 81249128, size= 15536639, Id=83
/dev/cciss/c1d6p7 : start= 96785768, size= 12770399, Id=83
/dev/cciss/c1d6p8 : start=109556168, size= 12770399, Id=83
/dev/cciss/c1d6p9 : start=122326568, size= 12770399, Id=83
/dev/cciss/c1d6p10: start=135096968, size= 12770399, Id=83
/dev/cciss/c1d6p11: start=147867368, size=  4202399, Id=83
/dev/cciss/c1d6p12: start=152069768, size= 16099679, Id=83
/dev/cciss/c1d6p13: start=168169448, size= 16099679, Id=83

------------------
--- c1d7.in
------------------
# partition table of /dev/cciss/c1d7
unit: sectors

/dev/cciss/c1d7p1 : start=        8, size=102489591, Id=83
/dev/cciss/c1d7p2 : start=102489600, size= 15536640, Id=83
/dev/cciss/c1d7p3 : start=118026240, size= 15536640, Id=83
/dev/cciss/c1d7p4 : start=133562880, size=100808640, Id= 5
/dev/cciss/c1d7p5 : start=133562888, size= 15536639, Id=83
/dev/cciss/c1d7p6 : start=149099528, size= 15536639, Id=83
/dev/cciss/c1d7p7 : start=164636168, size= 12770399, Id=83
/dev/cciss/c1d7p8 : start=177406568, size= 12770399, Id=83
/dev/cciss/c1d7p9 : start=190176968, size= 12770399, Id=83
/dev/cciss/c1d7p10: start=202947368, size= 12770399, Id=83
/dev/cciss/c1d7p11: start=215717768, size=    48959, Id=83
/dev/cciss/c1d7p12: start=215766728, size=   106079, Id=83

------------------
--- c2d2.in
------------------
# partition table of /dev/cciss/c2d2
unit: sectors

/dev/cciss/c2d2p1 : start=        8, size=128903511, Id=83
/dev/cciss/c2d2p2 : start=128903520, size= 15536640, Id=83
/dev/cciss/c2d2p3 : start=144440160, size= 15536640, Id=83
/dev/cciss/c2d2p4 : start=159976800, size= 74394720, Id= 5
/dev/cciss/c2d2p5 : start=159976808, size= 15536639, Id=83
/dev/cciss/c2d2p6 : start=175513448, size= 15536639, Id=83
/dev/cciss/c2d2p7 : start=191050088, size= 12770399, Id=83
/dev/cciss/c2d2p8 : start=203820488, size= 12770399, Id=83
/dev/cciss/c2d2p9 : start=216590888, size=   269279, Id=83
/dev/cciss/c2d2p10: start=216860168, size=   252959, Id=83
/dev/cciss/c2d2p11: start=217113128, size=    89759, Id=83
/dev/cciss/c2d2p12: start=217202888, size=   106079, Id=83

------------------
--- c3d2.in
------------------
# partition table of /dev/cciss/c3d2
unit: sectors

/dev/cciss/c3d2p1 : start=        8, size= 16589272, Id=83
/dev/cciss/c3d2p2 : start= 16589280, size= 15536640, Id=83
/dev/cciss/c3d2p3 : start= 32125920, size= 15536640, Id=83
/dev/cciss/c3d2p4 : start= 47662560, size=186708960, Id= 5
/dev/cciss/c3d2p5 : start= 47662568, size= 15536639, Id=83
/dev/cciss/c3d2p6 : start= 63199208, size= 15536639, Id=83
/dev/cciss/c3d2p7 : start= 78735848, size= 12770399, Id=83
/dev/cciss/c3d2p8 : start= 91506248, size= 12770399, Id=83
/dev/cciss/c3d2p9 : start=104276648, size= 12770399, Id=83
/dev/cciss/c3d2p10: start=117047048, size= 12770399, Id=83
/dev/cciss/c3d2p11: start=129817448, size= 12770399, Id=83
/dev/cciss/c3d2p12: start=142587848, size= 16099679, Id=83
/dev/cciss/c3d2p13: start=158687528, size= 16099679, Id=83

------------------
--- c3d3.in
------------------
# partition table of /dev/cciss/c3d3
unit: sectors

/dev/cciss/c3d3p1 : start=        8, size= 16475032, Id=83
/dev/cciss/c3d3p2 : start= 16475040, size= 15593760, Id=83
/dev/cciss/c3d3p3 : start= 32068800, size= 15536640, Id=83
/dev/cciss/c3d3p4 : start= 47605440, size=186766080, Id= 5
/dev/cciss/c3d3p5 : start= 47605448, size= 15536639, Id=83
/dev/cciss/c3d3p6 : start= 63142088, size= 15536639, Id=83
/dev/cciss/c3d3p7 : start= 78678728, size= 14483999, Id=83
/dev/cciss/c3d3p8 : start= 93162728, size= 12770399, Id=83
/dev/cciss/c3d3p9 : start=105933128, size= 12770399, Id=83
/dev/cciss/c3d3p10: start=118703528, size= 12770399, Id=83
/dev/cciss/c3d3p11: start=131473928, size=  8608799, Id=83
/dev/cciss/c3d3p12: start=140082728, size= 16099679, Id=83

------------------
--- c4d2.in
------------------
# partition table of /dev/cciss/c4d2
unit: sectors

/dev/cciss/c4d2p1 : start=        8, size= 16475032, Id=83
/dev/cciss/c4d2p2 : start= 16475040, size= 15536640, Id=83
/dev/cciss/c4d2p3 : start= 32011680, size= 15536640, Id=83
/dev/cciss/c4d2p4 : start= 47548320, size=186823200, Id= 5
/dev/cciss/c4d2p5 : start= 47548328, size= 15536639, Id=83
/dev/cciss/c4d2p6 : start= 63084968, size= 15536639, Id=83
/dev/cciss/c4d2p7 : start= 78621608, size= 12770399, Id=83
/dev/cciss/c4d2p8 : start= 91392008, size= 12770399, Id=83
/dev/cciss/c4d2p9 : start=104162408, size= 12770399, Id=83
/dev/cciss/c4d2p10: start=116932808, size= 12770399, Id=83
/dev/cciss/c4d2p11: start=129703208, size= 12770399, Id=83
/dev/cciss/c4d2p12: start=142473608, size= 16099679, Id=83
/dev/cciss/c4d2p13: start=158573288, size= 16099679, Id=83
```

```
------------------
--- c4d3.in
------------------
# partition table of /dev/cciss/c4d3
unit: sectors

/dev/cciss/c4d3p1 : start=        8, size= 16475032, Id=83
/dev/cciss/c4d3p2 : start= 16475040, size= 16475040, Id=83
/dev/cciss/c4d3p3 : start= 32950080, size= 15536640, Id=83
/dev/cciss/c4d3p4 : start= 48486720, size=185884800, Id= 5
/dev/cciss/c4d3p5 : start= 48486728, size= 15536639, Id=83
/dev/cciss/c4d3p6 : start= 64023368, size= 15536639, Id=83
/dev/cciss/c4d3p7 : start= 79560008, size= 14483999, Id=83
/dev/cciss/c4d3p8 : start= 94044008, size= 12770399, Id=83
/dev/cciss/c4d3p9 : start=106814408, size= 12770399, Id=83
/dev/cciss/c4d3p10: start=119584808, size= 12770399, Id=83
/dev/cciss/c4d3p11: start=132355208, size=  8608799, Id=83
/dev/cciss/c4d3p12: start=140964008, size= 16099679, Id=83

------------------
--- log.in
------------------

,44000,
,44000,
,44000,
,,E
,44000,
,44000,
,44000,
,44000,
,44000,
,44000,
,44000,
,44000,
,44000,
,44000,
,44000,

------------------
--- links.sh
------------------
ln -sf /dev/cciss/c0d0p1 /home/oracle/tpcc_disks/ordr_0_0
ln -sf /dev/cciss/c2d0p1 /home/oracle/tpcc_disks/ordr_0_1
ln -sf /dev/cciss/c2d1p1 /home/oracle/tpcc_disks/ordr_0_2
ln -sf /dev/cciss/c3d0p1 /home/oracle/tpcc_disks/ordr_0_3
ln -sf /dev/cciss/c4d0p1 /home/oracle/tpcc_disks/ordr_0_4
ln -sf /dev/cciss/c5d0p1 /home/oracle/tpcc_disks/ordr_0_5
ln -sf /dev/cciss/c0d1p1 /home/oracle/tpcc_disks/ordr_0_6
ln -sf /dev/cciss/c2d3p1 /home/oracle/tpcc_disks/ordr_0_7
ln -sf /dev/cciss/c2d4p1 /home/oracle/tpcc_disks/ordr_0_8
ln -sf /dev/cciss/c3d1p1 /home/oracle/tpcc_disks/ordr_0_9
ln -sf /dev/cciss/c4d1p1 /home/oracle/tpcc_disks/ordr_0_10
ln -sf /dev/cciss/c5d1p1 /home/oracle/tpcc_disks/ordr_0_11
ln -sf /dev/cciss/c0d2p1 /home/oracle/tpcc_disks/ordr_0_12
ln -sf /dev/cciss/c1d1p1 /home/oracle/tpcc_disks/icust1_0_0
ln -sf /dev/cciss/c1d2p1 /home/oracle/tpcc_disks/istok_0_0
ln -sf /dev/cciss/c3d2p1 /home/oracle/tpcc_disks/ordr_0_13
ln -sf /dev/cciss/c4d2p1 /home/oracle/tpcc_disks/roll1
ln -sf /dev/cciss/c5d2p1 /home/oracle/tpcc_disks/hist_0_0
ln -sf /dev/cciss/c0d3p1 /home/oracle/tpcc_disks/hist_0_1
ln -sf /dev/cciss/c1d3p1 /home/oracle/tpcc_disks/hist_0_2
ln -sf /dev/cciss/c1d4p1 /home/oracle/tpcc_disks/hist_0_3
ln -sf /dev/cciss/c3d3p1 /home/oracle/tpcc_disks/hist_0_4
ln -sf /dev/cciss/c4d3p1 /home/oracle/tpcc_disks/hist_0_5
ln -sf /dev/cciss/c5d3p1 /home/oracle/tpcc_disks/hist_0_6

ln -sf /dev/cciss/c0d0p2 /home/oracle/tpcc_disks/stok_0_0
ln -sf /dev/cciss/c2d0p2 /home/oracle/tpcc_disks/stok_0_1
ln -sf /dev/cciss/c2d1p2 /home/oracle/tpcc_disks/stok_0_2
ln -sf /dev/cciss/c3d0p2 /home/oracle/tpcc_disks/stok_0_3
ln -sf /dev/cciss/c4d0p2 /home/oracle/tpcc_disks/stok_0_4
ln -sf /dev/cciss/c5d0p2 /home/oracle/tpcc_disks/stok_0_5
ln -sf /dev/cciss/c0d1p2 /home/oracle/tpcc_disks/stok_0_6
ln -sf /dev/cciss/c2d3p2 /home/oracle/tpcc_disks/stok_0_7
ln -sf /dev/cciss/c2d4p2 /home/oracle/tpcc_disks/stok_0_8
ln -sf /dev/cciss/c3d1p2 /home/oracle/tpcc_disks/stok_0_9
ln -sf /dev/cciss/c4d1p2 /home/oracle/tpcc_disks/stok_0_10
ln -sf /dev/cciss/c5d1p2 /home/oracle/tpcc_disks/stok_0_11
ln -sf /dev/cciss/c0d2p2 /home/oracle/tpcc_disks/stok_0_12
ln -sf /dev/cciss/c1d1p2 /home/oracle/tpcc_disks/stok_0_13
ln -sf /dev/cciss/c1d2p2 /home/oracle/tpcc_disks/stok_0_14
ln -sf /dev/cciss/c3d2p2 /home/oracle/tpcc_disks/stok_0_15
ln -sf /dev/cciss/c4d2p2 /home/oracle/tpcc_disks/stok_0_16
ln -sf /dev/cciss/c5d2p2 /home/oracle/tpcc_disks/stok_0_17
ln -sf /dev/cciss/c0d3p2 /home/oracle/tpcc_disks/iordr2_0_0
ln -sf /dev/cciss/c1d3p2 /home/oracle/tpcc_disks/iordr2_0_1
ln -sf /dev/cciss/c1d4p2 /home/oracle/tpcc_disks/iordr2_0_2
ln -sf /dev/cciss/c3d3p2 /home/oracle/tpcc_disks/iordr2_0_3
ln -sf /dev/cciss/c4d3p2 /home/oracle/tpcc_disks/iordr2_0_4
ln -sf /dev/cciss/c5d3p2 /home/oracle/tpcc_disks/hist_0_7

ln -sf /dev/cciss/c0d0p3 /home/oracle/tpcc_disks/stok_0_18
ln -sf /dev/cciss/c2d0p3 /home/oracle/tpcc_disks/stok_0_19
ln -sf /dev/cciss/c2d1p3 /home/oracle/tpcc_disks/stok_0_20
ln -sf /dev/cciss/c3d0p3 /home/oracle/tpcc_disks/stok_0_21
ln -sf /dev/cciss/c4d0p3 /home/oracle/tpcc_disks/stok_0_22
ln -sf /dev/cciss/c5d0p3 /home/oracle/tpcc_disks/stok_0_23
ln -sf /dev/cciss/c0d1p3 /home/oracle/tpcc_disks/stok_0_24
ln -sf /dev/cciss/c2d3p3 /home/oracle/tpcc_disks/stok_0_25
ln -sf /dev/cciss/c2d4p3 /home/oracle/tpcc_disks/stok_0_26
ln -sf /dev/cciss/c3d1p3 /home/oracle/tpcc_disks/stok_0_27
ln -sf /dev/cciss/c4d1p3 /home/oracle/tpcc_disks/stok_0_28
ln -sf /dev/cciss/c5d1p3 /home/oracle/tpcc_disks/stok_0_29
ln -sf /dev/cciss/c0d2p3 /home/oracle/tpcc_disks/stok_0_30
ln -sf /dev/cciss/c1d1p3 /home/oracle/tpcc_disks/stok_0_31
ln -sf /dev/cciss/c1d2p3 /home/oracle/tpcc_disks/stok_0_32
ln -sf /dev/cciss/c3d2p3 /home/oracle/tpcc_disks/stok_0_33
ln -sf /dev/cciss/c4d2p3 /home/oracle/tpcc_disks/stok_0_34
ln -sf /dev/cciss/c5d2p3 /home/oracle/tpcc_disks/stok_0_35
ln -sf /dev/cciss/c0d3p3 /home/oracle/tpcc_disks/stok_0_36
ln -sf /dev/cciss/c1d3p3 /home/oracle/tpcc_disks/stok_0_37
ln -sf /dev/cciss/c1d4p3 /home/oracle/tpcc_disks/stok_0_38
ln -sf /dev/cciss/c3d3p3 /home/oracle/tpcc_disks/stok_0_39
ln -sf /dev/cciss/c4d3p3 /home/oracle/tpcc_disks/stok_0_40
ln -sf /dev/cciss/c5d3p3 /home/oracle/tpcc_disks/stok_0_41

ln -sf /dev/cciss/c0d0p5 /home/oracle/tpcc_disks/stok_0_42
ln -sf /dev/cciss/c2d0p5 /home/oracle/tpcc_disks/stok_0_43
ln -sf /dev/cciss/c2d1p5 /home/oracle/tpcc_disks/stok_0_44
ln -sf /dev/cciss/c3d0p5 /home/oracle/tpcc_disks/stok_0_45
ln -sf /dev/cciss/c4d0p5 /home/oracle/tpcc_disks/stok_0_46
ln -sf /dev/cciss/c5d0p5 /home/oracle/tpcc_disks/stok_0_47
ln -sf /dev/cciss/c0d1p5 /home/oracle/tpcc_disks/stok_0_48
ln -sf /dev/cciss/c2d3p5 /home/oracle/tpcc_disks/stok_0_49
ln -sf /dev/cciss/c2d4p5 /home/oracle/tpcc_disks/stok_0_50
ln -sf /dev/cciss/c3d1p5 /home/oracle/tpcc_disks/stok_0_51
ln -sf /dev/cciss/c4d1p5 /home/oracle/tpcc_disks/stok_0_52
ln -sf /dev/cciss/c5d1p5 /home/oracle/tpcc_disks/stok_0_53
ln -sf /dev/cciss/c0d2p5 /home/oracle/tpcc_disks/stok_0_54
ln -sf /dev/cciss/c1d1p5 /home/oracle/tpcc_disks/stok_0_55
ln -sf /dev/cciss/c1d2p5 /home/oracle/tpcc_disks/stok_0_56
ln -sf /dev/cciss/c3d2p5 /home/oracle/tpcc_disks/stok_0_57
ln -sf /dev/cciss/c4d2p5 /home/oracle/tpcc_disks/stok_0_58
ln -sf /dev/cciss/c5d2p5 /home/oracle/tpcc_disks/stok_0_59
ln -sf /dev/cciss/c0d3p5 /home/oracle/tpcc_disks/stok_0_60
ln -sf /dev/cciss/c1d3p5 /home/oracle/tpcc_disks/stok_0_61
ln -sf /dev/cciss/c1d4p5 /home/oracle/tpcc_disks/stok_0_62
ln -sf /dev/cciss/c3d3p5 /home/oracle/tpcc_disks/stok_0_63
ln -sf /dev/cciss/c4d3p5 /home/oracle/tpcc_disks/stok_0_64
ln -sf /dev/cciss/c5d3p5 /home/oracle/tpcc_disks/stok_0_65

ln -sf /dev/cciss/c0d0p6 /home/oracle/tpcc_disks/stok_0_66
ln -sf /dev/cciss/c2d0p6 /home/oracle/tpcc_disks/stok_0_67
ln -sf /dev/cciss/c2d1p6 /home/oracle/tpcc_disks/stok_0_68
ln -sf /dev/cciss/c3d0p6 /home/oracle/tpcc_disks/stok_0_69
ln -sf /dev/cciss/c4d0p6 /home/oracle/tpcc_disks/stok_0_70
ln -sf /dev/cciss/c5d0p6 /home/oracle/tpcc_disks/stok_0_71
ln -sf /dev/cciss/c0d1p6 /home/oracle/tpcc_disks/stok_0_72
ln -sf /dev/cciss/c2d3p6 /home/oracle/tpcc_disks/stok_0_73
ln -sf /dev/cciss/c2d4p6 /home/oracle/tpcc_disks/stok_0_74
ln -sf /dev/cciss/c3d1p6 /home/oracle/tpcc_disks/stok_0_75
ln -sf /dev/cciss/c4d1p6 /home/oracle/tpcc_disks/stok_0_76
ln -sf /dev/cciss/c5d1p6 /home/oracle/tpcc_disks/stok_0_77
ln -sf /dev/cciss/c0d2p6 /home/oracle/tpcc_disks/stok_0_78
ln -sf /dev/cciss/c1d1p6 /home/oracle/tpcc_disks/stok_0_79
ln -sf /dev/cciss/c1d2p6 /home/oracle/tpcc_disks/stok_0_80
ln -sf /dev/cciss/c3d2p6 /home/oracle/tpcc_disks/stok_0_81
ln -sf /dev/cciss/c4d2p6 /home/oracle/tpcc_disks/stok_0_82
ln -sf /dev/cciss/c5d2p6 /home/oracle/tpcc_disks/stok_0_83
ln -sf /dev/cciss/c0d3p6 /home/oracle/tpcc_disks/stok_0_84
ln -sf /dev/cciss/c1d3p6 /home/oracle/tpcc_disks/stok_0_85
ln -sf /dev/cciss/c1d4p6 /home/oracle/tpcc_disks/stok_0_86
ln -sf /dev/cciss/c3d3p6 /home/oracle/tpcc_disks/stok_0_87
ln -sf /dev/cciss/c4d3p6 /home/oracle/tpcc_disks/stok_0_88
ln -sf /dev/cciss/c5d3p6 /home/oracle/tpcc_disks/stok_0_89

ln -sf /dev/cciss/c0d0p7 /home/oracle/tpcc_disks/cust_0_0
ln -sf /dev/cciss/c2d0p7 /home/oracle/tpcc_disks/cust_0_1
ln -sf /dev/cciss/c2d1p7 /home/oracle/tpcc_disks/cust_0_2
ln -sf /dev/cciss/c3d0p7 /home/oracle/tpcc_disks/cust_0_3
ln -sf /dev/cciss/c4d0p7 /home/oracle/tpcc_disks/cust_0_4
ln -sf /dev/cciss/c5d0p7 /home/oracle/tpcc_disks/cust_0_5
ln -sf /dev/cciss/c0d1p7 /home/oracle/tpcc_disks/cust_0_6
ln -sf /dev/cciss/c2d3p7 /home/oracle/tpcc_disks/cust_0_7
ln -sf /dev/cciss/c2d4p7 /home/oracle/tpcc_disks/cust_0_8
ln -sf /dev/cciss/c3d1p7 /home/oracle/tpcc_disks/cust_0_9
ln -sf /dev/cciss/c4d1p7 /home/oracle/tpcc_disks/cust_0_10
ln -sf /dev/cciss/c5d1p7 /home/oracle/tpcc_disks/cust_0_11
ln -sf /dev/cciss/c0d2p7 /home/oracle/tpcc_disks/cust_0_12
ln -sf /dev/cciss/c1d1p7 /home/oracle/tpcc_disks/cust_0_13
ln -sf /dev/cciss/c1d2p7 /home/oracle/tpcc_disks/cust_0_14
ln -sf /dev/cciss/c3d2p7 /home/oracle/tpcc_disks/cust_0_15
ln -sf /dev/cciss/c4d2p7 /home/oracle/tpcc_disks/cust_0_16
ln -sf /dev/cciss/c5d2p7 /home/oracle/tpcc_disks/cust_0_17
ln -sf /dev/cciss/c0d3p7 /home/oracle/tpcc_disks/icust2_0_0
ln -sf /dev/cciss/c1d3p7 /home/oracle/tpcc_disks/icust2_0_1
ln -sf /dev/cciss/c1d4p7 /home/oracle/tpcc_disks/icust2_0_2
ln -sf /dev/cciss/c3d3p7 /home/oracle/tpcc_disks/icust2_0_3
ln -sf /dev/cciss/c4d3p7 /home/oracle/tpcc_disks/icust2_0_4
ln -sf /dev/cciss/c5d3p7 /home/oracle/tpcc_disks/icust2_0_5

ln -sf /dev/cciss/c0d0p8 /home/oracle/tpcc_disks/cust_0_18
ln -sf /dev/cciss/c2d0p8 /home/oracle/tpcc_disks/cust_0_19
ln -sf /dev/cciss/c2d1p8 /home/oracle/tpcc_disks/cust_0_20
ln -sf /dev/cciss/c3d0p8 /home/oracle/tpcc_disks/cust_0_21
ln -sf /dev/cciss/c4d0p8 /home/oracle/tpcc_disks/cust_0_22
ln -sf /dev/cciss/c5d0p8 /home/oracle/tpcc_disks/cust_0_23
ln -sf /dev/cciss/c0d1p8 /home/oracle/tpcc_disks/cust_0_24
ln -sf /dev/cciss/c2d3p8 /home/oracle/tpcc_disks/cust_0_25
```

```
ln -sf /dev/cciss/c2d4p8 /home/oracle/tpcc_disks/cust_0_26
ln -sf /dev/cciss/c3d1p8 /home/oracle/tpcc_disks/cust_0_27
ln -sf /dev/cciss/c4d1p8 /home/oracle/tpcc_disks/cust_0_28
ln -sf /dev/cciss/c5d1p8 /home/oracle/tpcc_disks/cust_0_29
ln -sf /dev/cciss/c0d2p8 /home/oracle/tpcc_disks/cust_0_30
ln -sf /dev/cciss/c1d1p8 /home/oracle/tpcc_disks/cust_0_31
ln -sf /dev/cciss/c1d2p8 /home/oracle/tpcc_disks/cust_0_32
ln -sf /dev/cciss/c3d2p8 /home/oracle/tpcc_disks/cust_0_33
ln -sf /dev/cciss/c4d2p8 /home/oracle/tpcc_disks/cust_0_34
ln -sf /dev/cciss/c5d2p8 /home/oracle/tpcc_disks/cust_0_35
ln -sf /dev/cciss/c0d3p8 /home/oracle/tpcc_disks/cust_0_36
ln -sf /dev/cciss/c1d3p8 /home/oracle/tpcc_disks/cust_0_37
ln -sf /dev/cciss/c1d4p8 /home/oracle/tpcc_disks/cust_0_38
ln -sf /dev/cciss/c3d3p8 /home/oracle/tpcc_disks/cust_0_39
ln -sf /dev/cciss/c4d3p8 /home/oracle/tpcc_disks/cust_0_40
ln -sf /dev/cciss/c5d3p8 /home/oracle/tpcc_disks/cust_0_41

ln -sf /dev/cciss/c0d0p9 /home/oracle/tpcc_disks/cust_0_42
ln -sf /dev/cciss/c2d0p9 /home/oracle/tpcc_disks/cust_0_43
ln -sf /dev/cciss/c2d1p9 /home/oracle/tpcc_disks/cust_0_44
ln -sf /dev/cciss/c3d0p9 /home/oracle/tpcc_disks/cust_0_45
ln -sf /dev/cciss/c4d0p9 /home/oracle/tpcc_disks/cust_0_46
ln -sf /dev/cciss/c5d0p9 /home/oracle/tpcc_disks/cust_0_47
ln -sf /dev/cciss/c0d1p9 /home/oracle/tpcc_disks/cust_0_48
ln -sf /dev/cciss/c2d3p9 /home/oracle/tpcc_disks/cust_0_49
ln -sf /dev/cciss/c2d4p9 /home/oracle/tpcc_disks/cust_0_50
ln -sf /dev/cciss/c3d1p9 /home/oracle/tpcc_disks/cust_0_51
ln -sf /dev/cciss/c4d1p9 /home/oracle/tpcc_disks/cust_0_52
ln -sf /dev/cciss/c5d1p9 /home/oracle/tpcc_disks/cust_0_53
ln -sf /dev/cciss/c0d2p9 /home/oracle/tpcc_disks/ware_0_0
ln -sf /dev/cciss/c1d1p9 /home/oracle/tpcc_disks/cust_0_54
ln -sf /dev/cciss/c1d2p9 /home/oracle/tpcc_disks/cust_0_55
ln -sf /dev/cciss/c3d2p9 /home/oracle/tpcc_disks/idist_0_0
ln -sf /dev/cciss/c4d2p9 /home/oracle/tpcc_disks/cust_0_56
ln -sf /dev/cciss/c5d2p9 /home/oracle/tpcc_disks/cust_0_57
ln -sf /dev/cciss/c0d3p9 /home/oracle/tpcc_disks/cust_0_58
ln -sf /dev/cciss/c1d3p9 /home/oracle/tpcc_disks/cust_0_59
ln -sf /dev/cciss/c1d4p9 /home/oracle/tpcc_disks/cust_0_60
ln -sf /dev/cciss/c3d3p9 /home/oracle/tpcc_disks/cust_0_61
ln -sf /dev/cciss/c4d3p9 /home/oracle/tpcc_disks/cust_0_62
ln -sf /dev/cciss/c5d3p9 /home/oracle/tpcc_disks/cust_0_63

ln -sf /dev/cciss/c0d2p10 /home/oracle/tpcc_disks/dist_0_0
ln -sf /dev/cciss/c1d1p10 /home/oracle/tpcc_disks/cust_0_64
ln -sf /dev/cciss/c1d2p10 /home/oracle/tpcc_disks/cust_0_65
ln -sf /dev/cciss/c3d2p10 /home/oracle/tpcc_disks/tpccaux
ln -sf /dev/cciss/c4d2p10 /home/oracle/tpcc_disks/cust_0_66
ln -sf /dev/cciss/c5d2p10 /home/oracle/tpcc_disks/cust_0_67
ln -sf /dev/cciss/c0d3p10 /home/oracle/tpcc_disks/cust_0_68
ln -sf /dev/cciss/c1d3p10 /home/oracle/tpcc_disks/cust_0_69
ln -sf /dev/cciss/c1d4p10 /home/oracle/tpcc_disks/cust_0_70
ln -sf /dev/cciss/c3d3p10 /home/oracle/tpcc_disks/cust_0_71
ln -sf /dev/cciss/c4d3p10 /home/oracle/tpcc_disks/cust_0_72
ln -sf /dev/cciss/c5d3p10 /home/oracle/tpcc_disks/cust_0_73

ln -sf /dev/cciss/c0d2p11 /home/oracle/tpcc_disks/system_1
ln -sf /dev/cciss/c1d1p11 /home/oracle/tpcc_disks/sp_0
ln -sf /dev/cciss/c1d2p11 /home/oracle/tpcc_disks/item_0_0
ln -sf /dev/cciss/c3d2p11 /home/oracle/tpcc_disks/iware_0_0
ln -sf /dev/cciss/c4d2p11 /home/oracle/tpcc_disks/cust_0_74
ln -sf /dev/cciss/c5d2p11 /home/oracle/tpcc_disks/cust_0_75
ln -sf /dev/cciss/c0d3p11 /home/oracle/tpcc_disks/cust_0_76
ln -sf /dev/cciss/c1d3p11 /home/oracle/tpcc_disks/cust_0_77
ln -sf /dev/cciss/c1d4p11 /home/oracle/tpcc_disks/cust_0_78
ln -sf /dev/cciss/c3d3p11 /home/oracle/tpcc_disks/cust_0_79
ln -sf /dev/cciss/c4d3p11 /home/oracle/tpcc_disks/nord_0_0
ln -sf /dev/cciss/c5d3p11 /home/oracle/tpcc_disks/nord_0_1

ln -sf /dev/cciss/c0d2p12 /home/oracle/tpcc_disks/iitem_0_0
ln -sf /dev/cciss/c1d1p12 /home/oracle/tpcc_disks/temp_0_0
ln -sf /dev/cciss/c1d2p12 /home/oracle/tpcc_disks/control_001
ln -sf /dev/cciss/c3d2p12 /home/oracle/tpcc_disks/control_002
ln -sf /dev/cciss/c4d2p12 /home/oracle/tpcc_disks/temp_0_1
ln -sf /dev/cciss/c5d2p12 /home/oracle/tpcc_disks/temp_0_2
ln -sf /dev/cciss/c0d3p12 /home/oracle/tpcc_disks/temp_0_3
ln -sf /dev/cciss/c1d3p12 /home/oracle/tpcc_disks/temp_0_4
ln -sf /dev/cciss/c1d4p12 /home/oracle/tpcc_disks/temp_0_5
ln -sf /dev/cciss/c3d3p12 /home/oracle/tpcc_disks/temp_0_6

ln -sf /dev/cciss/c4d3p12 /home/oracle/tpcc_disks/stok_0_92
ln -sf /dev/cciss/c5d3p12 /home/oracle/tpcc_disks/stok_0_91
ln -sf /dev/cciss/c1d1p13 /home/oracle/tpcc_disks/stok_0_90
ln -sf /dev/cciss/c4d2p13 /home/oracle/tpcc_disks/cust_0_82
ln -sf /dev/cciss/c5d2p13 /home/oracle/tpcc_disks/cust_0_81
ln -sf /dev/cciss/c0d3p13 /home/oracle/tpcc_disks/cust_0_80
ln -sf /dev/cciss/c1d3p13 /home/oracle/tpcc_disks/icust1_0_1
ln -sf /dev/cciss/c1d4p13 /home/oracle/tpcc_disks/istok_0_1
ln -sf /dev/cciss/c3d3p13 /home/oracle/tpcc_disks/dist_0_1

ln -sf /dev/sda1 /home/oracle/tpcc_disks/log_1_1
ln -sf /dev/sda2 /home/oracle/tpcc_disks/log_1_2
ln -sf /dev/sda3 /home/oracle/tpcc_disks/log_1_3
ln -sf /dev/sda5 /home/oracle/tpcc_disks/log_1_4
ln -sf /dev/sda6 /home/oracle/tpcc_disks/log_1_5
ln -sf /dev/sda7 /home/oracle/tpcc_disks/log_1_6
ln -sf /dev/sda8 /home/oracle/tpcc_disks/log_1_7
ln -sf /dev/sda9 /home/oracle/tpcc_disks/log_1_8
```

Appendix C: Tunable Parameters

SEQUENCE OF EVENTS FOR PERFORMANCE RUN

1. Boot up systems clients, servers, & RTEs).
2. Startup the Oracle listener
3. Startup the database on the server using start.sh script.
4. Set priorities and bind interrupts/processe to processors using setrrpri.sh script.
5. Start the RTE.
6. Adjust RTE throttle.

```
---------------------------
ocl1dbinit.ini
---------------------------

[tpcc]
StartTerm=1
DBConnections=200
KMaxterms=81
DeliveryQueues=250
DeliveryThreads=60



---------------------------
ocl2dbinit.ini
---------------------------

[tpcc]
StartTerm=80001
DBConnections=200
KMaxterms=81
DeliveryQueues=250
DeliveryThreads=60



---------------------------
ocl3dbinit.ini
---------------------------

[tpcc]
StartTerm=160001
DBConnections=200
kMaxterms=81
DeliveryQueues=250
DeliveryThreads=60



---------------------------
tnsnames.ora
---------------------------

# tnsnames.ora Network Configuration File:
C:\app\Administrator\product\11.1.0\client_1\network\admin\tnsnames
.ora
# Generated by Oracle configuration tools.

TPCC =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.204.20)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = tpcc)
    )
  )




---------------------------
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

# 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.file-max = 6815744

vm.nr_hugepages=43165
#vm.nr_hugepages=35000
kernel.sem = 250 32000 250 128
net.ipv4.ip_local_port_range = 9000 65500
fs.aio-max-nr=1048576
net.core.rmem_default=262144
net.core.wmem_default=262144
net.core.rmem_max=4194304
net.core.wmem_max=1048576


---------------------------
limits.conf
---------------------------

# /etc/security/limits.conf
#
#Each line describes a limit for a user in the form:
#
#<domain>        <type>  <item>  <value>
#
#Where:
#<domain> can be:
#        - an user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#        - the wildcard %, can be also used with %group syntax,
#                 for maxlogin limit
#
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
#
#<item> can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
#        - nofile - max number of open files
#        - rss - max resident set size (KB)
#        - stack - max stack size (KB)
#        - cpu - max CPU time (MIN)
#        - nproc - max number of processes
#        - as - address space limit
#        - maxlogins - max number of logins for this user
#        - maxsyslogins - max number of logins on the system
#        - priority - the priority to run user process with
#        - locks - max number of file locks the user can hold
#        - sigpending - max number of pending signals
#        - msgqueue - max memory used by POSIX message queues
(bytes)
#        - nice - max nice priority allowed to raise to
#        - rtprio - max realtime priority
#
#<domain>      <type>  <item>         <value>
#

#*               soft    core            0
```

```
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#@student        -       maxlogins       4
oracle    hard  memlock 268435456
oracle    soft  memlock 268435456
oracle    hard  nproc 16384
oracle    soft  nproc 16384
oracle    hard  nofile  65536
oracle    soft  nofile  65536

# End of file
```

```
---------------------------
start.sh
---------------------------

#!/bin/sh

sqlplus / as sysdba <<!
startup pfile=test.ora
quit
!
```

```
---------------------------
test.ora
---------------------------


compatible = 10.1.0.0.0
db_name = tpcc
control_files =
(/home/oracle/tpcc_disks//control_001,/home/oracle/tpcc_disks//cont
rol_002)
processes=800
sessions=1500
transactions=800
db_files=240
dml_locks=800
db_block_size=2048
remote_login_passwordfile=shared
utl_file_dir=*
aq_tm_processes=0
max_dump_file_size=1M
pre_page_sga=TRUE

db_cache_size                   = 11000M
db_keep_cache_size              = 53500M
db_16k_cache_size               = 11000M
db_recycle_cache_size           = 6600M
db_8k_cache_size                = 512M
shared_pool_size                = 3272M

java_pool_size=0
disk_asynch_io=true
db_block_checking=false
db_block_checksum=false
undo_management=auto
undo_retention=0
undo_tablespace=undo_1
transactions_per_rollback_segment=1
plsql_optimize_level=2
replication_dependency_tracking=false
db_file_multiblock_read_count=32
fast_start_mttr_target=0
parallel_max_servers=0
log_buffer=10485760
log_checkpoint_interval=0
log_checkpoint_timeout=0
log_checkpoints_to_alert=true
timed_statistics=false
statistics_level=basic
query_rewrite_enabled=false
trace_enabled=false
result_cache_max_size=0
timed_statistics=false
```

```
---------------------------
setrrpri.sh
---------------------------

#!/bin/sh
# socket 0: 0,1,2,3,4,5.  HT: 6,7,8,9,10,11,12
sleep $1
#sleep 60

/root/tpcc-scripts/rr -p 48 $(ps auxw | grep ora_ | grep -v grep |
awk '{print $2}')
/root/tpcc-scripts/rr -p 48 $(ps auxw | grep oracle | grep -v grep
| awk '{print $2}')
/root/tpcc-scripts/rr -p 48 $(ps auxw | grep LISTENER | grep -v
grep | awk '{print $2}')

# Run dbwr at  higher priority
/root/tpcc-scripts/rr -p 48 $(ps auxw | grep ora_dbw | grep -v grep
| awk '{print $2}')

# Run lgwr at a higher priority
/root/tpcc-scripts/rr -p 49 $(ps auxw | grep ora_lgwr | grep -v
grep | awk '{print $2}')

# Want to bind lgwr to same CPU as log HBA is bound to, cpu 1
taskset -pc 1 $(ps aux | grep ora_lgwr | grep -v grep | awk '{print
$2}')

# Bind dbwr to first socket exclude log cpu(cpu1)
taskset -pc 0,2-5,7-11 $(ps aux | grep ora_dbw0 | grep -v grep |
awk '{print $2}')
taskset -pc 0,2-5,7-11 $(ps aux | grep ora_dbw1 | grep -v grep |
awk '{print $2}')
#taskset -pc 0,2-11 $(ps aux | grep ora_dbw2 | grep -v grep | awk
'{print $2}')
#taskset -pc 0,2-11 $(ps aux | grep ora_dbw3 | grep -v grep | awk
'{print $2}')

#bind all interrupts to all cores on fist socket
#data HBA
echo 01 > /proc/irq/59/smp_affinity
echo 04 > /proc/irq/99/smp_affinity
echo 08 > /proc/irq/130/smp_affinity
echo 010 > /proc/irq/210/smp_affinity
#
##log HBA, cpu1
echo 02 > /proc/irq/131/smp_affinity
#
## eth0 , cpu 0
echo 020 > /proc/irq/139/smp_affinity
echo 020 > /proc/irq/147/smp_affinity
#
/root/cfgcciss $2
#ps -elf > ps.1
```

```
---------------------------
cfgcciss.c
---------------------------

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <linux/cciss_ioctl.h>

int main(int argc, char* argv[]) {

  cciss_coalint_struct cfg_coalint_old;
  cciss_coalint_struct cfg_coalint_new;
  int fd;
  int i, delay;
  char ctrlname[20];

  if (argc<2) {
    printf("useage: %s [interrupt delay]\n", argv[0]);
    exit(0);
  }

  delay = atoi(argv[1]);
  if (delay < 0) {
    printf("delay need to be >=0\n");
    exit(0);
  }

  for (i=0; i<12; i++) {
    if (i==0 ) {
                printf("Skipping Log controller %d\n",i);
                continue;
               }
    sprintf(ctrlname, "/dev/cciss/c%dd0", i);

    if ((fd = open(ctrlname, O_RDWR)) == -1) {
      continue;
    }

    if (ioctl(fd, CCISS_GETINTINFO, &cfg_coalint_old) != 0) {
      printf("error in reading cciss info");
      continue;
    }

    cfg_coalint_new.delay = delay;
    cfg_coalint_new.count = 0;

    if (ioctl(fd, CCISS_SETINTINFO, &cfg_coalint_new) !=0 ||
        ioctl(fd, CCISS_GETINTINFO, &cfg_coalint_new) !=0 ) {
      printf("error in setting cciss");
      continue;
    }

    printf("ctrl #%d: interrupt delay changed from %d to %d\n",
      i, cfg_coalint_old.delay, cfg_coalint_new.delay);
    printf("ctrl #%d: interrupt count changed from %d to %d\n",
      i, cfg_coalint_old.count, cfg_coalint_new.count);
```

```
    close(fd);
  }
}



---------------------------
affinity.c
---------------------------

/*
 * Simple command-line tool to set affinity
 *  Robert Love, 20020311
 *
 *  Modifications by Arun Sharma <arun.sharma@intel.com> for linux-
ia64
 */

#include <stdio.h>
#include <stdlib.h>
#include <sched.h>

#include <linux/unistd.h>
#include <sys/types.h>

/*
 * provide the proper syscall information if our libc
 * is not yet updated.
 */
#ifndef __NR_sched_setaffinity
#define __NR_sched_setaffinity  1231
#define __NR_sched_getaffinity  1232

unsigned long
sched_setaffinity(pid_t pid, unsigned int len, unsigned long
*user_mask_ptr)
{
        return (unsigned long) syscall(__NR_sched_setaffinity, pid,
len, user_mask_ptr);
}

unsigned long
sched_getaffinity(pid_t pid, unsigned int len, unsigned long
*user_mask_ptr)
{
        return (unsigned long) syscall(__NR_sched_getaffinity, pid,
len, user_mask_ptr);
}

#endif

int main(int argc, char * argv[])
{
  unsigned long new_mask;
  unsigned long cur_mask;
  unsigned int len = sizeof(new_mask);
  pid_t pid;

  if (argc != 3) {
    printf(" usage: %s <pid> <cpu_mask>\n", argv[0]);
    return -1;
  }

  pid = atol(argv[1]);
  sscanf(argv[2], "%08lx", &new_mask);

  if (sched_getaffinity(pid, len, &cur_mask) < 0) {
    printf("error: could not get pid %d's affinity.\n", pid);
    return -1;
  }

  printf(" pid %d's old affinity: %08lx\n", pid, cur_mask);

  if (sched_setaffinity(pid, len, &new_mask)) {
    printf("error: could not set pid %d's affinity.\n", pid);
    return -1;
  }

  if (sched_getaffinity(pid, len, &cur_mask) < 0) {
    printf("error: could not get pid %d's affinity.\n", pid);
    return -1;
  }

  printf(" pid %d's new affinity: %08lx\n", pid, cur_mask);

  return 0;
}



---------------------------
rr.c
---------------------------

#include <stdio.h>
#include <unistd.h>
#include <sched.h>
#include <sys/types.h>

main(int argc, char *argv[])
{
        struct sched_param sp;
        int i;

        if (argc < 4) {
                fprintf(stderr, "usage: %s -p <prio> pid...\n",
argv[0]);
                exit(-1);
        }

        if (!strcmp("-p", argv[1])) {
                sp.sched_priority = atoi(argv[2]);
        }

/*      printf("setting priority to: %d\n", sp.sched_priority);*/
        for (i = 3; i < argc; i++) {
                pid_t pid = atoi(argv[i]);
                if (sched_setscheduler(pid, SCHED_RR, &sp) == -1) {
                        perror("sched_setscheduler");
                        exit(-1);
                }
        }

        exit(0);
}



---------------------------
T8hour.pr
---------------------------

echo
########################################################################
#############
#
#
#                       PRTE COMMAND FILE FOR v6-1-0
#
#
#
########################################################################
#############
noecho

disable initialized messages
disable stopped_messages


########################################################################
####
#
#
# PRTE internal variables.
#
#
#
#   set {var} {val}
#
#
#
########################################################################
####
#
# startup_interval must be set (before connects).  It controls the
rate at
#                 which prte user processes are forked off
initially.
#
# start_interval   controls the rate at which prte users are
started when the
#                 "start" command is issued at the console level.
#
# resume_interval  controls how fast resumes are done when the
"resume"
#                 command is issued at the console level.  (NOTE:
resumes
#                 done on the tester's behalf by the master user
are
#                 controlled by the network variable RESUME_DELAY
set below).
#
# stop_interval    controls how fast stops are done when the "stop"
#                 command is issued at the console level.  (NOTE:
stops done
#                 on the tester's behalf by the master user are
controlled by
#                 the network variable STOP_DELAY set below).
#
# type_rate        is the typing delay between each character???

# .0001 .0002 .001 .001 ko
set startup_interval  0.0001
set start_interval  0.0002
set resume_interval 0.03
set stop_interval 0.00003
set type_rate   0.0
```

```
echo

#################################################################
#############
#
#
# Initializing connections.
#
#
#
#################################################################
#############


noecho

#################################################################
####
#
#
# Connect commands.
#
#
#
#   connect {exe} {prte to run on} {# users} {machine to connect
to}  #
#
#
#################################################################
####
#
# delay between the fork for each user process is startup_interval,
# defined above in the "PRTE internal variables" section.
#
# NOTE: The order of the connect statements is relevant since it
determines
#       the order in which prte user id's get assigned.  All
connect statements
#       for tpcc users (web_user, unix_user) should come first,
followed by
#       the connect statement for reduce, followed by the connect
#       statement for tpcc_master.
#

connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1
connect ~tpcc/bin/web_user  n49 1000  ocl1

connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1
connect ~tpcc/bin/web_user  n55 1000  ocl1

connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1
connect ~tpcc/bin/web_user  n57 1000  ocl1

connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2
connect ~tpcc/bin/web_user  n58 1000  ocl2

connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2
connect ~tpcc/bin/web_user  n59 1000  ocl2

connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
```

```
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2
connect ~tpcc/bin/web_user  n60 1000  ocl2

connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3
connect ~tpcc/bin/web_user  n61 1000  ocl3

connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3
connect ~tpcc/bin/web_user  n62 1000  ocl3

connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3
connect ~tpcc/bin/web_user  n63 1000  ocl3

connect ~tpcc/bin/reduce  localhost 1 localhost
connect ~tpcc/bin/tpcc_master localhost 1 localhost
```

```
# NOTE:   timeout MUST be set after the connect statements for it
to
#   take effect.
#
# timeout          is how long prte will wait for an outstanding
command to
#                  return its expected prompt before timing out.
#

set timeout 3600



echo


####################################################################
#############
#
#
# Setting PRTE network variables.
#
#
#
####################################################################
#############


noecho


####################################################################
####
#
#
# PRTE network variables.
#
#
#
#   set network_variable {name} {val}
#
#
#
####################################################################
####


#################################
#                               #
#  FRONT END NETWORK VARIABLES  #
#                               #
#################################
#
# FE_NAMES       A comma seperated list of the front end network
node names.
#
# FE_USER_COUNTS A comma seperated list of the users to run on each
front end.
#               NOTE: The order of counts in this list should
match the order
#          of names in FE_NAMES.
# ADMIN_USER_COUNT is the number of aides to run.
#
# ADMIN_FE_NAMES is a comma seperated list of FEs on which the
aides will
#    operated.

set network_variable FE_NAMES ocl1,ocl2,ocl3

set network_variable FE_USER_COUNTS 80000,80000,80000

set network_variable ADMIN_USER_COUNT 0
set network_variable ADMIN_FE_NAMES cl109


###############################
#                             #
#  REDUCER NETWORK VARIABLES  #
#                             #
###############################
#
# REDUCER_UPDATE_INTERVAL       The interval, in seconds, between
updates
#                               displayed on the console.
#
# REDUCER_HEADER_INTERVAL       Every REDUCER_HEADER_INTERVAL
updates the
#                               column headers will be displayed on
the
#                               console.
#

set network_variable REDUCER_UPDATE_INTERVAL 30
set network_variable REDUCER_HEADER_INTERVAL 6


#################################
#                               #
#  TPCC USER NETWORK VARIABLES  #
#                               #
```

```
#################################
#
# TPCC_USER_LOG_TYPE controls what information the prte users log
to thier
#                   respective files. This is a bit mask.
#
#                   0 - no logging
#                   1 - timer logging (required for asci data
reduction)
#                   2 - sut data logging (required for durability)
#                   4 - script logging (required by the tpcc user
script)
#                   8 - user sut data logging (required by web
users for
#                       error checking)
#
#                   In general, leave this at 12 for web clients
doing binary
#                   data reduction, and 13 for web clients doing
asci data
#                   reduction.
#
# TPCC_USER_FLUSH_LOG is whether or not to flush every write to the
log.
#
# DURABILITY_LOGGING is whether or not to parse new order response
pages for
#                   durability data (to be sent to reducer). This
variable
#        is a boolean so legal values are 0,f,F and 1,t,T.
#
# C_LAST  is the constant value used for customer last names.
#   This value must be chosen with care.  It must be based on
#   the value you used when populating your database.


set network_variable TPCC_USER_LOG_TYPE   0
set network_variable TPCC_USER_FLUSH_LOG  0
#set network_variable TPCC_USER_LOG_TYPE  12
#set network_variable TPCC_USER_FLUSH_LOG 1
set network_variable DURABILITY_LOGGING   0
set network_variable C_LAST     87


#####################################
#                                   #
#  CONFIGURATION NETWORK VARIABLES  #
#                                   #
#####################################
#
# CGI_SCRIPT_NAME is the name of the application to run on the
front ends.
#
# LOAD_DLL_TIMEOUT is how long master should wait (in seconds) for
the dll
#      to initially load before timing out.
#

#set network_variable CGI_SCRIPT_NAME /webacmsxploop.dll
#set network_variable CGI_SCRIPT_NAME /webacmsxploop1500.dll
#set network_variable CGI_SCRIPT_NAME /webacmsxpora84.dll
#set network_variable CGI_SCRIPT_NAME /webacmsxpora8.dll
set network_variable CGI_SCRIPT_NAME  /tpcc/modtpcc.dll
set network_variable LOAD_DLL_TIMEOUT 600

#####################################
#                                   #
#  TEST CONTROL NETWORK VARIABLES   #
#                                   #
#####################################
#
# LOOPBACK_MODE
#   0 - Full end-to-end runs.
#   1 - Back end loopback runs (not implemented yet)
#   2 - Front end loopback runs
#   3 - RTE loopback runs
#
# RUN_NUMBER          is used to tag all output files with the run
number.
#
#                 1 - the primary measurement run.
#                 2 - the repeatability run.
#                 5 - the 50% run.
#                 8 - the 80% run.
#
#                     If you are unsure which run this really will
end up being,
#                     just leave it at 1, and you can rename files
later if you
#                     need to.
#
# VERSION_NUMBER      is used to tag all output files with the
version number.
#                     This is used if you submit files to the
auditor, and then
#                     need to rerun the test, and resubmit files to
the auditor,
#                     for some reason.  For example, you submit a
repeatability
#                     run (RUN_NUMBER 2, VERSION_NUMBER 1) and the
auditor finds
#                     a problem and asks you to re-run the test
(RUN_NUMBER 1,
#                     VERSION_NUMBER 2).
#                     Under normal circumstances, this can just be
left at 1.
#
# TEST_RESULTS_DIR    is the full directory path where the test's
run directory
#                     will be created.  All files (data, log, etc)
will be
#                     put into the run directory.
#
# WARMUP_TIME         is the time in seconds to warm up.  This is
the period
#                     of time after all users have started doing
transactions
#                     and before the measurement interval begins.
#
# STEADY_STATE_TIME is the time for which the test is considered to
be
#     in a steady running state.  It is during this time
#     that all data for measurement intervals will be
#     collected.
#
# MEASUREMENT_INTERVAL  defines the length of a test period within
the
#     STEADY_STATE_TIME. The steady state time may have 1
#     or more measurement intervals. Each measurement
#     interval can be thought of as a seperate measurement
#     run.
#
# COOLDOWN_TIME       is the length of time the test will continue
to run
#                     after the measurement interval is over.  This
time can
#                     be used for doing various types of data
collection by
#                     hand if desired that might otherwise have a
negative
#                     impact on the measured test results. Even if
you are
#                     not collecting any extra data by hand, it is
recommended
#                     that you keep this value at something like
300 or 600
#                     to avoid "clipping" effects at the end of the
measurement
#                     interval.
#
# CHECKPOINT_INTERVAL is the total time between the start of each
#     checkpoint command.
#
# CKPT_PROXIMITY_ADDITIONAL_OFFSET  This value will be added to any
#     required proximity time to give the actual start
#     time of the first checkpoint in the measurement
#     interval.
#
# LOGIN_DELAY         is the delay between logins on a per front
end basis.
#                     NOTE: This is similar to the prte internal
variable
#                     resume_interval (tpcc users start, then
immediately
#                     pause, so the act of logging in is just a
resume) but
#                     not exactly the same.
#
# RESUME_DELAY        is the delay between resumes on a per front
end basis.
#                     NOTE: This is similar to the prte internal
variable
#                     resume_interval but not exactly the same.
#
# STOP_DELAY          is the delay between stops on a per front end
basis.
#                     NOTE: This is similar to the prte internal
variable
#                     stop_interval but not exactly the same.
#
# SYNC_OFFSET   how many users we'll allow to have outstanding
#     when doing crowd control.
#
# SYNC_UPDATE   how often user login/resume/stop progress is
printed
#     out to the console (heartbeat of user synchronization
#     effectively).
#
# MSG_TIMEOUT   how long we'll wait for status and sync messages.
#

set network_variable LOOPBACK_MODE   0

set network_variable RUN_NUMBER          1
set network_variable VERSION_NUMBER      1
set network_variable TEST_RESULTS_DIR    /results/
#set network_variable LOG_DIR    /home/tpcc/logs/
#set network_variable RUN_DIR    /home/tpcc/logs/
```

```
set network_variable WARMUP_TIME    3600.0
set network_variable STEADY_STATE_TIME    28800.0
set network_variable MEASUREMENT_INTERVAL 28800.0
set network_variable COOLDOWN_TIME    300.0

#set network_variable WARMUP_TIME   2700.0
#set network_variable STEADY_STATE_TIME   10800.0
#set network_variable MEASUREMENT_INTERVAL  10800.0
#set network_variable COOLDOWN_TIME   600.0

set network_variable CHECKPOINT_INTERVAL 0
set network_variable CKPT_PROXIMITY_ADDITIONAL_OFFSET 0
# .05 .08 .04 ko
set network_variable LOGIN_DELAY         0.002
#set network_variable RESUME_DELAY        0.08 #w2k lnx 10i
set network_variable RESUME_DELAY        0.005
set network_variable STOP_DELAY          0.001
# 100 5000
set network_variable SYNC_OFFSET    256
set network_variable SYNC_UPDATE    2000

set network_variable MSG_TIMEOUT    1200.0

set network_variable NO_THINK_TIME 12.05
set network_variable NO_THINK_TIME_UPDATE_INTERVAL 15.0


set network_variable DY_MIX_PERCENT 4.001
set network_variable OS_MIX_PERCENT 4.001
set network_variable SL_MIX_PERCENT 4.001
set network_variable PT_MIX_PERCENT 43.003

# In general, the SEED network variable should not be set. A random
value
# based on process id and the current time will be used.  This
varaible is
# really only exposed in case you want to exactly reproduce a
previous run
# using that previous run's seed.

#set network_variable SEED    123127777

#####################################################################
####
#
# AUDIT UTILITIES -- these are the replacement for the audit
# shell scripts -- they currently only work for Oracle on DUNIX.
# They do the following:
#     Collect logspace info
#     Write data to audit table for later use in runcheck
#     Collect checkpoint info
#     Run optional custom scripts on back-end before or after the
test
#     For Oracle, collect bstat/estat (optional)
#
#####################################################################
####
#
# GET_ALL_AUDIT_FILES if True (or 1) will create the following:
#     Audit table for doing runcheck later
#     m1log.v1 -- a before & after snapshot of the logsize
#
# BE_NAMES           Comma-separated list of back-ends
#
# BE_USERNAME        Username to use when logging into back-ends
#                    NOTE: you must have .rhosts configured so no
password
#                    is needed.
#
# DATABASE_TYPE      Oracle, Sybase or MsSql
#
# DATABASE_USERNAME  Username and password for database.
# DATABASE_PASSWORD  Defaults are: tpcc/tpcc for Oracle and sa/<no-
passwd>
#                    for Sybase and MsSql
#
# Optional variables -- if you don't want them, comment them out or
set to ""
#
# ORACLE_STATS_SCRIPT_PATH
#                    Path to directory on back-end containing
Oracle's
#                    orst_<xxx>.sql files.
#                    For example: $ORACLE_HOME/bench/gen/sql
#
# CUSTOM_BEFORE_TEST_SCRIPT
# CUSTOM_AFTER_TEST_SCRIPT
#                    Path of executable file on back-end to be run
before/after
#                    the test.  For example, if you wanted to run
processor
#                    affinity and load some stored procedures
before a test,
#                    you could put the commands in a shell script
on the BE
#                    and call put the path to that shell script
into the
#                    CUSTOM_BEFORE_TEST_SCRIPT variable
#
#####################################################################
####

set network_variable GET_ALL_AUDIT_FILES FALSE


set network_variable BE_NAMES         pencil
set network_variable BE_USERNAME  oracle

set network_variable DATABASE_TYPE  oracle
set network_variable DATABASE_USERNAME  tpcc
set network_variable DATABASE_PASSWORD  tpcc

set network_variable MAX_W_ID  24000
set network_variable BASE_W_ID  1

#set network_variable DATABASE_TYPE MsSql
#set network_variable DATABASE_USERNAME  tpcc
#set network_variable DATABASE_PASSWORD  tpcc

set network_variable ORACLE_STATS_SCRIPT_PATH   ""
set network_variable CUSTOM_BEFORE_TEST_SCRIPT  ""
set network_variable CUSTOM_AFTER_TEST_SCRIPT   ""



#####################################################################
####


# now start all the users.  delay between each user being started
is controled
# by start_interval defined above in the "PRTE internal variables"
section.
#



echo


#####################################################################
#############
#
#
# Starting all PRTE users (may take a while, depending on the
number of users) #
#
#
#####################################################################
#############

noecho

#disable stop

#start
```

Appendix D:
Third Party Letters

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

August 4, 2010

Hewlett-Packard Corporation
Bryon Georgson
20555 SH 249
Houston, TX 77070

Here is the information you requested regarding pricing for several Microsoft products to be used in conjunction with your TPC benchmark testing.

All pricing shown is in US Dollars ($).

Part Number	Description	Unit Price	Quantity	Price
P73-03883	**Windows Server 2008 R2 Standard Edition** *Server License with 5 CALs* *No Discounts Applied*	$1,029	3	$3,087
127-00012	**Visual Studio Standard 2005** *Full License* *No Discount Applied*	$250	1	$250
N/A	**Microsoft Problem Resolution Services** *Professional Support* *(1 Incident)*	$245	1	$245

A list of Microsoft's resellers can be found at
http://www.microsoft.com/products/info/render.aspx?view=22&type=mnp&content=22/licensing

All products listed above are currently orderable and available.

Defect support is included in the purchase price. Additional support is available from Microsoft PSS on an incident by incident basis at $245 per call.

This quote is valid for the next 90 days.

Reference ID: PCbyge1008040000000997.

DeepSurplus

New, Surplus, Closeout & Overstocked Cabling Supplies

Phone C
949 6
Monday-F

School ar
Fax (949)

HOME | Network Cabling & Structured Wiring | Home Theater (Audio/Video) | Computer Cabling & Accessories

Network Cabling & Structured Wiring > Network Patch Cables > Ethernet CAT6 Network Patch Cables > Ethernet CAT6 Patch Cables; 7ft >

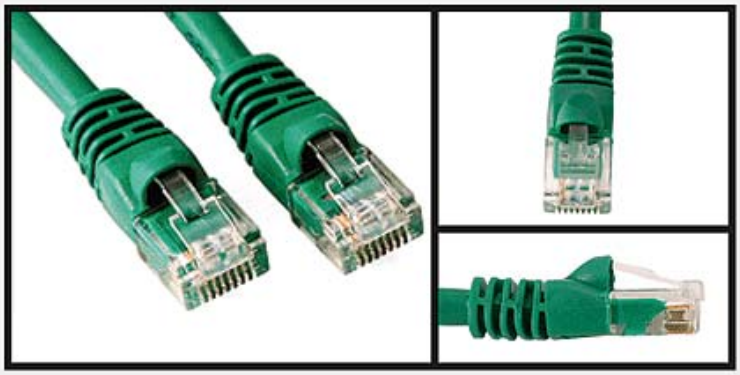
Register | Log In

Shopping Cart
- Your Cart is Empty
- View Cart

Search
Go

- Bulk Cable
- Network Patch Cables
  - Ethernet Cat5E Network Patch Cables
  - Ethernet CAT6 Network Patch Cables
    - Ethernet CAT6 Patch Cables; 6in
    - Ethernet CAT6 Patch Cables; 9in
    - Ethernet CAT6 Patch Cables; 1ft
    - Ethernet CAT6 Patch Cables; 2ft
    - Ethernet CAT6 Patch Cables; 3ft
    - Ethernet CAT6 Patch Cables; 5ft
    - **Ethernet CAT6 Patch Cables; 7ft**
      - 7ft Gray Cat 6 Patch Cable, Molded $1.20 each
      - 7ft Pink Cat 6 Patch Cable, Molded As low as $1.16

7ft Green Cat 6 Patch Cable, Molded
As low as $1.16



Quantity	Price
1 – 2	$1.70
3 – 4	$1.67
5 – 7	$1.63
8 – 11	$1.61
12 – 15	$1.59
16 – 19	$1.58
20 – 29	$1.57
30 – 39	$1.55
40 – 49	$1.54
50 – 99	$1.52
100 – 199	$1.34
200 – 299	$1.31
300 – 499	$1.27
500 – 999	$1.19
1000 +	$1.16

1 Purchase

ADD TO CART TO ESTIMATE SHIPPING

ShareThis New

Meets or exceeds the ANSI/TIA/EIA-568-B.2-1 standard for CAT 6 CMR, communication riser cable, and c
pulls, special CAT 6 rated gold plated RJ45 connectors on each end and boots to protect the tab of the RJ

Part #: CB242-7GN Tell a Friend

Condition: New
Mfg: Abergetty

21. Netgear GS108NA

Part number: 736585-V
Mfg.Part No.: GS108NA

Netgear GS108 ProSafe 8-Port Gigabit Desktop Switch

Add to Wishlist Email a Friend

Our Price:
$75.99

BUY NOW

Back to list
Related Products

Description

Now you can have a powerful, high-speed network on a small scale. NETGEAR's GS108 Gigabit Ethernet Switch lets you build a system that provides a full, dedicated 1000, 100, or 10 Mbps connection so you can move very large files across your network instantly. The GS108 also lets you painlessly integrate 10, 100, and 1000 Mbps devices. This potent switch is packed with ease-of-use features to simplify your workday experience. Its compact design fits neatly into small work spaces and, since it's self-cooling without a fan, it runs silently and unobtrusively.

Specifications

Manufacturer	NETGEAR
Manufacturer Part #	GS108NA
Product Description	Ga108 8port 10/100/1000 Switch
Device Type	Switch
Form Factor	External
Approximate Dimensions (WxDxH)	3.7 in x 4 in x 1.1 in
Approximate Weight	9.9 oz
Localization	North America
Ports Qty	8 x Ethernet 10Base-T, Ethernet 100Base-TX, Ethernet 1000Base-T
Data Transfer Rate	1 Gbps
Data Link Protocol	Ethernet, Fast Ethernet, Gigabit Ethernet
Communication Mode	Half-duplex, full-duplex
Features	Full duplex capability, auto-sensing per device, auto-uplink (auto MDI/MDI-X), store and forward
Compliant Standards	IEEE 802.3u, IEEE 802.3i, IEEE 802.3wb, IEEE 802.1p
Manufacturer Warranty	Limited lifetime warranty

Limit two per customer. To view terms and conditions, and to submit your rebate, please visit us at www.mcsatherebates.com and reference promotional code 08-62381. All rebate submissions must be postmarked within 30 days of purchase date.

Related Products

BUY NOW

01. Netgear FA311
Netgear FA311 10/100Bt Fast Enet PCI RJ45Network adapter $26.99

02. Netgear FS605NA
Netgear FS605 5-Port 10/100 Switch $30.99

03. Netgear GA311NA

Appendix E: Database Pricing

From: mary beth.pierantoni [mary.beth.pierantoni@oracle.com]
Sent: Wednesday, July 21, 2010 2:17 PM
To: Georgson, Bryon
Subject: Oracle pricing

Product	**Price**	**Quantity**	**Extended Price**
Oracle Database 11g Standard Edition One, Per Processor, Unlimited Users, **3 years**	$2900	1*	$2900
Premium Support for **3 years**	$1276	3	$3828
Oracle Unbreakable Linux Support Program: Enterprise Linux Basic Limited for **3 years**	$1497	1	$1497
Oracle TOTAL			**$8225**

Appendix F:TPC-Energy Disclosure Report

A.1. TPC-Energy Clause 2-related items (Methodology)

A.1.1. Minimum abient temperature
The minimum abient temperature must be disclosed.

Minimum temperature reported by EMSC = 20.88 C

A.1.2. External electric power source characteristics
The characteristics of the external electric power source must be disclosed. In particular, the voltage, frequency in Hertz, and phase information must be reported.

The external electric power source has the following characteristics: 208V, 60Hz, and single phase.

A.1.3. Air-pressure alterations
A statement is required that assures that nothing was done to alter the air-pressure in the measurement environment.

Nothing was done to alter the air-pressure in the measurement environment.

A.1.4. Temperature measurement
A description of where the temperature was measured and how it was determined that this was representative of the lowest ambient temperature is required.

Temperature was measured at the SUT air inlet and the air conditioning returns blow cold air at SUT air inlet.

A.1.5. Cooling method
If a method of cooling other than circulation of ambient air is employed in the REC, a statement describing this method must be included.

No other method of cooling was used.

A.1.6. PTD license
To be compliant with licenses associated with EMS, the following statement must be included in every FDR which contains a TPC-Energy Metric:

The power and temperature characteristics of the MEC were measure using TPC's Energy Measurement Software (EMS). This includes the EMS-PTD, a modified version of the SPEC PTDaemon, which is provided under license from the Standard Performance Evaluation Corporation (SPEC).

A.2. TPC-Energy Clause 3-related items (Metrics)

A.2.1. Primary Metric
The normalized work derived from the Performance Metric (as described in Clause 3.2.1) must be disclosed.

4.22 watts / KtpmC

The computation for total energy used for each measurement segment that contributes to a Performance Metric must be disclosed. If the energy of the entire Priced Configuration is not derived from direct measurements, the methods for deriving the energy for components that were not measured must be disclosed (See Clause 7.3.3.4).

			Full Load Energy									
PMU	Full Load Average Watts Reading	% of Reading Uncertainty	Watts Reading Correction	Wattage Range Setting	% of Range Uncertainty	Wattage Range Correction	Total Wattage Correction	Accuarcy Correction Factor	Reported Watt - Seconds	Adjusted Watt - Seconds	Reported Seconds	Adjusted Average Watts
SUT PMU-1	745.92	0.10%	+0.75	1500	0.10%	+1.50	+2.25	0.30%	5,371,385	5,387,558	7,200	748.27
SUT PMU-2	361.75	0.10%	+0.36	1500	0.10%	+1.50	+1.86	0.51%	2,604,993	2,618,400	7,200	363.67
4 x Monitor NamePlate	112	0.00%	+0.00	0	0.00%	+0.00	+0.00	0.00%	806,400	806,400	7,200	112.00
REC Total	1219.67								8,782,778	8,812,358		1223.94

All monitors power consumption in the measurement were calculated using nameplate values.

The duration of each measurement that produces a Performance Metric must be disclosed.

The duration for the measured runs were 120 minutes. The duration for the idle measurements were 10 minutes.

The average power requirement for each measurement that produces one of thes metrics.

TPC-C measurement interval and idle average power requirements:

	Watts / KtpmC	Full Load Avg Watts	Full Load % of REC	Full Load Watt Mins.	Idle Avg. Watts	Idle % of REC
SUT PMU-1	2.57	745.92	60.9%	89,523	694.33	60.8%
SUT PMU-2	1.25	361.75	29.6%	43,417	329.43	28.9%
Total REC	4.22	1224	100%	146873	1141	100%

The TPC-Energy Primary Metric must be disclosed, including the calculation that is used to derive it.

Total REC Energy Consumption = 146,873 Watt-minutes
SUT Total work = 34,804,800 transactions

146,873 Watt-minutes / 34,804,800 transactions = 0.00422 Watts / tpmC
0.00422 Watts / tpmC * 1000 = 4.22 watts / KtpmC

A.2.2. Secondary Metrics At Reported Performance

If the TPC-Energy Secondary Metrics are reported, the components of the REC that are included in each subsystem must be identified. This can be achieved with separate lists to be included in the FDR or with a specific designation in the price spreadsheet. Every component in the REC that consumes energy must be included in exactly one subsystem.

No TPC-Energy Secondary Metrics are being reported.

A.2.3. Idle Power reporting

The Idle Power measurement / calclulation for thee REC must be reported as numerical quantities.

The Idle Power measurement for REC = 1,141

If TPC-Energy Secondary Metrics are reported, then the Idle Power measurement / calculation for each subsystem must also be reported as numerical quantities.

No TPC-Energy Secondary Metrics are being reported.

The length of time between the conclusion of the performance measurement and the start of the idle measurement must be reported.
The Idle measurement began 20 minutes after the conclusion of the performance measurement.

The duration of the idle measurement must be reported.

The Idle measurement duration was 10 minutes.

A statement is required that assures that the system is in a state that is ready to run the Application(s) of the benchmark for the duration of the idle measurement.

The system is in a state that is ready to run the Application(s) of the benchmark for the duration of the idle measurement. This was verified by executing a single transaction after the idle measurement period was completed. The transaction time of this single transaction was compared to the 90th percentile and found to meet the required specifications.

A.2.4. Disclosure requirements when only part of the REC is measured for power.

If all PMU's of the REC are not measured for energy use, the FDR must include a description of which PMUs of REC were measure with a power analyzer. The FDR must disclose which PMUs of the REC were computed based on the energy measurements of similar PMUs. A diagram must be included that identifies the portions of the configuration which were measured for energy use and which were calculated. This diagram may be combined with other diagrams required by the TPC Benchmark Standard.

- *The method used to determing which PMUs were measured must be disclosed.*
- *The power values for the each partial-REC measurement for duration of the performance and idle measurements must be disclosed.*
- *The calculation for the power requirements of the entire REC and, if applicable, each subsystem must be disclosed.*

			Full Load Energy									
PMU	Full Load Average Watts Reading	% of Reading Uncertainty	Watts Reading Correction	Wattage Range Setting	% of Range Uncertainty	Wattage Range Correction	Total Wattage Correction	Accuarcy Correction Factor	Reported Watt - Seconds	Adjusted Watt - Seconds	Reported Seconds	Adjusted Average Watts
SUT PMU-1	745.92	0.10%	+0.75	1500	0.10%	+1.50	+2.25	0.30%	5,371,385	5,387,558	7,200	748.27
SUT PMU-2	361.75	0.10%	+0.36	1500	0.10%	+1.50	+1.86	0.51%	2,604,993	2,618,400	7,200	363.67
4 x Monitor NamePlate	112	0.00%	+0.00	0	0.00%	+0.00	+0.00	0.00%	806,400	806,400	7,200	112.00
REC Total	**1219.67**								**8,782,778**	**8,812,358**		**1223.94**

			Idle Load Energy									
PMU	Idle Average Watts Reading	% of Reading Uncertainty	Watts Reading Correction	Wattage Range Setting	% of Range Uncertainty	Wattage Range Correction	Total Wattage Correction	Accuarcy Correction Factor	Reported Watt / Seconds	Adjusted Watt / Seconds	Reported Seconds	Adjusted Average Watts
SUT PMU-1	694.33	0.10%	+0.69	1500	0.10%	+1.50	+2.19	0.32%	417,290	418,608	600	697.68
SUT PMU-2	329.43	0.10%	+0.33	1500	0.10%	+1.50	+1.83	0.56%	197,985	199,084	600	331.81
4 x Monitor NamePlate	112	0.00%	+0.00	0	0.00%	+0.00	+0.00	0.00%	67,200	67,200	600	112.00
REC Total	**1135.76**								**682,475**	**684893**		**1141.49**

The four monitors power consumption were calculated using the nameplate values. All the rest of the PMUs are measured in the charts above.

A.2.5. Disclosure requirements when component substitution is used.

If the TPC Benchmark Standard allows the Priced Configuration to differ from the Measured Configuration, the methods used to assign energy or power characteristics to the substitute components must be disclosed.
The method use to determine which PMUs were measured must be disclosed.
The power values for the each partial-REC measurement for dureation of the performance and idle measurements must be disclosed

.
There were no component substitutions.

A.3. TPC-Energy Clause 4-related items (Drivers/Controller0

A statement indicating the version of EMS used must be included in the FDR, including a statement that no alterations of this code swere made for the benchmark, except as specified by Clause 7.3.4.3. This includes levels for the EMS-PTD Manager, EMS-PTD and EMS-controller.

The EMS version used was 1.1.1 and no alterations were made.

Input paramters for the EMS software must be disclosed.

The following EMS script was used to configure the EMS software

```
start log power-perfrun > DBServer+Storage
sleep 5
start log pwer-perfrun-clients > clients
sleep 5
start log temperature-perfrun > TempML350G6
sleep 5
time sync
sleep 5
set sample 1000 > DBServer+Storage
sleep 5
set sample 1000 > clients
sleep 5
set sample 10000 > REC-DBServerTempML350G6
sleep 5
start data “run 1”
```

Any changes in the EMS components must be documented. Documentation must include a description of the issue, the reason the change was necessary for disclosure of the Result, and the changes made to resolve it. Any change to the TPC-Provided Code must be included with the submission as a Supporting File.

No changes to EMS components were made.

A.4. TPC-Energy Clause 6-related items (Instrumentation)

A.4.1. Power Analyzer Information

			Power Analyzer Specifications and Settings						
PMU	Make	Model	Serial Number	Calibration Date	Wattage Range Setting	Voltage Range Setting	Current Range Setting	% of reading	% of Range
SUT PMU-1	Yokogawa	WT210	91k208942	3/19/2010	1500	300	5	0.10%	0.10%
SUT PMU-2	Yokogawa	WT210	91GB45372	12/10/2009	1500	300	5	0.10%	0.10%
4 x Monitor NamePlate	N/A								

A.4.2. Temperature Sensor Information

Make and model.
Acccuracy and the source of information.

Digi Watchport/H Temperature probe.

Temperature accuracy from Manufacturers Datasheet:
+/- 3.6º F (+/- 2º C) at -40º F to 14º F (-40º C to -10º C)
+/- 0.9º F (+/- 0.5º C) at -14º F to 185º F (-10º C to -85º C)

\

A.5. TPC-Energy Clause 8-related items.

A.5.1 Auditor's attestation letter.

August 13, 2010

Mr. Bryon Georgson
Database Performance Engineer
Hewlett-Packard Company
20555 SH 249
Houston, TX 77070

I have verified by remote the TPC Benchmark™ C for the following configuration:
Platform: HP ProLiant ML350 G6
Database Manager: Oracle Database 11g Release 2 Standard Edition
Operating System: Oracle Enterprise Linux
Transaction Monitor: Microsoft COM+

System Under Test:				
CPU's	Memory	Disks (total)	90% Response	TpmC
1 Intel Xeon 6 core @ 2.67 Ghz	Main: 96 GB	24 @ 120 GB SSD 24 @ 500 GB 1 @ 72 GB (OS)	1.11	**290,040**
Clients: 3 ML150 G6				
1 Intel quad core @ 2.13 Ghz	2 GB	1 @ 250 GB	NA	**NA**

In addition to the performance metric, the energy consumption was measured during the performance runs in compliance with the TPC-Energy specification.

- The power analyzers used were verified to be approved and calibrated within one year prior to this measurement.

- The energy measurements met all requirements of the specification unless an exception is noted below.
- The calculations for the TPC-Energy Primary Metric were verified as completed correctly.
- The EMS software was verified to be the correct version and without any changes.
- The executive summary page and the FDR were verified for accuracy.

Auditor's Notes: None

Sincerely,

Lorna Livingtree

A.6. TPC-Energy Supporting Files Index

Clause	Description	Path
7.4.1	PTDM Log Files (txt)	006.report.full-power-perfrun.txt
7.4.1	PTDM Log Files (txt)	006.report.full-power-perfrun-clients.txt
7.4.1	PTDM Log Files (txt)	006.report.full-temperature-perfrun.txt
7.4.1	PTDM Log Files (txt)	006.report.idle-power-perfrun.txt
7.4.1	PTDM Log Files (txt)	006.report.idle-power-perfrun-clients.txt
7.4.1	PTDM Log Files (txt)	006.report.idle-temperature-perfrun.txt
7.4.1	PTDM Log Files (txt)	006.report.perf-power-perfrun.txt
7.4.1	PTDM Log Files (txt)	006.report.perf-power-perfrun-clients.txt
7.4.1	PTDM Log Files (txt)	006.report.perf-temperature-perfrun.txt
7.4.1	PTDM Log Files (xml)	power-perfrun-006.xml
7.4.1	PTDM Log Files (xml)	power-perfrun-clients-006.xml
7.4.1	PTDM Log Files (xml)	temperature-perfrun-006.xml
N/A		ML350G6-TPC-C-Energy_calculations2.xls