Hewlett-Packard Company

TPC Benchmark™ C

Full Disclosure Report
for
HP ProLiant DL580 - O2000 32P
Using
Oracle9i Enterprise Edition Release 2 with Real Application Clusters and Partitioning Options and Windows 2000 Advanced Server

Second Edition
September 2002

Second Edition – September 2002

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., 2002

Parallel Database Cluster Model O2000 and ProLiant are registered trademarks of Hewlett Packard Company.

ORACLE 9i Real Application clusters, Pro*C, PL/SQL, SQL*Net, SQL*Plus are registered trademarks of Oracle Corporation.

Windows and Windows 2000 Advanced Server are trademarks of Microsoft 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 5.0, released March 7, 2001.

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.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark C test conducted on the HP ProLiant DL580 - O2000 32P. The operating system used for the benchmark was Windows 2000 Advanced Server. The DBMS used was Oracle 9i Enterprise Edition Release 2 with Real Application Clusters and Partitioning Options.

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:

137,260.89 tpmC
$18.46 per tpmC
Available as of September 6, 2002.

Standard and Executive Summary Statements

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

Auditor

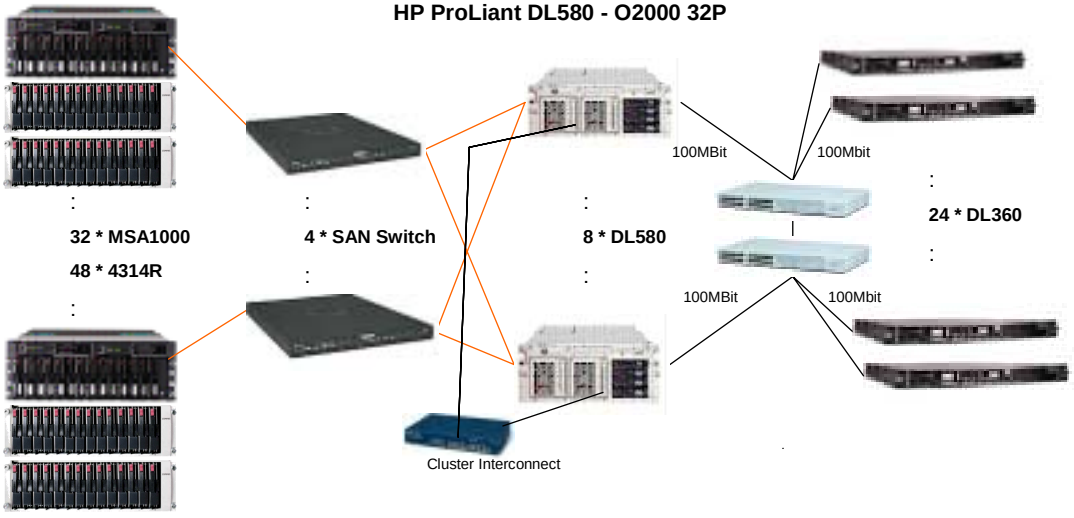
The benchmark configuration, environment and methodology were audited by Lorna Livingtree of Performance Metrics Inc. to verify compliance with the relevant TPC specifications.

	HP ProLiant DL580 - O2000 32P C/S	TPC-C Rev. 5.0 Report Date: September 6, 2002

Total System Cost	TPC-C Throughput	Price Performance	Availability Date
$2,533,095	**137,260.89 tpmC**	**$18.46**	**September 6, 2002**

Processors	Database Manager	Operating System	Other Software	Number of Users
32 PIII X900/ MHz/2MB-Servers 48 PIII 1000 MHz/ 133/256K - Clients	Oracle 9i Enterprise Edi. Rel. 2 with Real Application Clusters and Partitioning Options	Windows 2000 Advanced Server	BEA Tuxedo 6.5 CTS Microsoft Visual C++	**120240**

HP ProLiant DL580 - O2000 32P



System Components	Server Quantity	Server Description	Clients Quantity	Clients Description
Processor	32	Pentium III X900MHz/2MB	48	Pentium III 1000 MHz/133/ 256K
Memory	8	8 GB	24	512 MB
Disk Controllers	32	Fibre Channel Host Bus Adapter for WNT/W2K		
	32	Modular SAN Array 1000		
	48	StorageWorks Enclosure Model 4314R		
	4	SAN Switch 2/16		
Disk Drives	1120	18.2-GB 15K	24	18.2-GB 10K
	16	18.2-GB 10K		
Total Storage		20675 GB		
Tape Drives	1	12/24-Gigabyte DAT		

	HP ProLiant DL580 - O2000 32P wih 24 ProLiant DL360R	TPC-C Rev. 5.0	
		Report Date:	6-Sep-02

Description	Part Number	Third Party Brand	Pricing	Unit Price	Qty	Extended Price	3 yr. Maint. Price
Server Hardware							
ProLiant DL580R X900 2MB 1024	155618-003		1	16,129	8	129,032	
PIII X900-2MB Processor Option Kit for ML570/DL580	222627-B21		1	6,199	16	99,184	
4G SDRAM (4x1GB)	189083-B21		1	3,999	16	63,984	
NC3134 64 PCI Dual 10/100 All option kit	138603-B21		1	285	8	2,280	
Modular SAN Array 1000	201723-B21		1	9,995	32	319,840	
StorageWorks Enclosure Model 4314R - Rack-mountable	190209-001		1	2,955	48	141,840	
SAN Switch 2/16	287055-B21		1	30,499	4	121,996	
Fibre Channel Host Bus Adapter for WNT/W2K	245299-B21		1	1,792	32	57,344	
V570 Color Monitor - 15 inch CRT - Opal	228113-001		1	169	8	1,352	
12/24-Gigabyte DAT Drive (Internal)	295513-B22		1	682	1	682	
R1500	242704-001		1	880	4	3,520	
18.2-GB Pluggable 1" Universal WideUltra3 15K HDD	188122-B22		1	477	1120	534,240	
18.2-GB Pluggable 1" Universal WideUltra3 15K HDD (10% spares)	188122-B22		1	477	112		53,424
18.2GB Pluggable Ultra3 SCSI 10K 1" Universal HDD	142673-B22		1	319	16	5,104	
HP Rack Model 9142 (42U - Opal) - Flat Pallet	120663-B21		1	1,352	8	10,816	
HP Rack Sidewall Kit	120670-B21		1	212	1	212	
HP Rack Coupling Kit	120669-B21		1	85	7	595	
HP Enhanced Keyboard	122660-006		1	44	8	352	
HP Mouse	170299-B21		1	23	8	184	
FM-M1724-36YR 24x7 4HR 500 Srs Svr/EC	401782-002		1	1,795	8		14,360
FM-FC724-36 3YR 24X7 4HR RA4x/MSA1x/SA	402164-002		1	1,950	32		62,400
FM-4E724-36 3YR 24X7 4HR EMPTY DISK ENCL	171242-002		1	157	48		7,536
FM-FS724-36 3YR 24x7 4HR, FC/SAN SWITCH 16P	161308-002		1	4,538	4		18,152
LINKSYS EZXS88R ETHERFAST 8port 10/100 RACKMOUNT	101532	Linksys	5	120	3	360	
					Subtotal	**1,492,917**	**155,872**
Server Software							
Oracle9i DB Ent. Edi. Rel. 2 for Windows, 3Year 32P Unlimited Users	Run time	Oracle	3	640,000	1	640,000	
Real Application Cluster, 3 Year 32P, Ulimited Users	Run time	Oracle	3	320,000	1	320,000	
Partitioning, 3Year 32P, Unlimited Users	Run time	Oracle	3	160,000	1	160,000	
Oracle Database Server Support Package for 3 years	Run time	Oracle	3	48,000	1		48,000
Microsoft Windows 2000 Advanced Server	C10-00475	Microsoft	2	2,399	8	19,192	$5,850
Microsoft Visual C++ 6.0	048-00317	Microsoft	2	549	1	549	Incl Above
					Subtotal	**1,139,741**	**53,850**
Client Hardware							
ProLiant DL360R01 P1000-256K, 128MB 3-PACK	227214-001		1	6,237	8	49,896	
128 Reg 133MHz SDRAM DIMM	128277-B21		1	149	72	10,728	
Pentium III 1000MHz Processor (DL360)	210642-B21		1	799	24	19,176	
V570 Color Monitor - 15 inch CRT - Opal	228113-001		1	169	24	4,056	
HP Enhanced Keyboard	122660-006		1	44	24	1,056	
HP Mouse	170299-B21		1	23	24	552	
18.2GB Pluggable Ultra3 SCSI 10K 1" Universal HDD	142673-B22		1	319	24	7,656	
FM-L0724-363YR 24x7 4HR 300 SERIES SVR	162657-002		1	1,450	24		34,800
					Subtotal	**93,120**	**34,800**
Client Software							
BEA Tuxedo 6.5 Teir 1		BEA	4	3,000	24	72,000	45,360
Microsoft Windows 2000 Server	C11-00821	Microsoft	2	738	24	17,712	Incl. Above
					Subtotal	**89,712**	**45,360**
User Connectivity							
3Com® SuperStack® 3 Switch 3300 24-Port	3C16980A	3Com	6	999	4	3,996	
					Subtotal	**3,996**	**0**
Oracle Mandatory E-Business Discount (License & Support)		Oracle	3			**($292,000)**	**0**
Large Purchase and Cash discount (See Note 1)	16.0%		1			**($253,766)**	**($30,508)**
					Total	**$2,273,720**	**$259,374**

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:	**$2,533,095**
tpmC Rating:	**137,260.89**
$ / tpmC:	**$18.46**

Pricing: 1=HP Direct 2= Microsoft 3=Oracle (contact: MaryBeth Pierantoni @ 650-506-2118, See Appendeix G of FDR) 4=BEA 5= ecost.com 6=ecost.com

Note 1 = Discount based on HP Direct guidence and large cash purchase level.

Note:The benchmark results and test methodology were audited by Lorna Livingtree of Performance Metrics, Inc.

Numerical Quantities Summary

MQTH, Computed Maximum Qualified Throughput 137260.892 tpmC

Response Times (in seconds)	Average	90%	Maximum
New-Order	1.273	1.888	40.511
Payment	1.195	1.781	5.919
Order-Status	1.192	1.769	5.255
Delivery (interactive portion)	0.226	0.538	3.827
Delivery (deferred portion)	2.866	6.720	30.563
Stock-Level	1.602	2.306	6.498
Menu	0.231	0.555	3.867

Transaction Mix, in percent of total transaction	
New-Order	44.914%
Payment	43.020%
Order-Status	4.020%
Delivery	4.026%
Stock-Level	4.020%

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	18.005/0.000	18.009/14.607	18.028/145.963
Payment	3.010/0.000	3.016/12.016	3.033/120.108
Order-Status	2.010/0.000	2.016/10.018	2.030/100.097
Delivery (interactive)	2.010/0.000	2.016/5.0024	2.031/50.165
Stock-Level	2.010/0.000	2.016/5.0015	2.032/49.839

Test Duration

Ramp-up time	55 minutes
Measurement interval	120 minutes
Transactions (all types) completed during measurement interval	36,672,642
Ramp down time	30 minutes

Checkpointing

Number of checkpoints	4
Checkpoint interval	30 minutes

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 Database 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.

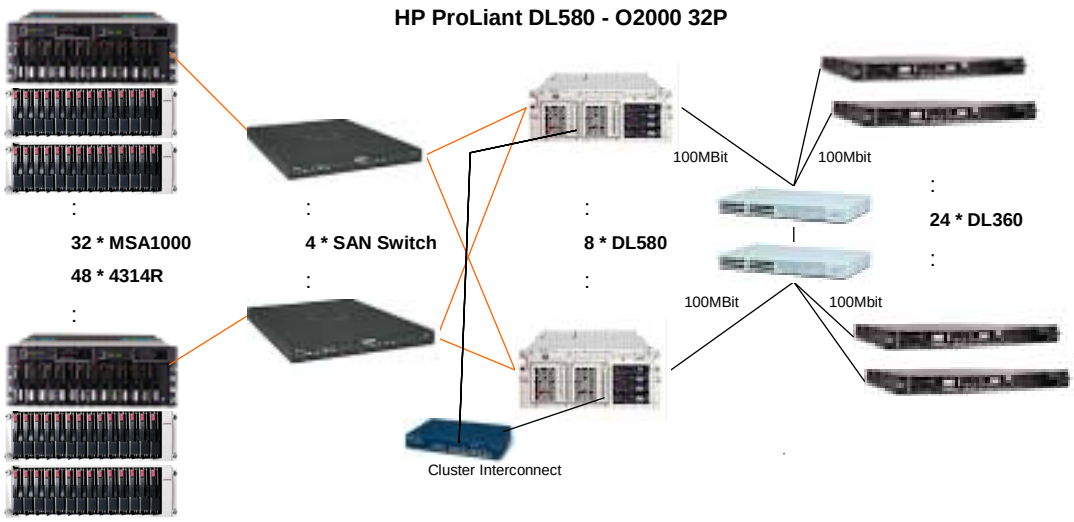
Appendix C contains the tunable parameters for the database, the operating system, and the transaction monitor.

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 on the following page

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.
1120 disks used in the benchmark have a capacity of 18.2GB 15K rpm.
16 disks used in the benchmark have a capacity of 18.2 GB 10K rpm (for OS)

Modular SAN Array 1000	SAN Switch # (HBA #)	Unformatted Capacity	Contents
1	1	254GB	Database log for Node 1
2	1	764GB	1/24 tables, indexes and rollback segs control file 1/8 system
3	1	764GB	1/24 tables, indexes and rollback segs
4	1	764GB	1/24 tables, indexes and rollback segs
5	1	254GB	Database log for Node 2
6	1	764GB	1/24 tables, indexes and rollback segs srvcfg 1/8 system
7	1	764GB	1/24 tables, indexes and rollback segs
8	1	764GB	1/24 tables, indexes and rollback segs
9	2	254GB	Database log for Node 3
10	2	764GB	1/24 tables, indexes and rollback segs 1/8 system
11	2	764GB	1/24 tables, indexes and rollback segs
12	2	764GB	1/24 tables, indexes and rollback segs
13	2	254GB	Database log for Node 4
14	2	764GB	1/24 tables, indexes and rollback segs 1/8 system
15	2	764GB	1/24 tables, indexes and rollback segs
16	2	764GB	1/24 tables, indexes and rollback segs
17	3	254GB	Database log for Node 5
18	3	764GB	1/24 tables, indexes and rollback segs 1/8 system
19	3	764GB	1/24 tables, indexes and rollback segs
20	3	764GB	1/24 tables, indexes and rollback segs
21	3	254GB	Database log for Node 6
22	3	764GB	1/24 tables, indexes and rollback segs 1/8 system
23	3	764GB	1/24 tables, indexes and rollback segs
24	3	764GB	1/24 tables, indexes and rollback segs
25	4	254GB	Database log for Node 7
26	4	764GB	1/24 tables, indexes and rollback segs 1/8 system
27	4	764GB	1/24 tables, indexes and Rollback segs
28	4	764GB	1/24 tables, indexes and rollback segs
29	4	254GB	Database log for Node 8 1/8 system
30	4	764GB	1/24 tables, indexes and rollback segs
31	4	764GB	1/24 tables, indexes and rollback segs
32	4	764GB	1/24 tables, indexes and rollback segs

Priced Configuration:

The priced configuration has additional 12/24-Gigabyte DAT drive. The twenty four client systems DL 360R 866MHz used in the test were substituted with twenty four DL 360 1000MHz in the configuration pricing, as the DL 360R 866MHz are no longer orderable. The NC6134 NICs used in the test were substituted with NC3134 NICs in the configuration pricing, as the NC6134s are no longer orderable. The 9.1GB 10K rpm disk drives used in the test (server and cient OS) were substituted with 18.2GB 10K rpm disk drives, as 9.1GB 10K rpm disk drives are no longer orderable. The SAN Switch 2/16 used in the benchmark is not available with the same part number any more, so substituted with the new part number. All other 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.

Horizontal partitioning based on warehouse id was used on the nord, ordr, ordl and hist tables.

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 ProLiant DL360R.

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	59.95%
Delivery	Skipped transactions	None
Transaction Mix	New Order	44.914%
	Payment	43.020%
	Order status	4.020%

Statistic		Value
	Delivery	4.026%
	Stock level	4.020%

Queuing Mechanism

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

BEA Tuxedo on each client system served as the queuing mechanism to the database. Each delivery request was submitted to BEA Tuxedo 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.

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

Durability from media failure was demonstrated on a database scaled for 3006 warehouses. The standard driving mechanism was used to generate the transaction load of 30060 users. The fully scaled database under full load would also have passed the following test.

Loss of Data

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

1. A partition on a disk was 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 30060 users and against two nodes.
4. The test was allowed to run for a minimum of 10 minutes.
5. The backed up partition was overwritten with garbage information.
6. ORACLE9i recorded errors about corrupt data on the partition. The database and the RTE were then shut down.
7. The database partition which was backed up in Step 1 was restored.
8. The database was then started. The database was recovered using the recover command from SQLPLUS.
9. The database was opened and ORACLE 9i performed instance recovery.
10. Consistency conditions were executed and verified.
11. Step 2 was repeated and the difference between the first and second counts was noted.
12. An RTE report was generated for the entire run time giving the number of NEW-ORDERS successfully returned to the RTE.
13. The counts in step 10 and 11 were compared and the results verified that all committed transactions had been successfully recovered.
14. Samples were taken from the RTE files and used to query the database to demonstrate successful transactions had corresponding rows in the ORDER table.

Loss of Log

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

1. The total number of New Orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving the beginning count. Consistency check 3 was verified before run.
2. The RTE was started with 30060 users and against two nodes.
3. The test was allowed to run for a minimum of 10 minutes.
4. A log disk containing log information for node 2 was removed.
5. The system continued running because the logs are mirrored.
6. The database and the RTE were then shut down.
7. The database was then started. Consistency conditions were executed and verified.
8. Step 1 was repeated and the difference between the first and second counts was noted.
9. An RTE report was generated for the entire run time giving the number of NEW-ORDERS successfully returned to the RTE.
10. The counts in step 7 and 8 were compared and the results verified that all committed transactions had been successfully recovered.
11. 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 (8 Nodes)

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 12024 warehouses under a full load of 120240 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 120240 users.
3. The test was allowed to run for a minimum of 10 minutes.
4. A checkpoint was issued.
5. Upon completion of the checkpoint a system crash and loss of memory were induced by turning all six of the computers in the cluster off. No battery backup or Uninterruptible Power Supply (UPS) were used to preserve the contents of memory.
6. The RTE was shutdown.
7. Power was restored and one of the systems restarted.
8. ORACLE 9i was restarted and performed an automatic recovery.
9. Consistency conditions were executed and verified.
10. Step 1 was repeated and the difference between the first and second counts was noted.
11. An RTE report was generated for the entire run time giving the number of NEW-ORDERS successfully returned to the RTE.
12. The counts in step 9 and 10 were compared and the results verified that all committed transactions had been successfully recovered.
13. 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 (1 Node)

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 12024 warehouses under a full load of 120240 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 120240 users.
3. The test was allowed to run for a minimum of 10 minutes.
4. A checkpoint was issued.
5. Upon completion of the checkpoint a system crash and loss of memory on a single node was induced by turning one of the nodes of the cluster off. No battery backup or Uninterruptible Power Supply (UPS) were used to preserve the contents of memory.
6. The RTE was shutdown.
7. The database was shutdown on remaining seven nodes.
8. Power was restored and restarted the system which was powered off.
9. ORACLE 9i was restarted and performed an automatic recovery.
10. Consistency conditions were executed and verified.
11. Step 1 was repeated and the difference between the first and second counts was noted.
12. An RTE report was generated for the entire run time giving the number of NEW-ORDERS successfully returned to the RTE.
13. The counts in step 10 and 11 were compared and the results verified that all committed transactions had been successfully recovered.
14. Samples were taken from the RTE files and used to query the database to demonstrate successful transactions had corresponding rows in the ORDER table.

Loss of Cluster interconnect

To demonstrate recovery from a permanent failure of an inter-node connection, the following steps were executed on a fully scaled database of 12024 warehouses under a full load of 120240 users.

1. The D_NEXT_O_ID fields for all rows in the District table were summed to determine a starting count of Orders in the database prior to the test.
2. The RTE was started with 120240 users.
3. After 10 minutes into the measurement period, the cluster interconnect switch was powered off.
4. The system detected the interconnect loss. The test was aborted on the RTEs.
5. All Oracle instances were shutdown.
6. ORACLE 9i was restarted and performed an automatic recovery.
7. Consistency conditions were executed and verified.
8. Step 1 was repeated and the difference between the first and second counts was noted.
9. An RTE report was generated for the entire run time giving the number of NEW-ORDERS successfully returned to the RTE.
10. The counts in step 8 and 9 were compared and the results verified that all committed transactions had been successfully recovered.
11. 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	12024
District	120240
Customer	360720000
History	360720000
Order	360720000
New Order	108216000
Order Line	3607291272
Stock	1202400000
Item	100000
Deleted Warehouse	0

Database Layout

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

The benchmarked configuration used four 16 port SAN switch 2/16. Each switch was connected to the eight nodes of the cluster and eight Modular SAN Array 1000. Total of 32 Modular SAN Arrays in the cluster. Each Modular SAN Array 1000 contained 14 disk drives. Eight Modular SAN Arrays had two RAID 1 partitions and were used for database log. Array accelerator was disabled for the log volumes. 24 of the Modular SAN Array had two additional StorageWorks Enclosure 4314R connected to it. Each of the StorageWorks Enclosure 4314R contained 14 disk drives. Each of these 24 Modular SAN Array had one RAID 0 volume (database) and one RAID 1 Volume (database backup). Array accelerator was set to 100% write cache for data volumes. All the disk drives were 18.2GB/15K rpm.

In addition, each of the system had two internal disk drives (RAID 1) for operating system.

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 9i Enterprise Edition Release 2 with Real Application Clusters and Partitioning Options 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 database tables Orders, New_Order, Order_Line, and History were partitioned horizontally by Warehouse ID. Each of these tables had 24 partitions.

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.

Segments:	Blocks used	Blocks Allocated
CUSTCLUSTER	90,597,505	95,490,048
DISTCLUSTER	125,255	299,425
HIST	7,999,488	49,139,712
ICUST1	2,167,296	3,065,856
ICUST2	4,504,576	6,137,856
IDIST	8,192	30,720
IITEM	3,840	12,800
INORD	1,093,632	2,457,600
IORDL	73,764,864	98,365,440
IORDR1	3,674,112	9,203,712
IORDR2	5,517,312	6,144,000
ISTOK	6,387,200	15,353,856
ITEMCLUSTER	3,840	4,096
IWARE	2,048	30,720
ORDR	5,799,936	49,139,712
STOKCLUSTER	120,741,005	128,040,448
SYSTEM	196,608	196,608
WARECLUSTER	12,529	307,200
	322,599,238	463,419,809
	BTB	
Free space	140,820,571	
Dynamic space	87,564,288	
Static Space	235,034,950	
Daily growth	16312708.34	
Daily spread	116351508.5	
60 days	1,213,797,450.60	Blocks
Block size	4,096.00	

60 day space	4,971,714,357,652.70	Blocks
	4,855,189,802.40	KB
	4,741,396.29	MB
	4,630.27	GB
Data Disks	1008	
Capacity	16.96	GB
Total	17095.68	GB
Log/NO_txn	12.51161408	KB
tpmC	137260.892	
8hr Log	824330548.3	KB
	805010.301	MB
	786.1428721	GB
	1572.285744	RAID 1
Log Disks	112	
Capacity	16.96	GB
Total	1899.52	GB

Clause 5 Related Items

Throughput

Measured tpmC must be reported

Measured tpmC	137,260.89 tpmC
Price per tpmC	$18.46 per tpmC

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	1.273	40.511	1.888
Payment	1.195	5.919	1.781
Order-Status	1.192	5.255	1.769
Interactive Delivery	0.226	3.827	0.538
Deferred Delivery	2.866	30.563	6.720
Stock-Level	1.602	6.498	2.306
Menu	0.231	3.867	0.555

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

Type	Minimum	Average	Maximum
New-Order	18.005	18.009	18.028
Payment	3.010	3.016	3.033
Order-Status	2.010	2.016	2.030
Interactive Delivery	2.010	2.016	2.031
Stock-Level	2.010	2.016	2.032

Table 5.3: Think Times

Type	Minimum	Average	Maximum
New-Order	0.000	14.607	145.963
Payment	0.000	12.016	120.108
Order-Status	0.000	10.018	100.097
Interactive Delivery	0.000	5.0024	50.165
Stock-Level	0.000	5.0015	49.839

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

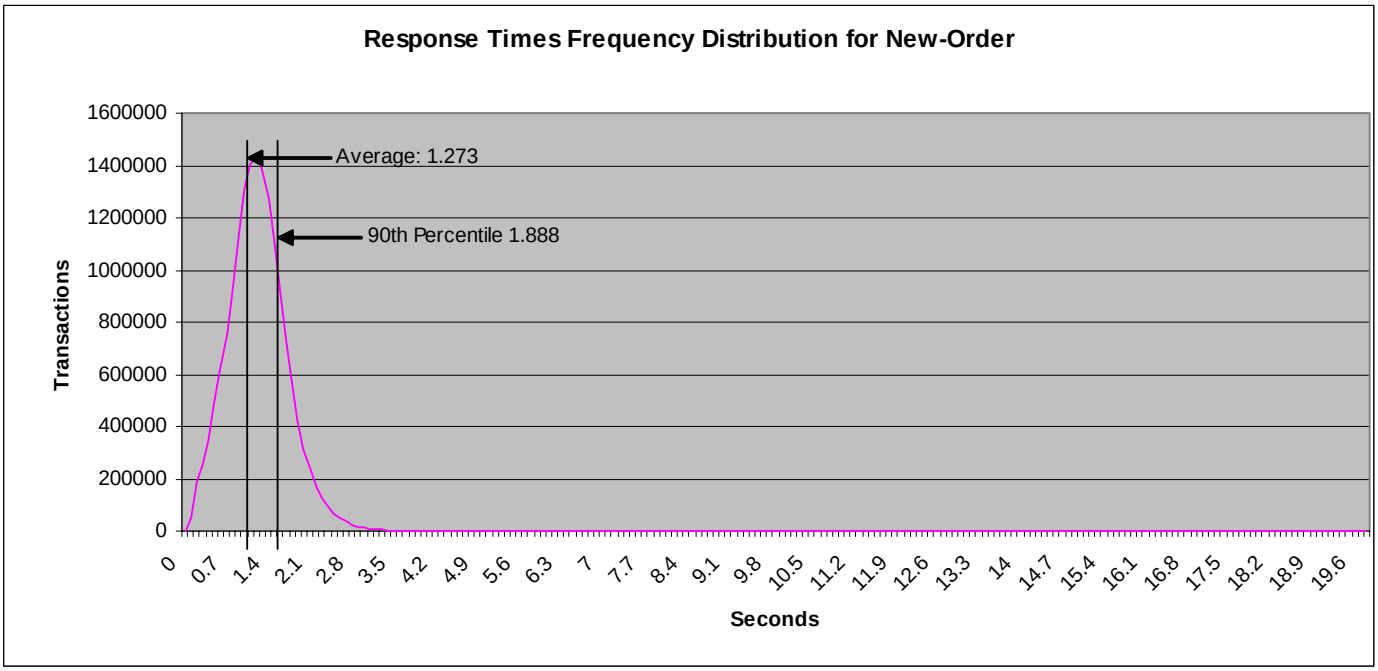


Figure 5.2: Response Times Frequency Distribution for Payment Transactions

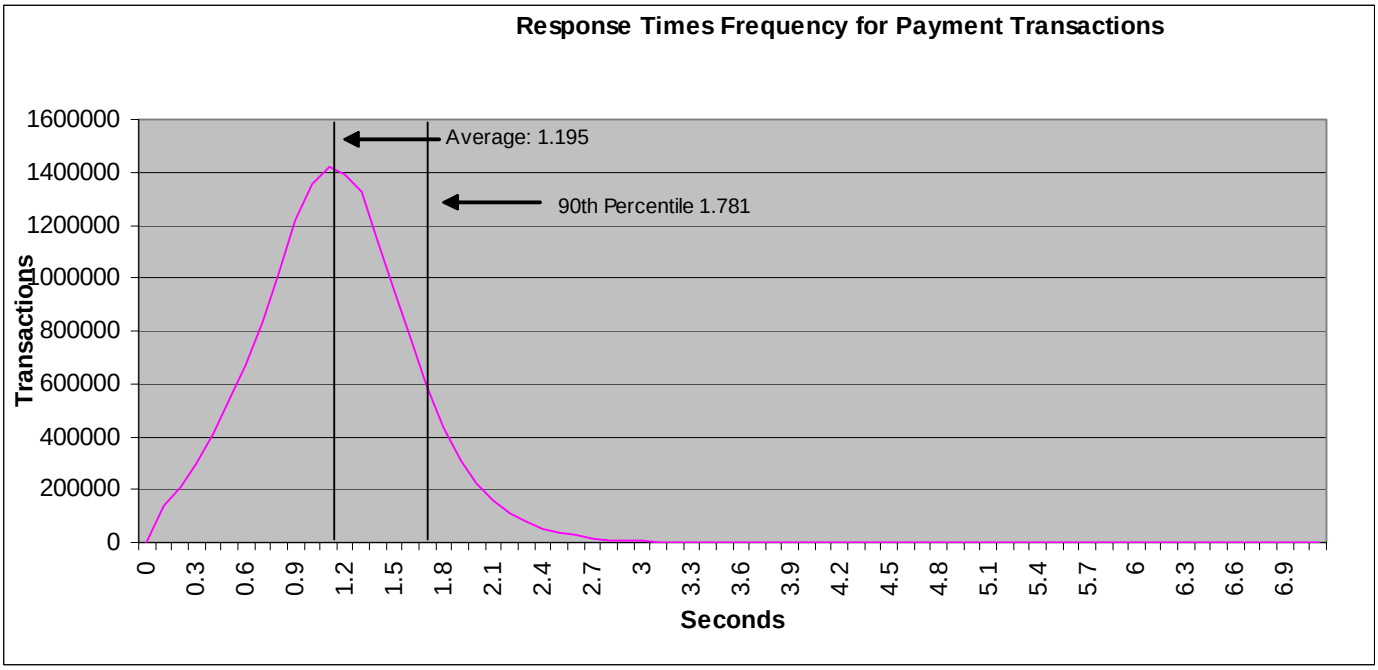


Figure 5.3: Response Times Frequency Distribution for Order Status Transactions

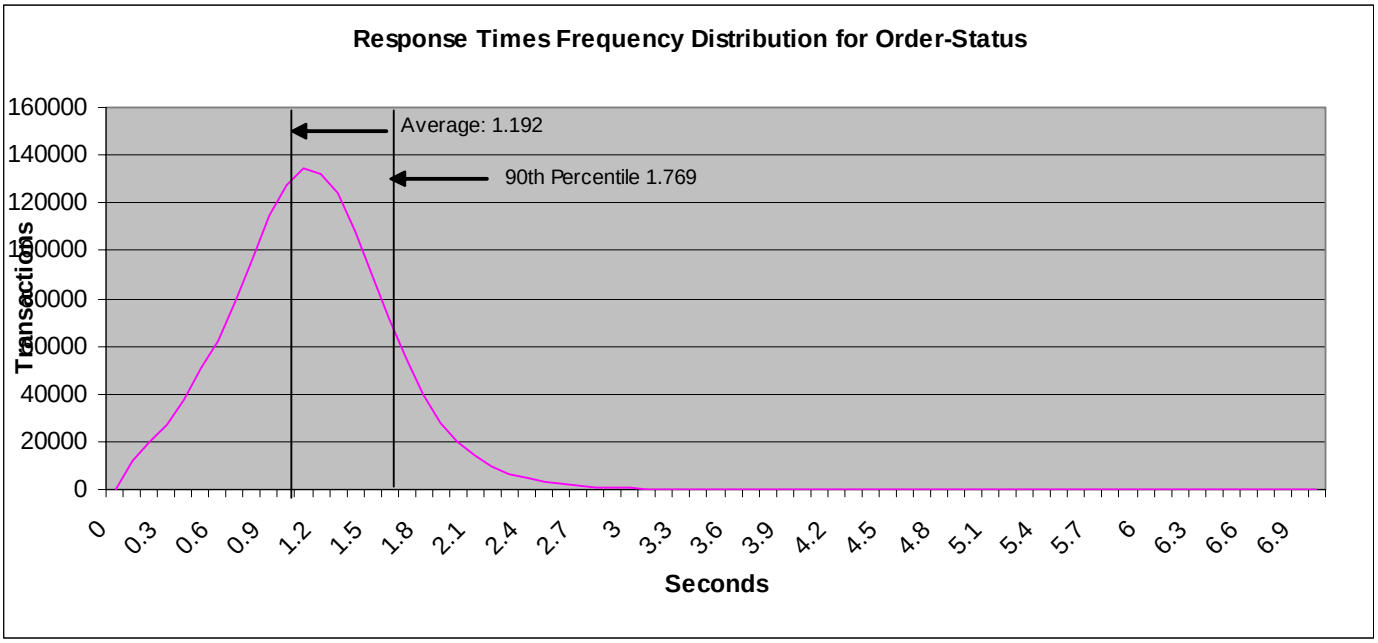


Figure 5.4: Response Times Frequency Distribution for Delivery Transactions

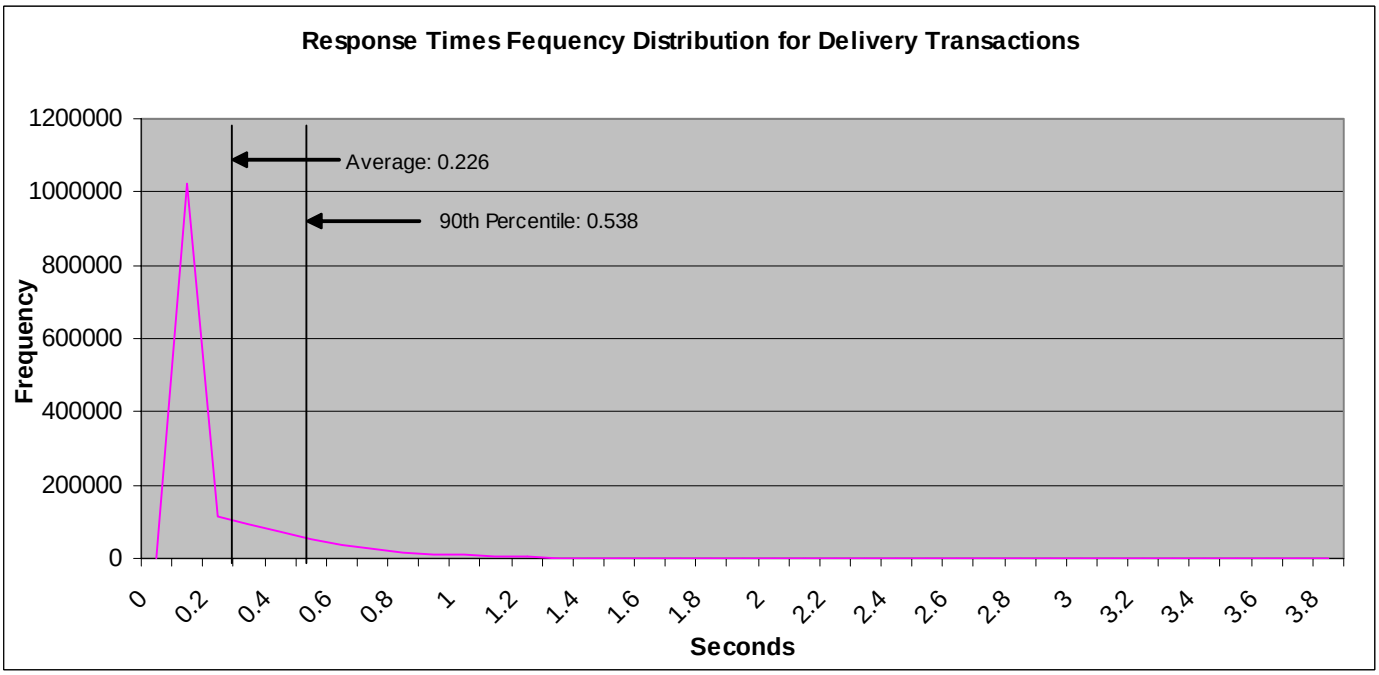


Figure 5.5: Response Times Frequency Distribution for Stock Level Transactions

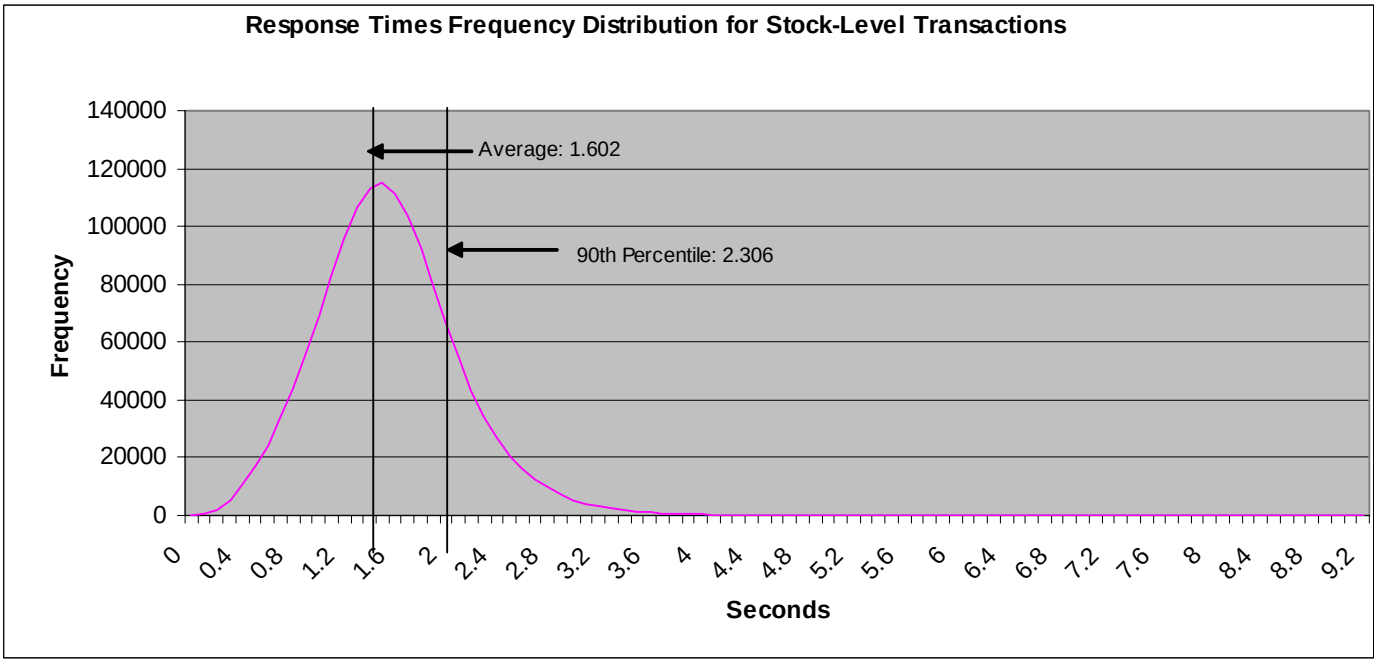


Figure 5.6: Response Time versus Throughput

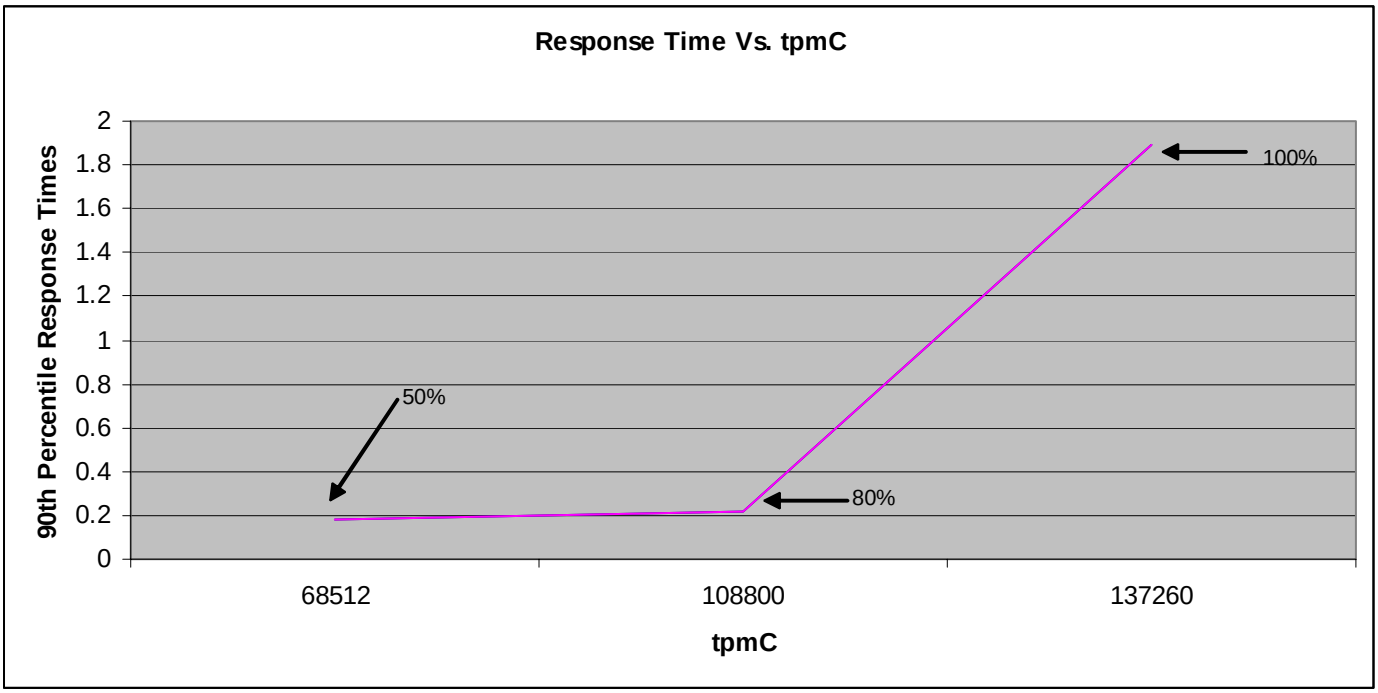


Figure 5.7: Think Times distribution for New Order Transactions

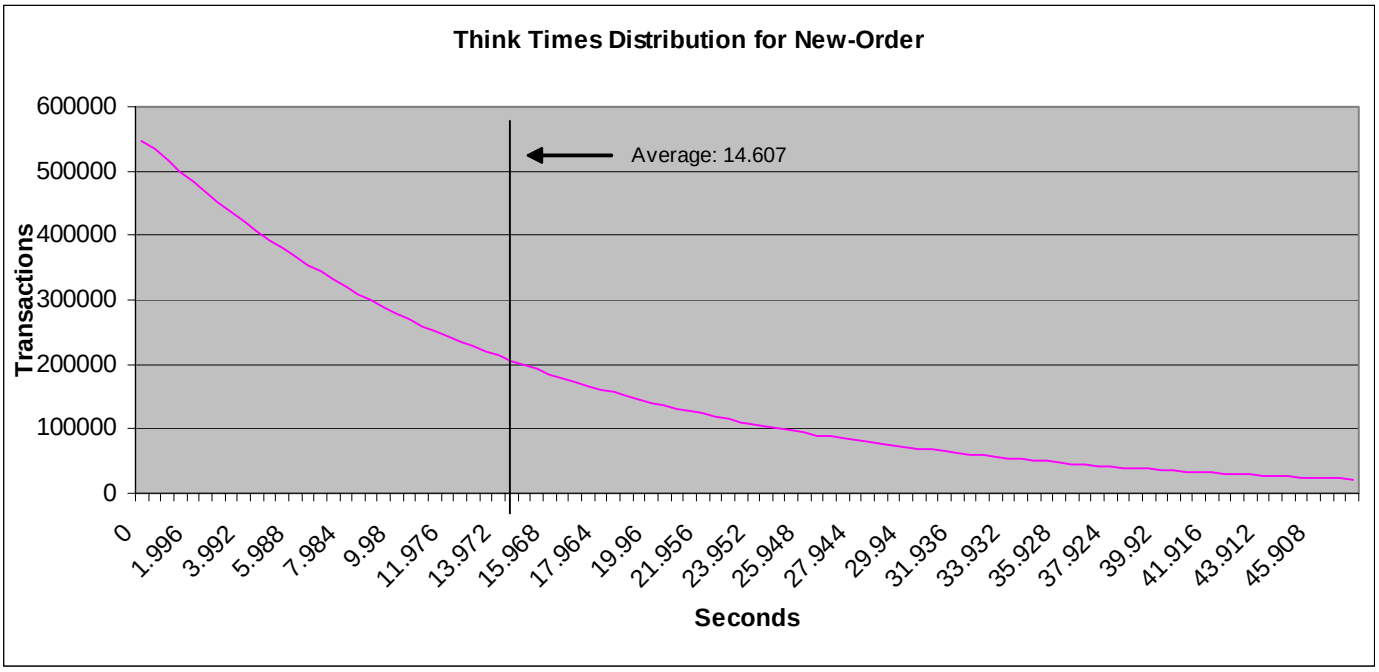
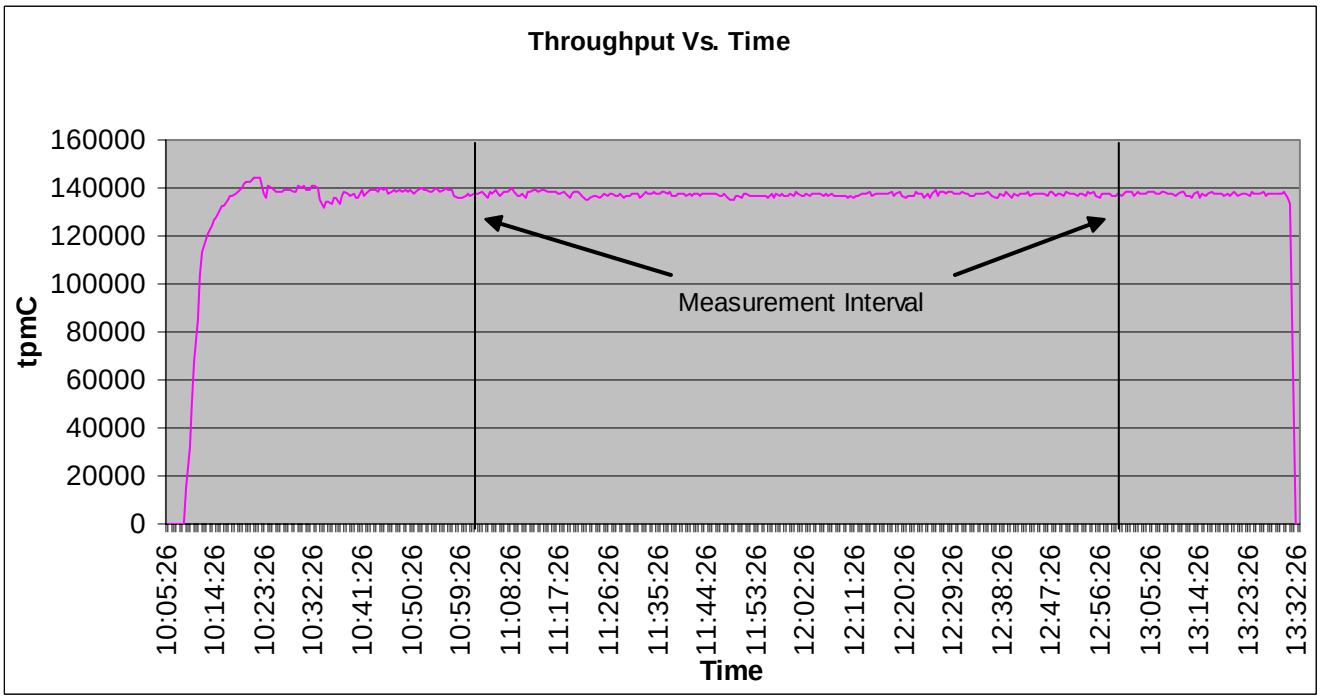


Figure 5.8: Throughput versus Time



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 the throughput data collected during the run and 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 Internet Information Server.

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 exactly 120 minutes long.

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 5.4: 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	59.95%
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.914%
	Payment	43.020%
	Order status	4.020%
	Delivery	4.026%
	Stock level	4.020%

Checkpoint Count and Location

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.

A checkpoint is the process of writing all modified data pages to disk. The TPC-C benchmark on HP ProLiant DL580 - O2000 32P was set up to checkpoint within every 30 minutes. One checkpoint occurred during the warm-up period and 4 checkpoints occurred during the measurement period.

Checkpoint Duration

The start time and duration in seconds of at least the four longest checkpoints during the measurement Interval must be disclosed.

Node	Start Time	End Time	Duration
1	11:02:20	11:25:27	0:23:07
2	11:02:36	11:25:40	0:23:04
3	11:03:14	11:26:40	0:23:26
4	11:03:23	11:26:54	0:23:31
5	11:03:23	11:27:06	0:23:43

6	11:03:27	11:27:17	0:23:50
7	11:03:28	11:27:14	0:23:46
8	11:03:33	11:02:20	0:23:46
1	11:27:58	11:50:58	0:23:00
2	11:28:10	11:51:09	0:22:59
3	11:29:14	11:52:36	0:23:22
4	11:29:28	11:52:52	0:23:24
5	11:29:44	11:53:29	0:23:45
6	11:29:49	11:53:26	0:23:37
7	11:29:52	11:53:29	0:23:37
8	11:29:53	11:53:36	0:23:43
1	11:53:29	12:16:46	0:23:17
2	11:53:39	12:16:45	0:23:06
3	11:55:07	12:18:35	0:23:28
4	11:55:24	12:18:54	0:23:30
5	11:56:00	12:19:48	0:23:48
6	11:56:05	12:19:51	0:23:46
7	11:56:06	12:19:57	0:23:51
8	11:56:10	12:19:40	0:23:30
1	12:19:16	12:42:27	0:23:11
2	12:19:18	12:42:33	0:23:15
3	12:21:08	12:44:38	0:23:30
4	12:21:29	12:45:04	0:23:35
5	12:22:17	12:46:03	0:23:46
6	12:22:26	12:46:10	0:23:44
7	12:22:27	12:46:19	0:23:52
8	12:22:32	12:46:10	0:23:38

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 8 ProLiant servers.

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. In the tested configuration, eight server systems and 24 client systems were connected into two cascaded 24 port 10/100 Ethernet switches. In the tested configuration there were eight driver systems (RTE), each of them connected to three clients systems using 10/100 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** **137,260.89 tpmC**
- **Price per tpmC** **$18.46 per tpmC**
- **Available** **September 6, 2002**

All hardware 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 9i Enterprise Edition Release 2 with Real Application Clusters and Partitioning Options
- 8 Microsoft Windows 2000 Advanced Server
- 24 Microsoft Windows 2000 Server
- 24 BEA Tuxedo CTS 6.5
- 1 Microsoft Visual C++

See Appendix G Oracle pricing.

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.

Lorna Livingtree
Performance Metrics Inc.
2229 Benita Dr. Suite 101
Rancho Cordova, CA 95670
916-635-2822

Availability of the Full Disclosure Report

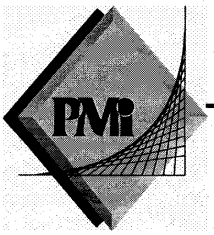
The Full Disclosure Report must be readily available to the public at a reasonable charge, similar to the charges for similar documents by the test sponsor. The report must be made available when results are made public. In order to use the phrase "TPC Benchmark™ C", the Full Disclosure Report must have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

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

Transaction Processing Performance Council
Presidio of San Francisco
Building 572B (surface)
P.O. Box 29920 (mail) San Francisco, CA 94129-0920
Voice: 415-561-6272
Fax: 415-561-6120
Email: info@tpc.org

or

Hewlett Packard Company
Database Performance Engineering
P.O. Box 692000
Houston, TX 77269-2000



PERFORMANCE METRICS INC.
TPC Certified Auditors

August 28, 2002

Mr. Raghunath Othayoth
Database Performance Engineer
Compaq Computer Corporation
20555 SH 249
Houston, TX 77070

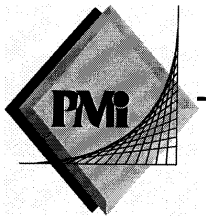
I have verified the new pricing for the HP Parallel Database Cluster configuration published in June 2002. This pricing has substituted disk drives, SAN switches and network cards that are no longer orderable from the manufacturer with components of equal or greater performance and capacity. All requirements for substitution were verified to be compliant. All other items on the price sheet remain the same, and some have had a price change compliant with the repricing rules.

Sincerely,

Lorna Livingtree

Lorna Livingtree
Auditor

2229 Benita Dr. Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 fax: (916) 858-0109 email: Lorna@PerfMetrics.com



PERFORMANCE METRICS INC.
TPC Certified Auditors

June 4, 2002

Mr. Raghunath Othayoth
Database Performance Engineer
Compaq Computer Corporation
20555 SH 249
Houston, TX 77070

I have verified on-site and by remote the TPC Benchmark™ C client/server for an 8 node cluster with the following configuration on each node:

Platform: HP Parallel Database Cluster Model 02000
Database Manager: Oracle9i
Operating System: Microsoft Windows 2000 Advanced Server
Transaction Monitor: BEA Tuxedo 6.5

Servers: 8 ProLiant DL580's with:				
CPU's	Memory	Disks (total)	90% Response	TpmC
4 Pentium III Xeon@900Mhz	Main: 8 GB Cache: 2 MB	1120 @ 18GB 16 @ 9.1 GB	1.88 sec	137,260.89
24 Clients: DL360R each with:				
2 Pentium III @866Mhz	Main: 512 MB Cache: 256K	1 @ 9.1GB	Na	Na

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

- The transactions were correctly implemented.
- The database files were properly sized and populated.
- The database was properly scaled with 12,024 warehouses. All warehouses were active during the performance run.
- All columns in the database could be updated with the same syntax.

2229 Benita Dr. Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 fax: (916) 858-0109 email: Lorna@PerfMetrics.com

- The ACID properties were successfully demonstrated. Both a single node and a full system failure was 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 day space calculation was verified.
- The controller cache was disabled on the log disk controllers.
- The steady state portion of the test was 120 minutes.
- One checkpoint was taken before the measured interval.
- Four checkpoints were taken during the measured interval.
- The system pricing was checked for major components and maintenance.
- Third party quotes were verified for compliance.

Auditor Notes:
At the conclusion of the benchmark, the tested client machines were no longer orderable from the sponsor with 866Mhz processors. Substitutes with 1Ghz processors were priced and all requirements for substitution were met and verified.

Sincerely,

Lorna Livingtree

Lorna Livingtree
Auditor

2229 Benita Dr. Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 fax: (916) 858-0109 email: Lorna@PerfMetrics.com

Appendix A: Source Code

```
-------------------------
makefile
-------------------------
#####################################################################
############
#
#
#                               FFE MAKEFILE
#
#
#
#####################################################################
############
#

version = v6-1-3L

## FWM 4-25-97
## The ACMSxp STDL compiler creates a TEMP directory (sub-dir below
current working dir)
## It also creates .h  files in the current working directory.
## Hopefully this will change when we get the final shipping
version of the compiler.
## This is an early FT version of the compiler.
##

#CEXTRA are parameters added on the make command line
CEXTRA=

#UNIX DEBUG MODE
#CDEBUG = -g2 -O0
#STRIP = ls -l
#ELSE NOT UNIX DEBUG MODE
#CDEBUG = -O5 -migrate -non_shared
CDEBUG = -g2 -migrate -non_shared
STRIP = strip
#ENDIF

CDEFS     = -DUSE_PROCESSES -DDEBUG
CFLAGS = -I. -IC:\oracle\ora90\oci\include -I..\..\prte\src -
DVERSION='"$(version)"'

# UPper case definitions are for UNIX
CC = cc $(CDEFS) $(CDEBUG) $(CFLAGS) $(CEXTRA)

crtl = MT
calling_convention = Gd

#NT DEBUG MODE
#cdebug = -Z7 -Od -$(crtl)d -D_DEBUG -D_CRTDBG_MAP_ALLOC -
DFFE_DEBUG
#ldebug = /debug:full /pdb:none /map
#ELSE NOT NT DEBUG MODE
cdebug = -Ox -$(crtl) -DNDEBUG
ldebug = /RELEASE
#ENDIF

cdefs     = -DWIN32 -D_WINDOWS -DUSE_FIBERS -D_WIN32_WINNT=0x0400 -
DWIN32_LEAN_AND_MEAN -D"VERSION=\"$(version)\""
cflags = /nologo /W3 /GX /Zi /I . /I C:\oracle\ora90\OCI\INCLUDE /I
..\..\prte\src /$(calling_convention)

EXPORTS = -export:HttpExtensionProc -export:GetExtensionVersion

# lowER case definitions are for WNT
cc    = cl $(cdefs) $(cdebug) $(cflags) $(CEXTRA)

TPCCHFILES =        ./tpcc.h \
                    ./web_ui.h \
                    ./tpccstruct.h \
                    ./tpccapi.h \
                    ./tpccerr.h


WNT_LIBS =          wsock32.lib \
                    kernel32.lib \
                    advapi32.lib \
                    user32.lib \
                    pdh.lib \
                    gdi32.lib \
                    comdlg32.lib \
                    winspool.lib


TUX_INC =       $(TUXDIR)\include

DBLIBINC =          $(MSSQL)/devtools/INCLUDE
DBLIBDIR =          $(MSSQL)/devtools/LIB

SYBASEINC =         $(SYBASE)\include
SYBASELIB =         $(SYBASE)\lib

# 6/29/1999 Modification made by Susan Georgson.
# Added ORACLE value to makefile because oci.h couldn't be found.
ORACLE =        c:\oracle\ora90
ORALIB =  $(ORACLE)\oci\LIB\MSVC
ORAINC =  $(ORACLE)\oci\INCLUDE
!IF "$(PROCESSOR_ARCHITECTURE)" == "x86"
OCILIB =  $(ORALIB)\oci.lib
!ENDIF
!IF "$(PROCESSOR_ARCHITECTURE)" == "ALPHA"
OCILIB =  $(ORALIB)\ora803.lib
!ENDIF

ACMSXP_LIBS =       $(STDL_LIB_DIR)\stdl_rtm.lib

# These h files are generated by acmsxp
acmsxphs =          temp\tpcc_acmsxp_pp.h

acmsxpcs =          temp\tpcc_dy_c_c.c \
                    temp\tpcc_no_c_c.c \
                    temp\tpcc_os_c_c.c \
                    temp\tpcc_pt_c_c.c \
                    temp\tpcc_sl_c_c.c \
                    temp\tpcc_gc_c_c.c

acmsxpobjs =        ..\obj\tpcc_fct.obj \
                    ..\obj\tpcc_ps.obj \
                    ..\obj\tpcc_dy_c_c.obj \
                    ..\obj\tpcc_no_c_c.obj \
                    ..\obj\tpcc_os_c_c.obj \
                    ..\obj\tpcc_pt_c_c.obj \
                    ..\obj\tpcc_sl_c_c.obj \
                    ..\obj\tpcc_gc_c_c.obj

acmsxpflags =

acmsxptempdir =     .\temp

webuiobjs =         ..\obj\web_ui.obj \
                    ..\obj\tpcc.obj \
                    ..\obj\logfile.obj

deliveryobjs =      ..\obj\deli_cli.obj \
                    ..\obj\deli_srv.obj \
                    ..\obj\buf.obj

nullobjs =          ..\obj\null_cli.obj \
                    ..\obj\null_srv.obj

ckptobjs =          ..\obj\ckpt.obj

tuxsrvobjs =        ..\obj\tux_srv.obj \
                    ..\obj\tux_cli.obj

TUX_LIBS =          $(TUXDIR)\lib\libtux.lib \
                    $(TUXDIR)\lib\libbuft.lib \
                    $(TUXDIR)\lib\libtux2.lib \
                    $(TUXDIR)\lib\libfml.lib \
                    $(TUXDIR)\lib\libfml32.lib \
                    $(TUXDIR)\lib\libgp.lib

#####################################################################
############
#
#
#       Generic tags for make
#
#
#
#####################################################################
############

all: \
          all_acmsxp \
          all_admin \
          all_loop \
#         all_msdblib \
#         all_syb \
#         all_ora \
          all_ora8

all_acmsxp: \
          $(acmsxphs) \
          $(acmsxpcs) \
          $(acmsxptempdir)\tpcc_stdl_workspaces.stdl \
          $(acmsxptempdir)\tpcc_stdl_workspaces.h

all_admin: \
          admin \
          webd

all_loop: \
          webloop \
          webacmsxploop
#         webmuxloop

all_msdblib: \
          webmsdblib \
          webacmsxp
#         webmuxmsdblib

all_syb: \
          websyb \
          webacmsxpsyb

all_ora: \
          jacketora \
          webora \
```

```
          webacmsxpora

all_ora8: \
          jacketora8 \
          webora8 \
          webacmsxpora8

all_ora_tux: \
          tuxora9 \
          webtux

admin:              ..\exe\admin.dll
webd:               ..\exe\webd.exe

webloop:  ..\exe\webloop.dll
webacmsxploop:      ..\exe\webacmsxploop.dll
#webmuxloop:        ..\exe\webmuxloop.dll

webmsdblib:         ..\exe\webmsdblib.dll
webacmsxp:          ..\exe\webacmsxp.dll
#webmuxmsdblib:     ..\exe\webmuxmsdblib.dll

websyb:             ..\exe\websyb.dll
webacmsxpsyb:       ..\exe\webacmsxpsyb.dll

jacketora:          ..\exe\jacketora.exe
webora:             ..\exe\webora.dll
webacmsxpora:       ..\exe\webacmsxpora.dll

jacketora8:         ..\exe\jacketora8.exe
webora8:  ..\exe\webora8.dll
webacmsxpora8:      ..\exe\webacmsxpora8.dll

tuxora9:  ..\exe\tuxora9.exe
webtux:             ..\exe\webtux.dll


######################################################################
############
#
#
#       Executables
#
#
#
######################################################################
############

..\exe\tuxloop.exe: \
          .\tuxserver.c .\tuxgen.c \
          .\dbloop.c
    $(cc) -DTPMONITOR -I%TUXDIR%\include \
          .\tuxserver.c .\tuxgen.c \
          .\dbloop.c
          $(TUX_LIBS) \
          $(WNT_LIBS) \
          -Fe..\exe\tuxloop.exe \
          -link $(ldebug) -implib:tuxgen.lib -subsystem:console

..\exe\tuxmsdblib.exe: \
          .\tuxserver.c .\tuxgen.c \
          .\tpccdb.c
    $(cc) -DTPMONITOR -I%TUXDIR%\include \
          .\tuxserver.c .\tuxgen.c \
          .\tpccdb.c \
          $(TUX_LIBS) \
          $(DBLIBDIR)\ntwdblib.lib \
          $(WNT_LIBS) \
          -Fe..\exe\tuxmsdblib.exe \
          -link $(ldebug) -implib:tuxgen.lib -subsystem:console

..\exe\jacketloop.exe: \
          ..\obj\jacket.obj \
          ..\obj\dbloop.obj
    $(cc) \
          ..\obj\jacket.obj \
          ..\obj\dbloop.obj \
          -Fe..\exe\jacketlooop.exe \
          -link $(ldebug)     -subsystem:console

..\exe\jacketora.exe: \
          ..\obj\jacket.obj \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db.obj ..\obj\oracle_txns.obj
    $(cc) \
          ..\obj\jacket.obj \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db.obj ..\obj\oracle_txns.obj \
          $(ORALIB)\ociw32.lib \
          $(WNT_LIBS) \
          -Fe..\exe\jacketora.exe \
          -link $(ldebug) -subsystem:console

..\exe\jacketora8.exe: \
          ..\obj\jacket.obj \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db8.obj ..\obj\oracle_txns8.obj
    $(cc) \
          ..\obj\jacket.obj \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db8.obj ..\obj\oracle_txns8.obj \
          $(OCILIB) \
          $(WNT_LIBS) \
          -Fe..\exe\jacketora8.exe \
          -link $(ldebug) -subsystem:console

..\exe\dbstub.exe: \
          ..\obj\dbstub.obj \
          ..\obj\dbloop.obj
    $(cc) \
          ..\obj\dbstub.obj \
          ..\obj\dbloop.obj \
          -Fe..\exe\dbstub.exe \
          -link $(ldebug) -subsystem:console

..\exe\webd.exe: \
          ..\obj\webd.obj \
          $(WEB_DLL)
    $(cc) \
          ..\obj\webd.obj \
          $(WEB_DLL) \
          $(WNT_LIBS) \
          -Fe..\exe\webd.exe \
          -link $(ldebug) -subsystem:console

..\exe\webstub.exe: \
          ..\obj\webstub.obj
    $(cc) \
          ..\obj\webstub.obj \
          ..\obj\webdbg.lib \
          -Fe..\exe\webstub.exe
          -link $(ldebug) -subsystem:console

dbtest.exe: \
          .\dbtest.c
    $(cc) -I$(DBLIBINC) \
          .\dbtest.c \
          $(DBLIBDIR)\ntwdblib.lib \
          -Fe..\obj\dbtest.exe
          -link $(ldebug) -subsystem:console

..\exe\webdbg.dll: \
          ..\obj\webdbg.obj
    $(cc) \
          ..\obj\webdbg.obj \
          $(WNT_LIBS) \
          -Fe..\exe\webdbg.dll
          -link $(ldebug) -dll $(EXPORTS)

..\exe\webloop.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\dbloop.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\dbloop.obj \
          $(WNT_LIBS) \
          -Fe..\exe\webloop.dll \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\webmsdblib.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\tpccdb.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\tpccdb.obj \
          $(DBLIBDIR)\NTWDBLIB.LIB \
          -Fe..\exe\webmsdblib.dll \
          $(WNT_LIBS) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\websyb.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\sybase_db.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\sybase_db.obj \
          $(SYBASELIB)\libsybdb.lib \
          -Fe..\exe\websyb.dll \
          $(WNT_LIBS) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\webora.dll: \
          $(webuiobjs) \
```

```
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db.obj ..\obj\oracle_txns.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db.obj ..\obj\oracle_txns.obj \
          -Fe..\exe\webora.dll \
          $(WNT_LIBS) $(ORALIB)\ociw32.lib \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\webora8.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db8.obj ..\obj\oracle_txns8.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(nullobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db8.obj ..\obj\oracle_txns8.obj \
          -Fe..\exe\webora8.dll \
          $(WNT_LIBS) $(OCILIB) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\webacmsxp.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\tpccdb.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\tpccdb.obj \
          $(DBLIBDIR)\NTWDBLIB.LIB \
          -Fe..\exe\webacmsxp.dll \
          $(ACMSXP_LIBS) $(WNT_LIBS) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

#..\exe\webmuxmsdblib.dll: \
#         $(webuiobjs) \
#         $(deliveryobjs) \
#         $(acmsxpobjs) \
#         $(ckptobjs) \
#         ..\obj\mux.obj \
#         ..\obj\tpccdb.obj
#    $(cc) \
#         $(webuiobjs) \
#         $(deliveryobjs) \
#         $(acmsxpobjs) \
#         $(ckptobjs) \
#         ..\obj\mux.obj \
#         ..\obj\tpccdb.obj \
#         $(DBLIBDIR)\NTWDBLIB.LIB \
#         -Fe..\exe\webmuxmsdblib.dll \
#         $(WNT_LIBS) \
#         -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\webacmsxpsyb.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\sybase_db.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\sybase_db.obj \
          $(SYBASELIB)\libsybdb.lib \
          -Fe..\exe\webacmsxpsyb.dll \
          $(ACMSXP_LIBS) $(WNT_LIBS) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\webacmsxpora.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db.obj ..\obj\oracle_txns.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db.obj ..\obj\oracle_txns.obj \
          $(ORALIB)\ociw32.lib \
          -Fe..\exe\webacmsxpora.dll \
          $(ACMSXP_LIBS) $(WNT_LIBS) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\webacmsxpora8.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db8.obj ..\obj\oracle_txns8.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db8.obj ..\obj\oracle_txns8.obj \
          $(OCILIB) \
          -Fe..\exe\webacmsxpora8.dll \
          $(ACMSXP_LIBS) $(WNT_LIBS) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)


..\exe\tuxora9.exe: \
                    $(webuiobjs) \
                    $(deliveryobjs) \
               $(tuxsrvobjs) \
                    $(ckptobjs) \
               ..\obj\oracle_db8.obj \
               ..\obj\oracle_txns8.obj
          $(cc) \
                    $(webuiobjs) \
                    $(deliveryobjs) \
                    $(ckptobjs) \
               $(tuxsrvobjs) \
               ..\obj\oracle_db8.obj \
               ..\obj\oracle_txns8.obj \
               $(OCILIB) \
               $(TUX_LIBS) \
               $(WNT_LIBS) \
               -Fe..\exe\tuxora9.exe \
               -link $(ldebug) -subsystem:console -incremental:no


..\exe\webtux.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(tuxsrvobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db8.obj ..\obj\oracle_txns8.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(tuxsrvobjs) \
          $(ckptobjs) \
          ..\obj\oracle_db8.obj ..\obj\oracle_txns8.obj \
          $(OCILIB) \
          -Fe..\exe\webtux.dll \
          $(TUX_LIBS) $(WNT_LIBS) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\webacmsxploop.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\dbloop.obj
    $(cc) \
          $(webuiobjs) \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\dbloop.obj \
          -Fe..\exe\webacmsxploop.dll \
          $(ACMSXP_LIBS) $(WNT_LIBS) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

#..\exe\webmuxloop.dll: \
#         $(webuiobjs) \
#         $(deliveryobjs) \
#         $(acmsxpobjs) \
#         $(ckptobjs) \
#         ..\obj\mux.obj \
#         ..\obj\dbloop.obj
#    $(cc) \
#         $(webuiobjs) \
#         $(deliveryobjs) \
#         $(acmsxpobjs) \
#         $(ckptobjs) \
#         ..\obj\mux.obj \
#         ..\obj\dbloop.obj \
#         -Fe..\exe\webmuxloop.dll \
#         $(WNT_LIBS) \
#         -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe\jacketacmsxploop.exe: \
          ..\obj\jacket.obj \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\dbloop.obj
    $(cc) \
          ..\obj\jacket.obj \
          $(deliveryobjs) \
          $(acmsxpobjs) \
          $(ckptobjs) \
          ..\obj\dbloop.obj \
          -Fe:..\exe\jacketacmsxploop.exe \
          $(ACMSXP_LIBS) $(WNT_LIBS) \
          -link $(ldebug) -subsystem:console -incremental:no
```

```
..\exe\oldwebtux.dll: \
          $(webuiobjs) \
          $(deliveryobjs) \
          .\tuxclient.c
          $(ckptobjs) \
    $(cc) -I%TUXDIR%\include -DTPMONITOR \
          $(webuiobjs) \
          $(deliveryobjs) \
          .\tuxclient.c \
          $(ckptobjs) \
          $(WNT_LIBS) \
          -Fe..\obj\oldwebtux.dll
          %TUXDIR%\lib\libtux.lib  %TUXDIR%\lib\libbuft.lib \
          %TUXDIR%\lib\libtux2.lib  %TUXDIR%\lib\libfml.lib \
          %TUXDIR%\lib\libfml32.lib %TUXDIR%\lib\libgp.lib \
          $(WNT_LIBS) \
          -link $(ldebug) -dll -subsystem:windows -incremental:no
$(EXPORTS)

..\exe/admin.dll : \
          ./admin.c
    $(cc) \
          ./admin.c \
          -Fo..\obj/admin.obj \
          -Fe..\exe/admin.dll \
          pdh.lib advapi32.lib \
          -link -dll  -subsystem:windows -incremental:no $(EXPORTS)

#####################################################################
############
#
#
#       Objects
#
#
#
#####################################################################
############

..\obj\jacket.obj: .\jacket.c $(TPCCHFILES)
    $(cc) -c .\jacket.c /Fo..\obj\jacket.obj

..\obj\tux_cli.obj: \
                ./tux_cli.c $(TPCCHFILES)
        $(cc) -c /I $(TUX_INC) -D_TMSTHREADS \
                ./tux_cli.c \
               /Fo..\obj\tux_cli.obj

..\obj\tux_srv.obj: \
                ./tux_srv.c $(TPCCHFILES)
        $(cc) -c /I $(TUX_INC) \
                ./tux_srv.c \
               /Fo..\obj\tux_srv.obj

..\obj/tpcc.o: ./tpcc.c $(TPCCHFILES)
    $(CC) -c ./tpcc.c /Fo..\obj/tpcc.o

..\obj\tpcc.obj: .\tpcc.c $(TPCCHFILES)
    $(cc) -c .\tpcc.c /Fo..\obj\tpcc.obj

..\obj/web_ui.o: ./web_ui.c $(TPCCHFILES)
    $(CC) -c ./web_ui.c /Fo..\obj/web_ui.o

..\obj\web_ui.obj: .\web_ui.c $(TPCCHFILES)
    $(cc) -c .\web_ui.c /Fo..\obj\web_ui.obj

..\obj\logfile.obj: .\logfile.c $(TPCCHFILES)
    $(cc) -c .\logfile.c /Fo..\obj\logfile.obj

..\obj\null_cli.obj: .\null_cli.c $(TPCCHFILES) \
          .\deli_srv.h
    $(cc) -c .\null_cli.c /Fo..\obj\null_cli.obj

..\obj\null_srv.obj: .\null_srv.c $(TPCCHFILES)
    $(cc) -c .\null_srv.c /Fo..\obj\null_srv.obj

#..\obj\mux.obj: .\mux.c \
#         $(acmsxphs)
#    $(cc) -c -I $(acmsxptempdir) $(acmsxpflags) \
#         .\mux.c /Fo..\obj\mux.obj

..\obj\tpccdb.obj: .\tpccdb.c $(TPCCHFILES)
    $(cc) -c .\tpccdb.c  /I $(DBLIBINC) /Fo..\obj\tpccdb.obj

..\obj\sybase_db.obj: .\sybase_db.c $(TPCCHFILES)
    $(cc) -c .\sybase_db.c  /I $(SYBASEINC) /Fo..\obj\sybase_db.obj

..\obj\deli_cli.obj: .\deli_cli.c $(TPCCHFILES) \
          .\deli_srv.h
    $(cc) -c .\deli_cli.c /I $(DBLIBINC) /Fo..\obj\deli_cli.obj

..\obj\deli_srv.obj: .\deli_srv.c $(TPCCHFILES) \
          .\deli_srv.h
    $(cc) -c .\deli_srv.c /I $(DBLIBINC) /Fo..\obj\deli_srv.obj

..\obj\ckpt.obj: .\ckpt.c $(TPCCHFILES) \
          .\ckpt.h
    $(cc) -c .\ckpt.c /I $(DBLIBINC) /Fo..\obj\ckpt.obj

..\obj\buf.obj: ..\..\prte\src\buf.c
    $(cc) -c ..\..\prte\src\buf.c /I $(DBLIBINC) /Fo..\obj\buf.obj

..\obj\deli_cli_ms.obj: .\deli_cli_ms.c .\deli_srv_ms.h
    $(cc) -c .\deli_cli_ms.c /I $(DBLIBINC)
/Fo..\obj\deli_cli_ms.obj

..\obj\deli_srv_ms.obj: .\deli_srv_ms.c .\deli_srv_ms.h
    $(cc) -c .\deli_srv_ms.c /I $(DBLIBINC)
/Fo..\obj\deli_srv_ms.obj

..\obj\tpcc_ps.obj: .\tpcc_ps.c
$(acmsxptempdir)\tpcc_stdl_workspaces.h
    $(cc) -c -I $(acmsxptempdir) $(acmsxpflags) \
          .\tpcc_ps.c /Fo..\obj\tpcc_ps.obj

..\obj\tpcc_fct.obj: .\tpcc_fct.c \
          .\tpcc_acmsxp.h .\deli_srv.h $(acmsxphs)
    $(cc) -c -I $(acmsxptempdir) $(acmsxpflags) \
          .\tpcc_fct.c /Fo..\obj\tpcc_fct.obj

..\obj/dbloop.o: ./dbloop.c $(TPCCHFILES)
    $(CC) -c ./dbloop.c  -Fo..\obj/dbloop.o

..\obj\dbloop.obj: .\dbloop.c $(TPCCHFILES)
    $(cc) -c .\dbloop.c  /Fo..\obj\dbloop.obj

..\obj\dbstub.obj: .\dbstub.c $(TPCCHFILES)
    $(cc) -c .\dbstub.c /Fo..\obj\dbstub.obj

..\obj\oracle_db.obj: .\oracle_db.c .\oracle_db.h \
          $(TPCCHFILES)
    $(cc) -c .\oracle_db.c  /I $(ORAINC) \
          /I . /Fo..\obj\oracle_db.obj

..\obj\oracle_txns.obj: .\oracle_txns.c .\oracle_db.h \
          $(TPCCHFILES)
    $(cc) -c .\oracle_txns.c  /I $(ORAINC) \
          /I . /Fo..\obj\oracle_txns.obj

..\obj\oracle_db8.obj: .\oracle_db8.c .\oracle_db8.h \
          $(TPCCHFILES)
    $(cc) -c .\oracle_db8.c  /I $(ORAINC) \
          /I . /Fo..\obj\oracle_db8.obj

..\obj\oracle_txns8.obj: .\oracle_txns8.c .\oracle_db8.h \
          $(TPCCHFILES)
    $(cc) -c .\oracle_txns8.c  /I $(ORAINC) \
          /I . /Fo..\obj\oracle_txns8.obj

..\obj\webdbg.obj: .\webdbg.c
    $(cc) -c .\webdbg.c /Fo..\obj\webdbg.obj

..\obj\webd.obj: .\webd.c
    $(cc) -c .\webd.c /Fo..\obj\webd.obj

..\obj\webstub.obj: .\webstub.c
    $(cc) -c .\webstub.c /Fo..\obj\webstub.obj


#****************************************************************
***
#*
#* acmsxp stuff
#*
#****************************************************************
**

#####################################################################
#########
#
# Generic ACMSxp server targets
#
#####################################################################
#########

#Note: crestdl takes tpcc_acmsxp.h and creates a the proper sizes
in
#      tpcc_stdl_workspaces.stdl -- this file ends up in ..\obj
always.
#      The STDL compiler will use this to generate
tpcc_stdl_workspaces.h

..\exe\crestdl.exe: .\crestdl.c .\tpcc_acmsxp.h .\tpccstruct.h
    $(cc) $(acmsxpflags) .\crestdl.c \
          /Fo..\obj /Fe..\exe\crestdl.exe \
          -link $(ldebug) -subsystem:console -incremental:no

$(acmsxptempdir)\tpcc_stdl_workspaces.stdl: ..\exe\crestdl.exe
          ..\exe\crestdl.exe -o $(acmsxptempdir)

$(acmsxptempdir)\tpcc_stdl_workspaces.h:
$(acmsxptempdir)\tpcc_stdl_workspaces.stdl
          stdl -k -M -h $(acmsxptempdir) -c
$(acmsxptempdir)\tpcc_stdl_workspaces.stdl

#####################################################################
############
# Start Web Server Stuff
#####################################################################
############

..\obj\tpcc_acmsxp_pp.obj : ..\obj\tpcc_acmsxp_pp.c
    $(cc) ..\obj\tpcc_acmsxp_pp.c

#Note that these .h&.c files are generated by the ACMSxp stdl
compiler.
#Hopefully we will be able to steer them elsewhere with the final
product.
#The -o switch does not currently do this.
$(acmsxpcs) \
$(acmsxphs) : .\tpcc_acmsxp_pp.stdl \
                    $(acmsxptempdir)\tpcc_stdl_workspaces.stdl
    stdl -T none -k -y -c -M -a async:c -i $(acmsxptempdir) -h
$(acmsxptempdir) .\tpcc_acmsxp_pp.stdl
```

```
..\obj\tpcc_dy_c_c.obj : $(acmsxptempdir)\tpcc_dy_c_c.c
    $(cc) -c $(acmsxptempdir)\tpcc_dy_c_c.c -
Fo..\obj\tpcc_dy_c_c.obj

..\obj\tpcc_no_c_c.obj : $(acmsxptempdir)\tpcc_no_c_c.c
    $(cc) -c $(acmsxptempdir)\tpcc_no_c_c.c -
Fo..\obj\tpcc_no_c_c.obj

..\obj\tpcc_os_c_c.obj : $(acmsxptempdir)\tpcc_os_c_c.c
    $(cc) -c $(acmsxptempdir)\tpcc_os_c_c.c -
Fo..\obj\tpcc_os_c_c.obj

..\obj\tpcc_pt_c_c.obj : $(acmsxptempdir)\tpcc_pt_c_c.c
    $(cc) -c $(acmsxptempdir)\tpcc_pt_c_c.c -
Fo..\obj\tpcc_pt_c_c.obj

..\obj\tpcc_sl_c_c.obj : $(acmsxptempdir)\tpcc_sl_c_c.c
    $(cc) -c $(acmsxptempdir)\tpcc_sl_c_c.c -
Fo..\obj\tpcc_sl_c_c.obj

..\obj\tpcc_gc_c_c.obj : $(acmsxptempdir)\tpcc_gc_c_c.c
    $(cc) -c $(acmsxptempdir)\tpcc_gc_c_c.c -
Fo..\obj\tpcc_gc_c_c.obj

#####################################################################
#############
# End Web Server Stuff
#####################################################################
#############



#####################################################################
############
#
#
#       Maintenance
#
#
#
#####################################################################
############

clean:
          - del ..\obj\*.obj
          - del ..\exe\*.exe
          - del ..\exe\*.exp
          - del ..\exe\*.dll
          - del ..\exe\*.lib
          - del ..\exe\*.pdb
          - del ..\exe\*.ilk
          - del ..\exe\*.map
          - del $(acmsxphs)
          - del $(acmsxptempdir)\*.h
          - del $(acmsxptempdir)\*.c
          - del $(acmsxptempdir)\*.stdl

special:
          $(SPECIAL)


--------------------------
ckpt.c
--------------------------
/*+**********************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1998 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*
```

```
**********************************************************************
*********/
/*+
 * Abstract: This file contains the Digital created front end
functions
 *                  for the tpcc benchmark.
 *
 * Author: W Carr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
*/

#include <windows.h>
#include <winsock2.h>
#include <stdio.h>

#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpccerr.h>

#include <tpcc.h>

#include <ckpt.h>

#define CALLERS_CONTEXT (pCallersContext)(0xffffffff)

static BOOL                     gbCKPTInitialized = FALSE;
static CRITICAL_SECTION         CKPTCriticalSection;
static DBContext    gDBC = INVALID_DB_CONTEXT;

int
CKPTStartup( void )
{
  LoginData                     Login;
  pLoginData                    pLogin = &Login;
  char                          tmp[16];
  DWORD                         dwTID;
  int                           retcode = ERR_SUCCESS;

  if( ! gbCKPTInitialized ) {
    InitializeCriticalSection( &CKPTCriticalSection );

    pLogin->pCC = CALLERS_CONTEXT;

    strcpy( pLogin->szUser, gszCkptUser );
    strcpy( pLogin->szPassword, gszCkptPassword );
    strcpy( pLogin->szServer, gszServer );
    strcpy( pLogin->szDatabase, gszDatabase );

    /* make up app_name */
    gethostname(tmp, sizeof(tmp));
    dwTID = GetCurrentThreadId( );
    sprintf( pLogin->szApplication, "%s:CP - %d", tmp, dwTID );

    retcode = CKPTConnect( pLogin );
    if( ERR_DB_SUCCESS == retcode )
      retcode = ERR_SUCCESS;

    gbCKPTInitialized = TRUE;
  }

  return retcode;
}

int
CKPTConnect( pLoginData pLogin )
{
  pLogin->status = TPCCConnectDB( &gDBC, pLogin );

  TPCCLog( "%s, dbprocptr = %8X\r\n", pLogin->szApplication, gDBC
);

  return pLogin->status;
}

int
CKPTDisconnect( void )
{
  return TPCCDisconnectDB( gDBC, CALLERS_CONTEXT );
}

int
CKPTShutdown( void )
{
  return CKPTDisconnect( );
}

int
TPCCCheckpoint( pCheckpointData pCheckpoint )
{
  int retcode;

  if( ! gbCKPTInitialized ) {
    retcode = ERR_CKPT_NOT_INITIALIZED;
  }
  else {
    EnterCriticalSection( &CKPTCriticalSection );
    retcode = TPCCCheckpointDB( gDBC, pCheckpoint );
    LeaveCriticalSection( &CKPTCriticalSection );

    if( ERR_DB_SUCCESS == retcode )
      retcode = ERR_SUCCESS;
  }
```

```
  return retcode;
}


--------------------------
ckpt.h
--------------------------
/*+**********************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1998 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED    *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE    *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER    *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY    *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY    *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE    *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT    *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS    *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

**********************************************************************
*********/
/*+
 * Abstract: This file contains the Digital created front end
functions
 *                  for the tpcc benchmark.
 *
 * Author: W Carr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
*/

int CKPTStartup( void );
int CKPTConnect( pLoginData pLogin );
int CKPTDisconnect( void );
int CKPTShutdown( void );
int TPCCCheckpoint( pCheckpointData pCheckpoint );



--------------------------
crestdl.c
--------------------------
/*+**********************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1998 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED    *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE    *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER    *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY    *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY    *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE    *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT    *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS    *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

**********************************************************************
*********/
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <windows.h>


#include <tpcc_acmsxp.h>

#define MAX(a,b) ((a)>(b)?(a):(b))

int __cdecl
main(int argc, char *argv[])
{
  FILE *fptr;
  char filename[] = "tpcc_stdl_workspaces.stdl";
  char fname[132];
  char *dir;
  int  iMaxSize;

  dir = getenv( "USR_OBJ" );

  if( 1 < argc ) {
    if( 0 == strcmp( "-o", argv[1] ))
      dir = argv[2];
  }

  sprintf(fname, "%s\\%s", dir, filename);
  if ( (fptr=fopen(fname,"w")) == NULL)
  {
    printf("\nCould not open file %s\n",fname);
    exit (0);
  }

  iMaxSize = 0;
  iMaxSize = MAX(iMaxSize,sizeof(DeliveryData));
  iMaxSize = MAX(iMaxSize,sizeof(NewOrderData));
  iMaxSize = MAX(iMaxSize,sizeof(OrderStatusData));
  iMaxSize = MAX(iMaxSize,sizeof(PaymentData));
  iMaxSize = MAX(iMaxSize,sizeof(StockLevelData));

  fprintf(fptr, "RECORD io_login_wksp\n");
  fprintf(fptr, "\tlogin_data TEXT SIZE %d;\n",sizeof( LoginData
));
  fprintf(fptr, "END RECORD;\n\n");
  fprintf(fptr, "RECORD io_dy_wksp\n");
  fprintf(fptr, "\tdy_data TEXT SIZE %d;\n",sizeof( DeliveryData
));
  fprintf(fptr, "END RECORD;\n\n");
  fprintf(fptr, "RECORD io_no_wksp\n");
  fprintf(fptr, "\tno_data TEXT SIZE %d;\n",sizeof( NewOrderData
));
  fprintf(fptr, "END RECORD;\n\n");
  fprintf(fptr, "RECORD io_pt_wksp\n");
  fprintf(fptr, "\tpt_data TEXT SIZE %d;\n",sizeof( PaymentData ));
  fprintf(fptr, "END RECORD;\n\n");
  fprintf(fptr, "RECORD io_os_wksp\n");
  fprintf(fptr, "\tos_data TEXT SIZE %d;\n",sizeof( OrderStatusData
));
  fprintf(fptr, "END RECORD;\n\n");
  fprintf(fptr, "RECORD io_sl_wksp\n");
  fprintf(fptr, "\tsl_data TEXT SIZE %d;\n",sizeof( StockLevelData
));
  fprintf(fptr, "END RECORD;\n\n");
  fprintf(fptr, "RECORD io_gc_wksp\n");
  fprintf(fptr, "\tgc_data TEXT SIZE %d;\n", iMaxSize );
  fprintf(fptr, "END RECORD;\n\n");
  fprintf(fptr, "RECORD int_gc_wksp\n");
  fprintf(fptr, "\ttrans INTEGER;\n");
  fprintf(fptr, "END RECORD;\n\n");

  fclose(fptr);
  printf("\n File %s generated.\n",fname);

  return 0;
}



--------------------------
deli_cli.c
--------------------------
/*+**********************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
```

```
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

/*+
 * Abstract: This file contains functions for the delivery server
client code.
 *
 * Author:  W Carr
 * Creation Date: July 1997
 *
 * Modified history:
 *
 *
*/

#define DELI_CLI_C

#include <windows.h>
#include <time.h>


#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpccerr.h>

#include <buf.h>
#include <deli_srv.h>


/* FUNCTION: int DELIClientStartup( void )
 *
 * PURPOSE:         This function prepares the delivery client for
processing.
 *
 * ARGUMENTS:       None
 *
 * RETURNS:         int                 Error code if unsuccessful
 *                                      ERR_SUCCESS if no error

 *
 * COMMENTS:        This code presumes that the server portion of
the code
 *                  has been started first, otherwise, a call to
the delivery
 *                  function will fail.
 *
 */

int DELIClientStartup( void )
{
  return ERR_SUCCESS;
}

/* FUNCTION: int DELIClientShutdown( void )
 *
 * PURPOSE:         This function cleans up allocated objects to
allow for
 *                  termination of the delivery subsystem.
 *
 * ARGUMENTS:       None
 *
 * RETURNS:         int                 Error code if unsuccessful
 *                                      ERR_SUCCESS if no error

 *
 * COMMENTS:        None
 *
 */

int DELIClientShutdown( void )
{
  return ERR_SUCCESS;
}

/* FUNCTION: int TPCCDelivery( CallersContext *pCC, int
iConnectionID,
 *                                  int iSyncID, DBContext
*pdbContext,
 *                                  int deadlock_retry,
pDeliveryData pDelivery )
 *
 * PURPOSE:         Writes the input portion the the delivery
structure to the
 *                  server.
 *
 * ARGUMENTS:       CallersContext      *pCC                callers
context struct ptr.
 *                  int                 iTermId
 *          terminal id of browser
 *                  int                 iSyncId             sync id
of browser
 *                  DBPROCESS *dbproc             connection db
process id
 *                  int                 deadlock_retry
 *          deadlock retry count
 *                  DeliveryData        *pDelivery
 *          Delivery data i/o struct ptr
 *
 * RETURNS:         int       ERR_DB_SUCCESS                success
 *                            ERR_DB_NOT_COMMITED other error
 *
 * COMMENTS:
 *
 */

int TPCCDelivery( pDeliveryData pDelivery,
                  pDeliveryData
CompletedDeliveries[DELIVERY_RESPONSE_COUNT] )
{
  size_t                    bw;
  size_t                    br;
  int                               ii;
  int                               status;

  /* get local queue time if nothing supplied */
  if( 0 == pDelivery->queue_time )
    time( &pDelivery->queue_time );

  if( gbUsePipe ) {
    if( 0 == WriteFile( ghPipeInputWrite, pDelivery,
sizeof(DeliveryDataInput),
                                &bw, NULL )) {
      status = GetLastError( );
      return ERR_DB_DELIVERY_NOT_QUEUED;
    }
  }
  else {
    /* Overload the delta time to be the local queue time in
milliseconds. */
    /* On the server side, we will get the tick count and subtract
to     */
    /* calcuate the delta.
                    */
    pDelivery->delta_time = GetTickCount( );

    /* Since the delivery transaction data structure has been
reserved by */
    /* the caller, we only wish to pass its address to the delivery
server. */
    /* Since the buffer code assumes that you wish to copy what is
at the */
    /* pointer value (it will dereference the pointer) what we want
to pass */
    /* is a pointer to the pointer. */
    if( BUF_SUCCESS != bufwrite( &pDelivery, sizeof( pDeliveryData
),
                                          &bw, INFINITE, inputbuf ))
      return ERR_DB_DELIVERY_NOT_QUEUED;

    for( ii = 0; ii < DELIVERY_RESPONSE_COUNT; ii++ ) {
      if( gbDisplayCompletions ) {
          status = bufread( &CompletedDeliveries[ii], sizeof(
pDeliveryData ),
                            &br, 0, outputbuf );
          if( BUF_READTIMEOUT == status )
            /* there was not a completed transaction waiting */
            CompletedDeliveries[ii] = NULL;
          else if( BUF_SUCCESS != status )
            return ERR_DELIVERY_OUTPUT_PIPE_READ;
      }
      else {
          /* no completion records are being sent */
          CompletedDeliveries[ii] = NULL;
      }
    }
  }

  return ERR_DB_SUCCESS;
}


--------------------------
deli_cli.h
--------------------------
#ifndef DELI_CLI_H
#define DELI_CLI_H
/*+**********************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
```

```
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

**********************************************************************
*********/
/*+
 * Abstract: This file contains definitions and declarations for
the
 *                  delivery server client code.
 *
 * Author: W Carr
 * Creation Date: July 1997
 *
 * Modified history:
 *
 *
*/

int DELIClientStartup( void );
int DELIClientShutdown( void );

#endif /* DELI_CLI_H */


--------------------------
deli_srv.c
--------------------------
/*+*******************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

/*+
 * Abstract: This file contains functions for the delivery server
queue.
 *
 * Author:  W Carr
 * Creation Date: July 1997
 *
 * Modified history:
 *
 *
*/

#define DELI_SRV_C

#include <windows.h>
#include <winsock.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <crtdbg.h>

#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpccerr.h>

#include <buf.h>
#include <tpcc.h>
#include <deli_srv.h>

#define FILENAMESIZE 256

static int          giDeliInitStatus = ERR_SUCCESS;
static LONG         gDeliThreadStartCtr;
static HANDLE       gDeliThreadStartEvent;

static int          iNumThreads         = 4;                /*
number threads to create */
static int          iNumQueued          = 10;               /* size
of delivery queue */
static int          iDeadlockRetry      = 3;                /*
number read check retries. */
static int          giLoginDelay        = 0;                /*
delay between db logins */

static FILE         *fpLog = NULL;                          /*
pointer to log file */
static char         szLogPath[256] = {0};

static BOOL         bDone;                                  /*
termination request flag */
static BOOL         bLog = FALSE;                           /* log
transactions to file ? */
static BOOL         bFlush = FALSE;                         /*
Flush log info when written*/
static BOOL         bDirectConnect = FALSE;                 /* Use
transport or connect */

/* Used for OS pipe deliveries */
static HANDLE       hPipeInputRead = INVALID_HANDLE_VALUE;

/* function prototypes */
void                DELIErrorMessage(int iError);
void                DELILog( pDeliveryData pDelivery );
unsigned  __stdcall DELIThread( void *ptr );
int                 DELIReadRegistrySettings(void);

/* FUNCTION: int DELIGetTransportData( BOOL *deli_direct_connect,
 *                                          int *num_dy_servers
)
 *
 * PURPOSE:         This function prepares the delivery subsystem
for execution.
 *
 * ARGUMENTS:       None
 *
 * RETURNS:         int       iError              Error code if
unsuccessfull
 *                            ERR_SUCCESS         No error
successfull code
 *
 *
 * COMMENTS:        None
 *
 */

int
DELIGetTransportData( BOOL *dy_use_transport, int *num_dy_servers,
                      int *num_queued_deliveries, int
*num_queued_responses )
{
  int                       status;

  if ( ERR_SUCCESS != (status = DELIReadRegistrySettings()) )
    return status;

  *dy_use_transport = !bDirectConnect;
  *num_dy_servers = iNumThreads;
  *num_queued_deliveries = iNumQueued;
  if( gbDisplayCompletions && !gbUsePipe )
    *num_queued_responses = iNumQueued;
  else
    *num_queued_responses = 0;

  return ERR_SUCCESS;
}

/* FUNCTION: int DELIServerStartup( BOOL deli_direct_connect,
 *                                          int num_dy_servers )
 *
```

```
 * PURPOSE:         This function prepares the delivery subsystem
for execution.
 *
 * ARGUMENTS:       None
 *
 * RETURNS:         int       iError              Error code if
unsuccessfull
 *                            ERR_SUCCESS         No error
successfull code
 *
 *
 * COMMENTS:        None
 *
 */

int
DELIServerStartup( BOOL dy_use_transport, int num_dy_servers,
                   int num_queued_deliveries, int
num_queued_responses )
{
  int                       iError;
  int                       ii;
  size_t            inputbufsize;
  size_t            outputbufsize;
  unsigned                  tid;
  unsigned long             ulhThread;
  HANDLE            hThread;

  if( gbUsePipe ) {
    /* start delivery pipe */
    inputbufsize = num_dy_servers *
      num_queued_deliveries * sizeof( DeliveryData );
    if(0 == CreatePipe(&hPipeInputRead, &ghPipeInputWrite, NULL,
inputbufsize)){
      iError = GetLastError( );
      return ERR_DELIVERY_PIPE_CREATE;
    }
  }
  else {
    /* create delivery buffer */
    inputbufsize = num_queued_deliveries * sizeof( pDeliveryData );
    if( BUF_SUCCESS != bufopen( inputbufsize, &inputbuf ))
      return ERR_DELIVERY_PIPE_OPEN;

    if( gbDisplayCompletions ) {
      /* create response buffer */
      outputbufsize = num_queued_responses * sizeof( pDeliveryData
);
      if( BUF_SUCCESS != bufopen( outputbufsize, &outputbuf ))
          return ERR_DELIVERY_PIPE_OPEN;
    }
  }

  /* prepare to start and verify starting of delivery threads */
  gDeliThreadStartCtr = iNumThreads;

  gDeliThreadStartEvent = CreateEvent( NULL, TRUE, FALSE, NULL );
  if ( gDeliThreadStartEvent == NULL )
    return ERR_CANT_CREATE_FORCE_THRED_STRT_EVENT;

  bDone = FALSE;

  if ( !bDone ){
    for( ii = 0; ii < iNumThreads; ii++ ) {
      ulhThread = _beginthreadex( NULL, 0, DELIThread, NULL, 0,
&tid );
      if( 0 == ulhThread )
          return ERR_CANT_START_DELIVERY_THREAD;
      hThread = ( HANDLE )ulhThread;
      CloseHandle( hThread );
      if( !dy_use_transport )
          Sleep( giLoginDelay );
    }
  }

  WaitForSingleObject( gDeliThreadStartEvent, INFINITE );

  return giDeliInitStatus;
}

/* FUNCTION: int DELIServerShutdown( void )
 *
 * PURPOSE:         This function cleans up allocated objects to
allow for
 *                  termination of the delivery subsystem.
 *
 * ARGUMENTS:       None
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

int
DELIServerShutdown( void )
{
  bDone = TRUE;

  if( gbUsePipe ) {
    CloseHandle( hPipeInputRead );
    CloseHandle( ghPipeInputWrite );
  }
  else {
    bufclose( inputbuf );
    bufclose( outputbuf );
  }

  if ( fpLog )
    fclose(fpLog);

  fpLog = NULL;

  return ERR_SUCCESS;
}

/* FUNCTION: void DELIErrorMessage(int iError)
 *
 * PURPOSE:         This function writes an error message to the
error log file.
 *
 * ARGUMENTS:       int                 iError    error id to be
logged
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

void
DELIErrorMessage(int iError)
{
  int ii;

  for( ii = 0; errorMsgs[ii].szMsg[0]; ii++ ) {
    if ( iError == errorMsgs[ii].iError ) {
      TPCCErr( "*Error(%d): %s\r\n", iError, errorMsgs[ii].szMsg );
      return;
    }
  }

  TPCCErr( "*Error(%d): Unknown Error.\r\n", iError );
  return;
}

/* FUNCTION: int DELIReadRegistrySettings(void)
 *
 * PURPOSE:         This function reads the registry for
initialization information.
 *
 * ARGUMENTS:       None
 *
 * RETURNS:         int       ERR_SUCCESS
 *                  Registry read Successful
 *
 *
 * COMMENTS:        None
 */

int
DELIReadRegistrySettings(void)
{
  HKEY    hKey;
  DWORD   size;
  DWORD   type;
  char    szTmp[FILENAMESIZE];
  int     status;
  int     iTmp;

  status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                        0, KEY_READ, &hKey);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_TPCC_KEY;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "LoginDelay", 0, &type, szTmp,
&size);
  if ( status == ERROR_SUCCESS )
    giLoginDelay = atoi(szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "NumberOfDeliveryThreads",
                           0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS )
    if ( 0 != ( iTmp = atoi(szTmp) ))
      iNumThreads = iTmp;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey,
"NumberOfQueuedDeliveryTransactions",
                           0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS )
    if ( 0 != ( iTmp = atoi(szTmp) ))
      iNumQueued = iTmp;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "DeadlockRetry", 0, &type, szTmp,
&size);
  if ( status == ERROR_SUCCESS )
    if ( 0 != ( iTmp = atoi(szTmp) ))
      iDeadlockRetry = iTmp;

  size = sizeof(szTmp);
  status =RegQueryValueEx(hKey, "DeliveryServerConnectsToDB",
                          0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS )
    if ( 0 == stricmp(szTmp, "TRUE") || 0 == strcmp(szTmp, "1"))
      bDirectConnect = TRUE;

  size = sizeof(szTmp);
  status =RegQueryValueEx(hKey, "FlushDeliveryLog", 0, &type,
szTmp, &size);
  if ( status == ERROR_SUCCESS )
    if ( 0 == stricmp(szTmp, "TRUE") || 0 == strcmp(szTmp, "1"))
```

```
      bFlush = TRUE;

  size = sizeof(szTmp);
  status =RegQueryValueEx(hKey, "DisplayDeliveryCompletions", 0,
&type, szTmp, &size);
  if ( status == ERROR_SUCCESS )
    if ( 0 == stricmp(szTmp, "FALSE") || 0 == strcmp(szTmp, "0"))
      gbDisplayCompletions = FALSE;

  size = sizeof(szTmp);
  status =RegQueryValueEx(hKey, "UseDeliveryPipe", 0, &type, szTmp,
&size);
  if ( status == ERROR_SUCCESS )
    if ( 0 == stricmp(szTmp, "TRUE") || 0 == strcmp(szTmp, "1"))
      gbUsePipe = TRUE;

  size = sizeof(szTmp);
  status =RegQueryValueEx(hKey, "WriteDeliveryLog", 0, &type,
szTmp, &size);
  if ( status == ERROR_SUCCESS )
    if ( 0 == stricmp(szTmp, "TRUE") || 0 == strcmp(szTmp, "1"))
      bLog = TRUE;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "PATH", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_PATH_VALUE;
  if( bLog ) {
    strcpy(szLogPath, szTmp);
    strcat(szLogPath, "delilog.");
    fpLog = fopen(szLogPath, "wb");
    if ( NULL == fpLog )
      return ERR_CANNOT_CREATE_RESULTS_FILE;
  }

  RegCloseKey(hKey);

  return ERR_SUCCESS;
}

/* FUNCTION: void TPCCDeliveryDeferredResponse( DeliveryData
*deliveryData )
 *
 * PURPOSE:         This function writes the completed delivery
transaction
 *                  record to the log file and to the queue for
reporting
 *                  to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK *pECB
 *                  int                             iTermId   client
browser terminal id
 *                  int                             iSyncId   client
browser sync id
 *                  DBContext           *pdb        pointer to the
database context
 *                  int                             retcode  return
code from db
 *                  DeliveryData                    *deliveryData
pointer to the delivery
 *                                                                data
structure.
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */

void
TPCCDeliveryDeferredResponse( int retcode, pDeliveryData pDelivery
)
{
  size_t                        bw;
  int                                   status;

  if ( ERR_DB_PENDING == retcode )
  {
    return;
  }
  else
  {
  /* The delta time was overloaded by the client to be the local
queue time */
  /* in milliseconds.  Here we get the tick count and subtract to
              */
  /* calcuate the delta.
                        */
    if( ERR_DB_SUCCESS != retcode)
    {
      /* send a flag to the reducer to mark an error on the
delivery */
      pDelivery->queue_time = 1;
      DELIErrorMessage(retcode);
    }
    pDelivery->delta_time = GetTickCount( ) - pDelivery-
>delta_time;

    /* update log */
    if( bLog )
      DELILog( pDelivery );

    if( gbDisplayCompletions && !gbUsePipe ) {
      /* send a delivery completion record */
      status = bufwrite( &pDelivery, sizeof(pDeliveryData),
                         &bw, INFINITE, outputbuf );
      if( BUF_SUCCESS != status ) {
          DELIErrorMessage( ERR_DELIVERY_OUTPUT_PIPE_WRITE );
          return;
      }
    }
    else {
      /* The delivery transaction is at the end of its life */
      UNRESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS, pDelivery );
    }
  }
}

/* FUNCTION: void DELIThread( void *ptr )
 *
 * PURPOSE:         This function is executed by each delivery
thread.
 *                  A thread will block on a read of the buffer
from the client.
 *                  Return from the read will indicate either data
is to be
 *                  processed, an error has occured, or the buffer
has been closed.
 *
 * ARGUMENTS:       void      *ptr      unused, passed by thread
creation routine.
 *
 * RETURNS:         None
 *
 * COMMENTS:        The registry key value NumberOfDeliveryThreads
controls how
 *                  many of these functions are running.
 *
 */

unsigned __stdcall
DELIThread( void *ptr )
{
  size_t                        br;
  pDeliveryData                             pDelivery;
  int                                       retcode;
  int                                       CC;
  CallersContext                *pCC;
  LoginData                                 login;
  char                                      tmp[16];
  int                                       status;
  DBContext                                 DBC;

  /* Initialize calling parameters */
  CC = 0;
  pCC = &CC;
  DBC = INVALID_DB_CONTEXT;

  /* Should we connect directly ourselves, or should we use the
transport to */
  /* perform the deliveries in the same manner as the other 4
transactions?  */
  if( bDirectConnect ) {
    gethostname(tmp, sizeof(tmp));
    login.w_id = 0;
    login.ld_id = 0;
    login.pCC = pCC;
    strcpy( login.szServer, gszServer );
    strcpy( login.szDatabase, gszDatabase );
    strcpy( login.szUser, gszUser );
    strcpy( login.szPassword, gszPassword );
    sprintf( login.szApplication, "%s:delisrv - %d",
                tmp, GetCurrentThreadId());
    status = TPCCConnectDB( &DBC, &login );
    if( ERR_DB_SUCCESS != status && ERR_SUCCESS != giDeliInitStatus
)
      giDeliInitStatus = status;
  }

  /* wait until all threads are started */
  if ( InterlockedDecrement( &gDeliThreadStartCtr ) == 0 )
    SetEvent( gDeliThreadStartEvent );

  WaitForSingleObject( gDeliThreadStartEvent, INFINITE );

  if( ERR_SUCCESS != giDeliInitStatus )
    return giDeliInitStatus;

  /* while delisrv running i.e. user has not requested termination
*/
  while( !bDone ){

    if( gbUsePipe ) {
      RESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS, pDelivery );
      if( 0 == ReadFile( hPipeInputRead, pDelivery, sizeof(
DeliveryDataInput ),
                         &br, NULL )) {
          status = GetLastError( );
          DELIErrorMessage( ERR_DELIVERY_PIPE_READ );
          continue;
      }
    }
    else {
      if(BUF_SUCCESS != bufread( &pDelivery, sizeof( pDeliveryData
), &br,
                                        INFINITE, inputbuf )) {
          DELIErrorMessage( ERR_DELIVERY_PIPE_READ );
          continue;
      }

    }
    if( bDirectConnect )
      retcode = TPCCDeliveryDB( DBC, pDelivery );
    else
      retcode = TPCCDeliveryDeferred( pDelivery );

    TPCCDeliveryDeferredResponse( retcode, pDelivery );
```

```
  }

  return ERR_SUCCESS;
}

/* FUNCTION: void DELILog( pDeliveryData pDelivery )
 *
 * PURPOSE:         Writes the delivery results to the delivery log
file.
 *
 * ARGUMENTS:       LPSYSTEMTIME        lpBegin             Local
delivery start time.
 *                  pDeliveryData       pDelivery Delivery data to
be written.
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

void
DELILog( pDeliveryData pDelivery )
{
  struct tm                   start;
  struct tm                   *end;
  time_t            endt;
  unsigned                    delta_time_seconds;
  unsigned                    delta_time_milliseconds;

  delta_time_seconds = pDelivery->delta_time / 1000;
  delta_time_milliseconds = pDelivery->delta_time -
(delta_time_seconds * 1000);

  CopyMemory( &start, localtime( &pDelivery->queue_time ), sizeof(
start ));
  endt = pDelivery->queue_time + delta_time_seconds;
  end = localtime( &endt );

  fprintf( fpLog,
             "%2.2d/%2.2d/%2.2d,"
             "%2.2d:%2.2d:%2.2d:%3.3d,"
             "%2.2d:%2.2d:%2.2d:%3.3d,"
             "%8.8d,"
             "%5.5d,%2.2d,"
             "%4.4d,%4.4d,%4.4d,%4.4d,%4.4d,"
             "%4.4d,%4.4d,%4.4d,%4.4d,%4.4d\r\n",
             start.tm_year, start.tm_mon, start.tm_mday,
             start.tm_hour, start.tm_min, start.tm_sec, 0,
             end->tm_hour, end->tm_min, end->tm_sec,
delta_time_milliseconds,
             pDelivery->delta_time,
             pDelivery->w_id, pDelivery->o_carrier_id,
             pDelivery->o_id[0], pDelivery->o_id[1],
             pDelivery->o_id[2], pDelivery->o_id[3],
             pDelivery->o_id[4], pDelivery->o_id[5],
             pDelivery->o_id[6], pDelivery->o_id[7],
             pDelivery->o_id[8], pDelivery->o_id[9] );

  if ( bFlush )
    fflush(fpLog);

  return;
}


--------------------------
deli_srv.h
--------------------------
#ifndef DELI_SRV_H
#define DELI_SRV_H
/*+**************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

/*+
 * Abstract: This file contains definitions of the delivery server
queue.
 *
 * Author:  W Carr
 * Creation Date: July 1997
 *
 * Modified history:
 *        21-Oct-1997 WCarr   Replaced NamedPipe with buffer code.
 *
*/

#ifdef DELI_SRV_C
# define DELI_GLOBAL(thing,init) thing = init
#else
# define DELI_GLOBAL(thing,init) extern thing
#endif

#if defined DELI_CLI_C || defined DELI_SRV_C
DELI_GLOBAL(BOOL gbDisplayCompletions,TRUE);      /* Send comp.
back to UI? */
DELI_GLOBAL(BUFPTR inputbuf,NULL);                /* submitted
deliveries */
DELI_GLOBAL(BUFPTR outputbuf,NULL);               /* completed
deliveries */

/* Used for OS pipe deliveries */
DELI_GLOBAL(BOOL gbUsePipe,FALSE);                /* Use OS pipe
delivery code */
DELI_GLOBAL(HANDLE ghPipeInputWrite,INVALID_HANDLE_VALUE);
#endif

int DELIGetTransportData( BOOL *dy_use_transport,
                                int *num_dy_servers,
                                int *num_queued_deliveries,
                                int *num_queued_responses );
int DELIServerStartup( BOOL dy_use_transport, int num_dy_servers,
                       int num_queued_deliveries, int
num_queued_responses );
int DELIServerShutdown( void );

#endif /* DELI_SRV_H */


--------------------------
logfile.c
--------------------------
/*+**************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

/*+
 * Abstract: This file contains the Digital created front end
functions
 *                 for the tpcc benchmark.
 *
```

```
 * Author: W Carr
 * Creation Date: October 1997
 *
 * Modified history:
 *
 *
*/

#include <windows.h>
#include <stdio.h>

#include <tpccerr.h>
#include <tpccstruct.h>
#include <tpccapi.h>

#include <tpcc.h>


static HANDLE                hLogFile  = INVALID_HANDLE_VALUE;

static CRITICAL_SECTION      ErrCriticalSection;
static CRITICAL_SECTION      LogCriticalSection;

/* FUNCTION: void TPCCOpenLog( void )
 *
 * PURPOSE:         This function opens the log file.
 *
 * ARGUMENTS:       None
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */
BOOL
TPCCOpenLog( void )
{
  char                       szFile[FILENAMESIZE];

  InitializeCriticalSection( &LogCriticalSection );
  InitializeCriticalSection( &ErrCriticalSection );

  strcpy( szFile, szTpccLogPath );
  strcat( szFile, "tpcclog" );
  hLogFile = CreateFile( szFile, GENERIC_WRITE, FILE_SHARE_READ,
NULL,
                         CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL );

  strcpy( szFile, szTpccLogPath );
  strcat( szFile, "tpccerr" );
  unlink( szFile );

  return( INVALID_HANDLE_VALUE != hLogFile );
}

/* FUNCTION: void TPCCCloseLog( void )
 *
 * PURPOSE:         This function closes the log file.
 *
 * ARGUMENTS:       None
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */
BOOL
TPCCCloseLog( void )
{
  CloseHandle( hLogFile );

  DeleteCriticalSection( &LogCriticalSection );
  DeleteCriticalSection( &ErrCriticalSection );

  return TRUE;
}

/* FUNCTION: void TPCCLog( char *szType, char *szStr )
 *
 * PURPOSE:         This function reports the date, time, operation
and
 *                  string to the log file.
 *
 * ARGUMENTS:       char    *szType     String containing the
operation type
 *                                      i.e. Query or Response.
 *                  char      *szStr    String associated with the
operation.
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */
void
TPCCLog( char *fmt, ... )
{
  va_list           marker;
  char                       szTmp[4096];
  char                       szArg[4096];
  SYSTEMTIME                 systemTime;
  int                        len;
  DWORD                      dwWriteLen;

  va_start( marker, fmt );
  vsprintf( szArg, fmt, marker );
  va_end( marker );

  GetLocalTime( &systemTime );

  len = sprintf( szTmp,
                 "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\t%s\r\n",
                 systemTime.wYear, systemTime.wMonth,
systemTime.wDay,
                 systemTime.wHour, systemTime.wMinute,
systemTime.wSecond,
                 szArg );

  EnterCriticalSection( &LogCriticalSection );
  WriteFile( hLogFile, szTmp, len, &dwWriteLen, NULL );
  LeaveCriticalSection( &LogCriticalSection );
}

void
TPCCErrInternal( char *szTmp, int len )
{
  DWORD                      dwWriteLen;
  HANDLE           hErrFile;
  char                       szFile[FILENAMESIZE];

  EnterCriticalSection( &ErrCriticalSection );

  strcpy( szFile, szTpccLogPath );
  strcat( szFile, "tpccerr" );

  hErrFile = CreateFile( szFile, GENERIC_WRITE, FILE_SHARE_READ,
NULL,
                         OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL,
NULL );
  SetFilePointer( hErrFile, 0, 0, FILE_END );

  WriteFile( hErrFile, szTmp, len, &dwWriteLen, NULL );

  CloseHandle( hErrFile );

  LeaveCriticalSection( &ErrCriticalSection );
}


void
TPCCErr( char *fmt, ...)
{
  va_list           marker;
  char                       szTmp[4096];
  char                       szArg[4096];
  SYSTEMTIME                 systemTime;
  int                        len;

  va_start( marker, fmt );
  vsprintf( szArg, fmt, marker );
  va_end( marker );

  GetLocalTime( &systemTime );

  len = sprintf( szTmp,
                 "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\t%s\r\n",
                 systemTime.wYear, systemTime.wMonth,
systemTime.wDay,
                 systemTime.wHour, systemTime.wMinute,
systemTime.wSecond,
                 szArg );

  TPCCErrInternal( szTmp, len );
}


void
TPCCTransactionErr( pConnData pConn, char *fmt, ...)
{
  va_list           marker;
  char                       szTmp[4096];
  char                       szArg[4096];
  SYSTEMTIME                 systemTime;
  int                        len;

  va_start( marker, fmt );
  vsprintf( szArg, fmt, marker );
  va_end( marker );

  GetLocalTime( &systemTime );

  len = sprintf( szTmp,
                 "%2.2d/%2.2d/%2.2d
%2.2d:%2.2d:%2.2d\tTransaction error. w_id: %d, ld_id: %d, pCC: %x,
status: %d, dbstatus: %d, %s\r\n",
                 systemTime.wYear, systemTime.wMonth,
systemTime.wDay,
                 systemTime.wHour, systemTime.wMinute,
systemTime.wSecond,
                 pConn->w_id, pConn->ld_id, pConn->pCC,
                 pConn->status, pConn->dbstatus,
                 szArg );

  TPCCErrInternal( szTmp, len );
}


--------------------------
oracle_db8.c
--------------------------
/*+ file: oracle_db8.c based on Oracle file tpccpl.c
/*+================================================================
=+
 |        Copyright (c) 1994  Oracle Corp, Redwood Shores, CA
 |
```

```
 |                   OPEN SYSTEMS PERFORMANCE GROUP
|
 |                        All Rights Reserved
|

+===============================================================
+
 | DESCRIPTION
 |    TPC-C transactions in PL/SQL.

+===============================================================-
*/
/*+***************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1998 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

#include <windows.h>
#include <process.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/timeb.h>

#include <oci.h>
#include <ocidfn.h>
#include <ociapr.h>

#define ORACLE_DB_C

#include <tpccerr.h>
#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpcc.h>
#include <oracle_db8.h>

#include <crtdbg.h>

#define DEADLOCKRETRIES 6

static int  bTpccExit;        /* exit delivery disconnect loop as
dll exiting. */
static    CRITICAL_SECTION    ErrorLogCriticalSection;




char szErrorLogName[256];
char szOraLogName[256];
char szOraErrorLogName[256];

/* prototypes */
int ORAReadRegistrySettings(void);
void vgetdate (unsigned char *oradt) ;
void cvtdmy (unsigned char *oradt, char *outdate);
void cvtdmyhms (unsigned char *oradt, char *outdate);


FILE  *vopen(char *fnam, char *mode)
{
FILE *fd;

#ifdef  DEBUG
  TPCCLog("tkvuopen() fnam: %s,   mode: %s\n", fnam, mode);
#endif

    fd = fopen((char *)fnam,(char *)mode);
    if (!fd){
        TPCCLog(" fopen on %s failed %d\n",fnam,fd);
    }
    return(fd);
}

int sqlfile(char *fnam, text *linebuf)
{
FILE *fd;
int nulpt = 0;

#ifdef  DEBUG
  TPCCLog("sqlfile() fnam: %s,  linebuf: %#x\n", fnam, linebuf);
#endif

    fd = vopen(fnam,"r");
    if(NULLP == fd)
    {
      return(ERR_DB_ERROR);
    }
    while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
    {
        nulpt = strlen((char *)linebuf);
    }
    return(nulpt);
}

int getfile(char *filename, text *filebuf)
{
  text parsbuf[SQL_BUF_SIZE];

  strcpy(parsbuf, szTpccLogPath);
  strcat(parsbuf, filename);
  return(sqlfile(parsbuf, filebuf));

}


int TPCCStartupDB( int iMaxConnections )
{
#ifdef DEBUG_TPCCSTARTUPDB
  _ASSERT(FALSE);
#endif

  InitializeCriticalSection(&ErrorLogCriticalSection);

  /* Tell Oracle this is multi-threaded code */
  OCIInitialize(OCI_THREADED | OCI_OBJECT ,(dvoid *)0,0,0,0);

  /* get values from the registry */
  ORAReadRegistrySettings();

  return ERR_DB_SUCCESS;

}

int TPCCShutdownDB(void)
{

  bTpccExit = TRUE;

  /* Add Oracle specific code */

  DeleteCriticalSection(&ErrorLogCriticalSection);

  return ERR_DB_SUCCESS;
}


int ocierror(char *fname, int lineno, OraContext *p, sword status)
{
  text errbuf[512];
  text tempbuf[512];
  sb4 errcode;
  OCIError *errhp;

  errhp = p->errhp;

  switch (status) {
  case OCI_SUCCESS:
    return RECOVERR;
    break;
  case OCI_SUCCESS_WITH_INFO:
    sprintf(errbuf, "Module %s Line %d\r\n", fname, lineno);
    strcat(errbuf, "Error - OCI_SUCCESS_WITH_INFO\r\n");
    break;
  case OCI_NEED_DATA:
    sprintf(errbuf, "Module %s Line %d\r\n", fname, lineno);
    strcat(errbuf, "Error - OCI_NEED_DATA\r\n");
    break;
  case OCI_NO_DATA:
    sprintf(errbuf, "Module %s Line %d\r\n", fname, lineno);
    sprintf(errbuf, "Error - OCI_NO_DATA\r\n");
    break;
  case OCI_ERROR:
    (void) OCIErrorGet (errhp, (ub4) 1,
                                (text *) NULL, &errcode, tempbuf,
                                (ub4) sizeof(errbuf),
OCI_HTYPE_ERROR);

    switch(errcode){
    case NOT_SERIALIZABLE:
    /* if error is NOT_SERIALIZABLE return without writing anything
*/
      return errcode;

    case DEADLOCK:
```

```
      TPCCLog("Warning Deadlock, being retried");
      return RECOVERR;

    case SNAPSHOT_TOO_OLD:
      /* SNAPSHOT_TOO_OLD is considered recoverable */
      TPCCLog("Error snapshot too old: %s", tempbuf);
      return RECOVERR;

    default:
    /* else write a message, All else are irrecoverable */
      TPCCLog("Module %s Line %d\r\nError - %s\r\n",
              fname, lineno, tempbuf);
      return errcode;
    }

    break;
  case OCI_INVALID_HANDLE:
    sprintf(errbuf, "Module %s Line %d\r\n", fname, lineno);
    strcat(errbuf, "Error - OCI_INVALID_HANDLE\r\n");
    TPCCLog("%s", errbuf);
    TPCCDisconnectDB(p, NULL);
    return IRRECERR;
    break;
  case OCI_STILL_EXECUTING:
    sprintf(errbuf, "Module %s Line %d\r\n", fname, lineno);
    strcat(errbuf, "Error - OCI_STILL_EXECUTE\r\n");
    break;
  case OCI_CONTINUE:
    sprintf(errbuf, "Module %s Line %d\r\n", fname, lineno);
    strcat(errbuf, "Error - OCI_CONTINUE\r\n");
    break;
  default:
    break;
  }
  TPCCLog("%s", errbuf);
  return RECOVERR;
}


/* FUNCTION: int TPCCConnectDB(CallersContext *pCC, int iTermId,
int iSyncId,
 *  OraContext **dbproc, char *server, char *database, char *user,
 *  char *password, char *app, int *spid, long *pack_size)
 *
 * PURPOSE:         This function opens the sql connection for use.
 *
 * ARGUMENTS:       CallersContext      *pCC      passed in
structure pointer from inetsrv.
 *                  int                           iTermId
 *          terminal id of browser
 *                  int                           iSyncId
 *          sync id of browser
 *                  OraContext                    **dbproc  pointer
to returned OraContext
 *                  char                          *server
 *          SQL server name
 *                  char                          *database SQL
server database
 *                  char                          *user
 *          user name
 *                  char                          *password user
password
 *                  char                          *app
 *          pointer to returned application array
 *                  int                           *spid
 *          pointer to returned spid
 *                  long                          *pack_size
 *          pointer to returned default pack size
 *
 * RETURNS:                   int       0         if successful
 *                                      1         if an error
occurs
 *
 * COMMENTS:        None
 *
 */


int TPCCConnectDB(OraContext **dbproc, pLoginData pLogin)
{

#define SERIAL_TXT "alter session set isolation_level =
serializable"
#ifdef SQL_TRACE
#define SQLTXT1 "alter session set sql_trace = true"
#endif

  /* Add Oracle specific code */

  text stmbuf[100];
  OraContext *p;
  char userstr[256];

  *dbproc = (OraContext *) malloc(sizeof(OraContext));

  p = *dbproc;

  /* initialize flags to not initialized */
  p->new_init = 0;
  p->pay_init = 0;
  p->ord_init = 0;
  p->sto_init = 0;
  p->del_init = 0;

  sprintf(userstr,"%s/%s@%s",
           pLogin->szUser,pLogin->szPassword,pLogin->szServer);

  OCIEnvInit(&(p->tpcenv), OCI_DEFAULT, 0, (dvoid **)0);

  OCIHandleAlloc((dvoid *)p->tpcenv, (dvoid **)&(p->tpcsrv),
OCI_HTYPE_SERVER,
                 0 , (dvoid **)0);
  OCIHandleAlloc((dvoid *)p->tpcenv, (dvoid **)&(p->errhp),
OCI_HTYPE_ERROR,
                 0 , (dvoid **)0);
  OCIHandleAlloc((dvoid *)p->tpcenv, (dvoid **)&(p->datecvterrhp),
OCI_HTYPE_ERROR,
                 0 , (dvoid **)0);
  OCIHandleAlloc((dvoid *)p->tpcenv, (dvoid **)&(p->tpcsvc),
OCI_HTYPE_SVCCTX,
                 0 , (dvoid **)0);
    if (RECOVERR != (OCIERROR(p, OCIServerAttach(p->tpcsrv, p->errhp,
                                                 (text *)0,
0, OCI_DEFAULT))))
      return ERR_DB_ERROR;

  OCIAttrSet((dvoid *)p->tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)p-
>tpcsrv,
             (ub4)0, OCI_ATTR_SERVER, p->errhp);
  OCIHandleAlloc((dvoid *)p->tpcenv, (dvoid **)&(p->tpcusr),
OCI_HTYPE_SESSION,
                 0 , (dvoid **)0);
  OCIAttrSet((dvoid *)p->tpcusr, OCI_HTYPE_SESSION, (dvoid *)pLogin-
>szUser,
             (ub4)strlen(pLogin->szUser),OCI_ATTR_USERNAME, p-
>errhp);
  OCIAttrSet((dvoid *)p->tpcusr, OCI_HTYPE_SESSION,
             (dvoid *)pLogin->szPassword,
             (ub4)strlen(pLogin->szPassword), OCI_ATTR_PASSWORD,
p->errhp);
    if (RECOVERR != (OCIERROR(p, OCISessionBegin(p->tpcsvc, p->errhp,
p->tpcusr,
                                                 OCI_CRED_RDBMS,
OCI_DEFAULT))))
      return (ERR_DB_ERROR);

   OCIAttrSet(p->tpcsvc, OCI_HTYPE_SVCCTX, p->tpcusr, 0,
OCI_ATTR_SESSION,
              p->errhp);

    /* run all transaction in serializable mode */

  OCIHandleAlloc(p->tpcenv, (dvoid **)&(p->curi), OCI_HTYPE_STMT, 0,
(dvoid**)0);
    sprintf ((char *) stmbuf, SERIAL_TXT);
  OCIStmtPrepare(p->curi, p->errhp, stmbuf, strlen((char *)stmbuf),
                 OCI_NTV_SYNTAX, OCI_DEFAULT);
    if (RECOVERR != OCIERROR(p, OCIStmtExecute(p->tpcsvc, p->curi, p-
>errhp,
                                               1, 0, 0, 0,
OCI_DEFAULT)))
      return (ERR_DB_ERROR);
  OCIHandleFree(p->curi, OCI_HTYPE_STMT);


#ifdef SQL_TRACE
    /* Turn on the SQL_TRACE */
    OCIHandleAlloc(p->tpcenv, (dvoid **)&(p->curi), OCI_HTYPE_STMT,
0, &xmem);
    sprintf ((char *) stmbuf, TRACE_TXT);
    OCIStmtPrepare(p->curi, p->errhp, stmbuf, strlen((char *)stmbuf),
                   OCI_NTV_SYNTAX, OCI_DEFAULT);
    if (RECOVERR != OCIERROR(p, OCIStmtExecute(p->tpcsvc, p->curi, p-
>errhp,
                                               1, 0, 0, 0,
OCI_DEFAULT)))
      return(ERR_DB_ERROR);
    OCIHandleFree((dvoid *)p->curi, OCI_HTYPE_STMT);
#endif /* End SQL_TRACE */

    /****  logon = 1;***/

    if (tkvcninit (&(p->bindvars.info.newOrder), p)) {
      TPCCDisconnectDB (p, NULL);
      return ERR_DB_ERROR;
    }
    else
      p->new_init = 1;

    if (tkvcpinit (&(p->bindvars.info.payment), p)) {
      TPCCDisconnectDB (p, NULL);
      return ERR_DB_ERROR;
    }
    else
      p->pay_init = 1;

    if (tkvcoinit (&(p->bindvars.info.orderStatus), p)) {
      TPCCDisconnectDB (p, NULL);
      return ERR_DB_ERROR;
    }
    else
      p->ord_init = 1;

    if (tkvcsinit (&(p->bindvars.info.stockLevel), p)) {
      TPCCDisconnectDB (p, NULL);
      return ERR_DB_ERROR;
    }
    else
      p->sto_init = 1;

    if (tkvcdinit (&(p->bindvars.info.delivery), p)) {
      TPCCDisconnectDB (p, NULL);
      return ERR_DB_ERROR;
    }
    else
      p->del_init = 1;
```

```
  return ERR_DB_SUCCESS;
}


/* FUNCTION: int TPCCDisconnectDB(OraContext *dbproc)
 *
 * PURPOSE:         This function closes the sql connection.
 *
 * ARGUMENTS:
 *                  OraContext          *dbproc   pointer to
OraContext
 *
 * RETURNS:         int       ERR_DB_SUCCESS      if successfull
 *                            error value         if an error
occurs
 *
 * COMMENTS:        None
 *
 */


int TPCCDisconnectDB(OraContext *dbproc, CallersContext *pCC){

  /* Add Oracle specific code */


   if (1 == dbproc->new_init) {
      tkvcndone(&(dbproc->nctx));
      dbproc->new_init = 0;
   }

   if (1 == dbproc->pay_init) {
      tkvcpdone(&(dbproc->pctx));
      dbproc->pay_init = 0;
   }
   if (1 == dbproc->ord_init) {
      tkvcodone(&(dbproc->octx));
      dbproc->ord_init = 0;
   }
   if (1 == dbproc->sto_init) {
      tkvcsdone(&(dbproc->sctx));
      dbproc->sto_init = 0;
   }
   if (1 == dbproc->del_init) {
      tkvcddone(&(dbproc->dctx));
      dbproc->del_init = 0;
   }

   OCIHandleFree((dvoid *)dbproc->tpcusr, OCI_HTYPE_SESSION);
   OCIHandleFree((dvoid *)dbproc->tpcsvc, OCI_HTYPE_SVCCTX);
   OCIHandleFree((dvoid *)dbproc->errhp, OCI_HTYPE_ERROR);
   OCIHandleFree((dvoid *)dbproc->datecvterrhp, OCI_HTYPE_ERROR);
   OCIHandleFree((dvoid *)dbproc->tpcsrv, OCI_HTYPE_SERVER);
   OCIHandleFree((dvoid *)dbproc->tpcenv, OCI_HTYPE_ENV);

#ifdef BATCH_DEL
   if (lfp) {
      fclose (lfp);
      lfp = NULL;
   }
#endif /* BATCH_DEL */


  return ERR_DB_SUCCESS;
}

/* FUNCTION: TPCCStockLevelDB(CallersContext      *pCC, int
iTermId, int iSyncId, OraContext *dbproc,  int deadlock_retry,
StockLevelData *pStockLevel)
 *
 * PURPOSE:         This function handles the stock level
transaction.
 *
 * ARGUMENTS:       CallersContext      *pCC
 *         passed in structure pointer from inetsrv.
 *                  int                 iTermId
 *         terminal id of browser
 *                  int                 iSyncId
 *         sync id of browser
 *                  OraContext          *dbproc
 *         connection db process id
 *                  StockLevelData      *pStockLevel
 *         stock level input / output data structure
 *                  int                 deadlock_retry
 *         retry count if deadlocked
 *
 * RETURNS:         int       ERR_DB_SUCCCESS     if successfull
 *                            error value         if deadlocked
 *
 * COMMENTS:        None
 *
 */

int TPCCStockLevelDB(OraContext *dbproc, pStockLevelData
pStockLevel)
{


  int tries,status;
  StockLevelData *pbindvars;

  pbindvars = &dbproc->bindvars.info.stockLevel;

  memcpy(pbindvars, pStockLevel, sizeof(StockLevelData));


  for ( tries = 0,status = RECOVERR;
        tries < DEADLOCKRETRIES && status == RECOVERR;  tries++)
{

    status = tkvcs(dbproc);
  }

  pStockLevel->low_stock = dbproc-
>bindvars.info.stockLevel.low_stock;
  if (status == RECOVERR) return ERR_DB_DEADLOCK_LIMIT;
  else return (status);

}

/* FUNCTION: int TPCCNewOrderDB(CallersContext *pCC, int iTermId,
int iSyncId, int iTermId, int iSyncId, OraContext *dbproc, int
deadlock_retry, NewOrderData  *pNewOrder)
 *
 * PURPOSE:         This function handles the new order
transaction.
 *
 * ARGUMENTS:       CallersContext      *pCC
 *         passed in structure pointer from inetsrv.
 *                  int                 iTermId
 *         terminal id of browser
 *                  int                 iSyncId
 *         sync id of browser
 *                  OraContext          *dbproc
 *         connection db process id
 *                  NewOrderData        *pNewOrder
 *         pointer to new order structure for input/output data
 *                  int                 deadlock_retry
 *         retry count if deadlocked
 *
 * RETURNS:         int       ERR_DB_SUCCESS
 *         transaction committed
 *                            ERR_DB_NOT_COMMITTED    item number
is not valid
 *                            ERR_DB_DEADLOCK_LIMIT
 *         deadlock max retry reached
 *                            ERR_DB_ERROR
 *
 *
 * COMMENTS:        None
 *
 */

#pragma message ("FIXME: return code is overloaded.  How to report
invalid item number?")
int TPCCNewOrderDB( OraContext *dbproc, pNewOrderData  pNewOrder)
{

  int tries,status;
  int ii;
  int jj;
  int datebufsize;
  OCIError *datecvterrhp = dbproc->datecvterrhp;
  unsigned char localcr_date[7];

  NewOrderData *pbindvars = &(dbproc->bindvars.info.newOrder);
  newctx *nctx = &(dbproc->nctx);
  newtemp *ntemp = &(dbproc->tempvars.new);

  vgetdate(localcr_date);
  cvtdmyhms(localcr_date,ntemp->entry_date);
  OCIDateFromText(datecvterrhp,ntemp->entry_date,strlen(ntemp-
>entry_date),"DD-MM-YYYY HH24:MI:SS",21,(text *) 0,0,&ntemp-
>cr_date);

  ntemp->n_retry = 0;

  memcpy(pbindvars, pNewOrder, sizeof(NewOrderData));
  for(jj= 0; jj<MAX_OL; jj++)
    {
      ntemp->nol_i_id[jj] = pbindvars->o_ol[jj].ol_i_id;
      ntemp->nol_supply_w_id[jj] = pbindvars-
>o_ol[jj].ol_supply_w_id;
      ntemp->nol_quantity[jj] = pbindvars->o_ol[jj].ol_quantity;
    }

  for ( tries = 0,status = RECOVERR;
        tries < DEADLOCKRETRIES && status == RECOVERR; tries++)
    {
      status = tkvcn(&dbproc->bindvars.info.newOrder, dbproc);
    }

  memcpy(pNewOrder, pbindvars, sizeof(NewOrderData));

  /* convert and/or copy data to our structure format */
  pNewOrder->c_discount = ntemp->c_discount*100.0;
  pNewOrder->w_tax = (float)ntemp->w_tax*100.0;
  pNewOrder->d_tax = (float)ntemp->d_tax*100.0;


  for (ii = 0; ii < pNewOrder->o_ol_cnt; ii++)
    {
      pNewOrder->o_ol[ii].ol_i_id = ntemp->nol_i_id[ii];
      pNewOrder->o_ol[ii].ol_supply_w_id = ntemp-
>nol_supply_w_id[ii];
      pNewOrder->o_ol[ii].ol_quantity = ntemp->nol_quantity[ii];
      strncpy(pNewOrder->o_ol[ii].i_name, ntemp->i_name[ii], 24);
      pNewOrder->o_ol[ii].s_quantity = ntemp->s_quantity[ii];
      pNewOrder->o_ol[ii].i_price = ntemp->i_price[ii]/100.0;
      pNewOrder->o_ol[ii].ol_amount = ntemp->nol_amount[ii]/100.0;
      pNewOrder->o_ol[ii].b_g[0]=ntemp->brand_generic[ii];
    }

 datebufsize=21;
  pNewOrder->o_entry_d.day = atoi(&(ntemp->entry_date[0]));
```

```
  pNewOrder->o_entry_d.month = atoi(&(ntemp->entry_date[3]));
  pNewOrder->o_entry_d.year = atoi(&(ntemp->entry_date[6]));
  pNewOrder->o_entry_d.hour = atoi(&(ntemp->entry_date[11]));
  pNewOrder->o_entry_d.minute = atoi(&(ntemp->entry_date[14]));
  pNewOrder->o_entry_d.second = atoi(&(ntemp->entry_date[17]));

  if (status == RECOVERR)
      return ERR_DB_DEADLOCK_LIMIT;
  else
      return (status);
}

/* FUNCTION: int TPCCPaymentDB(CallersContext *pCC, int iTermId,
int iSyncId, OraContext *dbproc, int deadlock_retry, PaymentData
*pPayment)
 *
 * PURPOSE:         This function handles the payment transaction.
 *
 * ARGUMENTS:       CallersContext      *pCC
 *         passed in structure pointer from inetsrv.
 *                  int                 iTermId
 *         terminal id of browser
 *                  int                 iSyncId
 *         sync id of browser
 *                  OraContext          *dbproc
 *         connection db process id
 *                  PaymentData         *pPayment           pointer
to payment input/output data structure
 *                  int                 deadlock_retry
 *         deadlock retry count
 *
 * RETURNS:         int       ERR_DB_SUCCESS                success
 *                            ERR_DB_DEADLOCK_LIMIT         max
deadlocked reached
 *                            ERR_DB_NOT_COMMITED invalid data
entry
 *
 * COMMENTS:        None
 *
 */

int TPCCPaymentDB(OraContext *dbproc, pPaymentData pPayment)
{
  int tries;
  int status;
  int datebufsize;
  OCIError *datecvterrhp = dbproc->datecvterrhp;

  PaymentData *pbindvars = &(dbproc->bindvars.info.payment);
  payctx *pctx = &(dbproc->pctx);
  paytemp *ptemp = &(dbproc->tempvars.pay);

  ptemp->p_retry = 0;

  memcpy(pbindvars, pPayment, sizeof(PaymentData));

  /* the db is stored in pennies - convert input to cents. */
  ptemp->h_amount = (int)(pbindvars->h_amount*100);

  for ( tries = 0,status = RECOVERR;
          tries < DEADLOCKRETRIES && status == RECOVERR; tries++) {

    if ((pbindvars->c_id) == 0) {
      (pbindvars->byname) = TRUE;
    }
    else {
      (pbindvars->byname) = FALSE;
    }

    status = tkvcp(&dbproc->bindvars.info.payment, dbproc);

  }

  memcpy(pPayment, pbindvars, sizeof(PaymentData));
 datebufsize=11;
  /* convert date format */
  OCIDateToText(datecvterrhp, &ptemp->customer_sdate,(text *) "DD-
MM-YYYY", 10, (text *) 0, 0, &datebufsize, &ptemp->c_since_str);
 datebufsize=DATE_SIZ;
  pPayment->c_credit_lim = ptemp->c_credit_lim/100.0;
  pPayment->c_discount = ptemp->c_discount*100.0;
  pPayment->c_balance = pPayment->c_balance/100.0;
  pPayment->h_amount = ptemp->h_amount/100.0;

  pPayment->c_since.day = atoi(&(ptemp->c_since_str[0]));
  pPayment->c_since.month = atoi(&(ptemp->c_since_str[3]));
  pPayment->c_since.year = atoi(&(ptemp->c_since_str[6]));
  pPayment->h_date.day = atoi(&(ptemp->h_date[0]));
  pPayment->h_date.month = atoi(&(ptemp->h_date[3]));
  pPayment->h_date.year = atoi(&(ptemp->h_date[6]));
  pPayment->h_date.hour = atoi(&(ptemp->h_date[11]));
  pPayment->h_date.minute = atoi(&(ptemp->h_date[14]));
  pPayment->h_date.second = atoi(&(ptemp->h_date[17]));

  if (status == RECOVERR) return ERR_DB_DEADLOCK_LIMIT;
  else
  {
   if (status == ERR_DB_ERROR)
   {
      return (status);
   }
   else
   {
      return (status);
   }
  }
}
```

```
/* FUNCTION: int TPCCOrderStatusDB(CallersContext *pCC, int
iTermId, int iSyncId, OraContext *dbproc, int deadlock_retry,
OrderStatusData *pOrderStatus)
 *
 * PURPOSE:         This function processes the Order Status
transaction.
 *
 * ARGUMENTS:       CallersContext      *pCC
 *         passed in structure pointer from inetsrv.
 *                  int                 iTermId
 *         terminal id of browser
 *                  int                 iSyncId
 *         sync id of browser
 *                  OraContext          *dbproc
 *         connection db process id
 *                  OrderStatusData     *pOrderStatus
 *         pointer to Order Status data input/output structure
 *                  int                 deadlock_retry
 *         deadlock retry count
 *
 * RETURNS:         int       ERR_DB_DEADLOCK_LIMIT
 *         max deadlock reached
 *                            ERR_DB_NOT_COMMITED           No
orders found for customer
 *                            ERR_DB_SUCCESS
 *         Transaction successfull
 *
 * COMMENTS:        None
 *
 */

int TPCCOrderStatusDB(OraContext *dbproc, pOrderStatusData
pOrderStatus)
{

  int tries,status;
  int ii;
  OrderStatusData *pbindvars = &(dbproc-
>bindvars.info.orderStatus);
  ordtemp *otemp = &(dbproc->tempvars.ord);
  OCIError *datecvterrhp = dbproc->datecvterrhp;

  memcpy(pbindvars, pOrderStatus, sizeof(OrderStatusData));

  for ( tries = 0,status = RECOVERR;
      tries < DEADLOCKRETRIES && status == RECOVERR; tries++) {

    if ((pbindvars->c_id) == 0) {
      (pbindvars->byname) = TRUE;
    }
    else {
      (pbindvars->byname) = FALSE;
    }

    status = tkvco(&dbproc->bindvars.info.orderStatus, dbproc);

 }
  if (status == ERR_DB_ERROR) return status;
  memcpy(pOrderStatus,pbindvars, sizeof(OrderStatusData));

  for (ii=0; ii < pOrderStatus->o_ol_cnt; ii++)
    {
      pOrderStatus->s_ol[ii].ol_supply_w_id = otemp-
>loc_ol_supply_w_id[ii];
      pOrderStatus->s_ol[ii].ol_i_id = otemp->loc_ol_i_id[ii];
      pOrderStatus->s_ol[ii].ol_quantity = otemp-
>loc_ol_quantity[ii];
      pOrderStatus->s_ol[ii].ol_amount = otemp-
>loc_ol_amount[ii]/100.0;
      pOrderStatus->s_ol[ii].ol_delivery_d.day =
          atoi(&(otemp->ol_delivery_date_str[ii][0]));
      pOrderStatus->s_ol[ii].ol_delivery_d.month =
          atoi(&(otemp->ol_delivery_date_str[ii][3]));
      pOrderStatus->s_ol[ii].ol_delivery_d.year =
          atoi(&(otemp->ol_delivery_date_str[ii][6]));
    };


  pOrderStatus->c_balance = pOrderStatus->c_balance/100.0;
  pOrderStatus->o_entry_d.day = atoi(&(otemp->entry_date_str[0]));
  pOrderStatus->o_entry_d.month = atoi(&(otemp-
>entry_date_str[3]));
  pOrderStatus->o_entry_d.year = atoi(&(otemp->entry_date_str[6]));
  pOrderStatus->o_entry_d.hour = atoi(&(otemp-
>entry_date_str[11]));
  pOrderStatus->o_entry_d.minute = atoi(&(otemp-
>entry_date_str[14]));
  pOrderStatus->o_entry_d.second = atoi(&(otemp-
>entry_date_str[17]));

  if (status == RECOVERR) return ERR_DB_DEADLOCK_LIMIT;
  else return (status);

}



/* FUNCTION: int TPCCDeliveryDB( CallersContext *pCC, int
iConnectionID,
   * int iSyncID, DBContext *pdbContext,
   * int deadlock_retry, pDeliveryData pDelivery )
   *
   * PURPOSE:       This function writes the delivery information
to the
   *            delivery pipe. The information is sent as a long.
   *
```

```
   * ARGUMENTS:     CallersContext      *pCC
passed in structure
   *                                              pointer from
inetsrv.
   *                int                 iTermId
terminal id of browser
   *                int                 iSyncId
sync id of browser
   *                OraContext          *dbproc
connection db process id
   *                int                 deadlock_retry
deadlock retry count
   *                DeliveryData        *pDelivery
pointer to Delivery data
   *                                              input/output
structure
   *
   * RETURNS:       int       ERR_DB_SUCCESS               success
   *                          ERR_DB_DEADLOCK_LIMIT        max
deadlocked reached
   *                          ERR_DB_NOT_COMMITED other error
   *
   * COMMENTS:      The pipe is initially created with 16K buffer
size this
   *            should allow for up to 4096 deliveries
   *                to be queued before an overflow condition would
occur.
   *                The only reason that an overflow would occur is
if the delivery
   *                application stopped listening while deliveries
were being
   *            posted.
   *
   */

int TPCCDeliveryDB( OraContext *dbproc, pDeliveryData pDeliveryData
)
{


  int retries = 0;
  int status;
  DeliveryData *pbindvars;

  pbindvars = &dbproc->bindvars.info.delivery;
  memcpy(pbindvars, pDeliveryData, sizeof(DeliveryData));

  for (retries = 0, status = RECOVERR;
       retries < DEADLOCKRETRIES &&status == RECOVERR; retries++){

    status = tkvcd(pDeliveryData, dbproc);
  }

  if(status == RECOVERR) return ERR_DB_DEADLOCK_LIMIT;
  else return (status);


}

int TPCCGetLastDBErrorDB(OraContext *dbproc)
{


  /* Add Oracle specific code */

  return ERR_DB_SUCCESS;

}


/* FUNCTION: int TPCCCheckpointDB(CallersContext *pCC, int iTermId,
int iSyncId, OraContext *dbproc, int deadlock_retry, Checkpoint
*pCheckpoint
 *
 * PURPOSE:         This function does a checkpoinnt transaction.
 *
 * ARGUMENTS:       CallersContext      *pCC                passed
in structure pointer
 *                                                          from
inetsrv.
 *                int                 iTermId
         terminal id of browser
 *                int                 iSyncId             sync id
of browser
 *                OraContext          *dbproc
         connection db process id
 *                Checkpoint          *Checkpoint         pointer
to Checkpoint data
 *                int                 deadlock_retry
         deadlock retry count
 *
 * RETURNS:         int       ERR_DB_DEADLOCK_LIMIT         max
deadlock reached
 *                          ERR_DB_NOT_COMMITED No orders found
for customer
 *                          ERR_DB_SUCCESS
         Transaction successfull
 *
 * COMMENTS:        None
 *
 */


#define CHECKPOINT_TXT "alter system switch logfile"

int TPCCCheckpointDB (OraContext *dbproc, pCheckpointData
pCheckpoint ) {

  text stmbuf[100];

  OCIHandleAlloc(dbproc->tpcenv, (dvoid **)&(dbproc->curi),
OCI_HTYPE_STMT,
                 0, (dvoid**)0);
  sprintf ((char *) stmbuf, CHECKPOINT_TXT);
  OCIERROR(dbproc, OCIStmtPrepare(dbproc->curi, dbproc->errhp,
stmbuf,
                                  strlen((char *)stmbuf),
                                  OCI_NTV_SYNTAX,
OCI_DEFAULT));
  if (RECOVERR != OCIERROR(dbproc,
                           OCIStmtExecute(dbproc->tpcsvc,
dbproc->curi,
                                          dbproc->errhp,
1, 0, 0, 0,
                                          OCI_DEFAULT)))
    return (ERR_DB_ERROR);
  OCIHandleFree(dbproc->curi, OCI_HTYPE_STMT);

  return ERR_DB_SUCCESS;

}

/* FUNCTION: int ORAReadRegistrySettings(void)
 *
 * PURPOSE:    This function reads the registry for initialization
information.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:    int  ERR_SUCCESS                   Reqistry read
Successful
 *
 *
 * COMMENTS:   None
 */

int ORAReadRegistrySettings(void)
{
  HKEY    hKey;
  DWORD   size;
  DWORD   type;
  char    szTmp[FILENAMESIZE];
  int     status;

  status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                        0, KEY_READ, &hKey);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_TPCC_KEY;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "OracleOPS", 0, &type, szTmp,
&size);
  if ( status == ERROR_SUCCESS )
    if ( 0 == stricmp(szTmp, "TRUE") || 0 == strcmp(szTmp, "1"))
      bOracleOPS = TRUE;


  RegCloseKey(hKey);

  return ERR_SUCCESS;
}


--------------------------
oracle_db8.h
--------------------------
/*+ file: oracle_db8.h based on Oracle file tpccpl.h
/*+================================================================
=+
 |        Copyright (c) 1994  Oracle Corp, Redwood Shores, CA
|
 |                    OPEN SYSTEMS PERFORMANCE GROUP
|
 |                         All Rights Reserved
|

+================================================================
+
 | DESCRIPTION
 |    header file for the TPC-C transactions.

+================================================================-
*/
/*+*****************************************************************
*********-*/
/*+
-*/
/*+  COPYRIGHT (c) 1998 BY
-*/
/*+  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
-*/
/*+  ALL RIGHTS RESERVED.
-*/
/*+
-*/
/*+  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED  -*/
/*+  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE  -*/
/*+  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER  -*/
/*+  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY  -*/
/*+  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY  -*/
```

```
/*+  TRANSFERRED.
-*/
/*+
-*/
/*+  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE  -*/
/*+  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT  -*/
/*+  CORPORATION.
-*/
/*+
-*/
/*+  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS  -*/
/*+  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
-*/
/*+
-*/
/*+
-*/
/*****************************************************************
*********-*/
#ifndef ORACLE_DB_H
#define ORACLE_DB_H



#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 */


#define RECOVERR -10
#define IRRECERR -20
#define NO_COMMIT -30
#define NOERR 111

#define DEADLOCKWAIT 10


#if (defined(__osf__) && defined(__alpha))
#define HDA_SIZ 512
#else
#define HDA_SIZ 256
#endif


#define MSG_SIZ 512
#define DATE_SIZ 20  /* DD-MM-YYYY.HH:MI:SS  plus null terminator
*/
#define NITEMS 15
#define NDISTS 10
#define ROWIDLEN 20
#define OCIROWLEN 20
#define DEL_DATE_LEN 7
#define SQL_BUF_SIZE 8192

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

/* global variables */
#ifdef ORACLE_DB_C
BOOL bOracleOPS = FALSE;
#else
extern bOracleOPS;
#endif /* ORACLE_DB_C */


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

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


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

   OCIBind *olamt_bp;

   ub2 cons_len[NDISTS];
   ub2 w_id_len[NDISTS];
   ub2 d_id_len[NDISTS];
   ub4 c_id_len[NDISTS];
   ub4 del_o_id_len[NDISTS];
   ub2 del_date_len[NDISTS];
   ub2 carrier_id_len[NDISTS];
   ub2 amt_len[NDISTS];
   ub2 no_rowid_len[NDISTS];
   ub2 no_rowid_ptr_len[NDISTS];
   ub2 o_rowid_len[NDISTS];
   ub2 o_rowid_ptr_len[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   ub2 inum_len;
#endif

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

   int cons[NDISTS];
   int w_id[NDISTS];
   int d_id[NDISTS];
   int c_id[NDISTS];
   int del_o_id[NDISTS];
   int carrier_id[NDISTS];
   int amt[NDISTS];
   ub4 del_o_id_rcnt;
   OCIRowid *no_rowid_ptr[NDISTS];
   OCIRowid *o_rowid_ptr[NDISTS];
   OCIDate del_date[NDISTS];
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   char inum[10];
#endif
   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];
   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 _amtctx {
  int ol_amt[NDISTS][NITEMS];
  sb2 ol_amt_ind[NDISTS][NITEMS];
  ub4 ol_amt_len[NDISTS][NITEMS];
  ub2 ol_amt_rcode[NDISTS][NITEMS];
  int ol_cnt[NDISTS];
};
typedef struct _amtctx amtctx;



struct _newctx {
  sb2 nol_i_id_ind[NITEMS];
  sb2 nol_supply_w_id_ind[NITEMS];
  sb2 nol_quantity_ind[NITEMS];
  sb2 nol_amount_ind[NITEMS];
  sb2 i_name_ind[NITEMS];
  sb2 s_quantity_ind[NITEMS];
  sb2 i_price_ind[NITEMS];
  sb2 ol_w_id_ind[NITEMS];
```

```
sb2 ol_d_id_ind[NITEMS];
sb2 ol_o_id_ind[NITEMS];
sb2 ol_number_ind[NITEMS];
sb2 cons_ind[NITEMS];
sb2 s_rowid_ind[NITEMS];
sb2 s_remote_ind[NITEMS];
sb2 s_quant_ind[NITEMS];
sb2 i_data_ind[NITEMS];
sb2 s_data_ind[NITEMS];
sb2 s_dist_info_ind[NITEMS];
sb2 ol_dist_info_ind[NITEMS];
sb2 null_date_ind[NITEMS];
sb2 s_bg_ind[NITEMS];

ub2 nol_i_id_len[NITEMS];
ub2 nol_supply_w_id_len[NITEMS];
ub2 nol_quantity_len[NITEMS];
ub2 nol_amount_len[NITEMS];
ub2 i_name_len[NITEMS];
ub2 s_quantity_len[NITEMS];
ub2 i_price_len[NITEMS];
ub2 ol_w_id_len[NITEMS];
ub2 ol_d_id_len[NITEMS];
ub2 ol_o_id_len[NITEMS];
ub2 ol_number_len[NITEMS];
ub2 cons_len[NITEMS];
ub2 s_rowid_len[NITEMS];
ub2 s_remote_len[NITEMS];
ub2 s_quant_len[NITEMS];
ub2 i_data_len[NITEMS];
ub2 s_data_len[NITEMS];
ub2 s_dist_info_len[NITEMS];
ub2 ol_dist_info_len[NITEMS];
ub2 null_date_len[NITEMS];
sb2 s_bg_len[NITEMS];

ub2 nol_i_id_rcode[NITEMS];
ub2 nol_supply_w_id_rcode[NITEMS];
ub2 nol_quantity_rcode[NITEMS];
ub2 nol_amount_rcode[NITEMS];
ub2 i_name_rcode[NITEMS];
ub2 s_quantity_rcode[NITEMS];
ub2 i_price_rcode[NITEMS];
ub2 ol_w_id_rcode[NITEMS];
ub2 ol_d_id_rcode[NITEMS];
ub2 ol_o_id_rcode[NITEMS];
ub2 ol_number_rcode[NITEMS];
ub2 cons_rcode[NITEMS];
ub2 s_rowid_rcode[NITEMS];
ub2 s_remote_rcode[NITEMS];
ub2 s_quant_rcode[NITEMS];
ub2 i_data_rcode[NITEMS];
ub2 s_data_rcode[NITEMS];
ub2 s_dist_info_rcode[NITEMS];
ub2 ol_dist_info_rcode[NITEMS];
ub2 null_date_rcode[NITEMS];
sb2 s_bg_rcode[NITEMS];

int ol_w_id[NITEMS];
int ol_d_id[NITEMS];
int ol_o_id[NITEMS];
int ol_number[NITEMS];
int cons[NITEMS];

OCIRowid  *s_rowid_ptr[NITEMS];

int s_remote[NITEMS];
char i_data[NITEMS][51];
char s_data[NITEMS][51];
char s_dist_info[NITEMS][25];

OCIDate null_date[NITEMS]; /* base date for null date entry */
OCIStmt *curn1;
OCIStmt *curn2;
OCIStmt *curn3[10];
OCIStmt *curn4;
OCIBind *i_price_bp;
OCIBind *i_name_bp;
OCIBind *s_bg_bp;
OCIBind *s_data_bp;
OCIBind *i_data_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;
ub4 s_data_count;
ub4 i_data_count;
OCIBind *ol_i_id_bp4;
OCIBind *ol_supply_w_id_bp4;
OCIBind *ol_quantity_bp4;
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_i_id_bp;
OCIBind *ol_supply_w_id_bp;
OCIBind *s_quantity_bp;
OCIBind *s_rowid_bp;
OCIBind *ol_quantity_bp;
OCIBind *s_remote_bp;
OCIBind *id_bp[10][15];
OCIBind *sd_bp[10][15];
OCIDefine *Dcons[10];
OCIDefine *Ds_rowid[10];
OCIDefine *Di_price[10];
OCIDefine *Di_data[10];
OCIDefine *Ds_dist_info[10];
OCIDefine *Ds_data[10];
OCIDefine *Ds_quantity[10];
OCIDefine *Di_name[10];
OCIBind *ol_o_id_bp;
OCIBind *ol_d_id_bp;
OCIBind *ol_w_id_bp;
OCIBind *ol_number_bp;
OCIBind *ol_amount_bp;
OCIBind *ol_dist_info_bp;
OCIBind *null_date_bp;
sb2 w_id_ind;
ub2 w_id_len;
ub2 w_id_rc;

sb2 d_id_ind;
ub2 d_id_len;
ub2 d_id_rc;

sb2 c_id_ind;
ub2 c_id_len;
ub2 c_id_rc;

sb2 o_all_local_ind;
ub2 o_all_local_len;
ub2 o_all_local_rc;

sb2 o_ol_cnt_ind;
ub2 o_ol_cnt_len;
ub2 o_ol_cnt_rc;

sb2 w_tax_ind;
ub2 w_tax_len;
ub2 w_tax_rc;

sb2 d_tax_ind;
ub2 d_tax_len;
ub2 d_tax_rc;

sb2 o_id_ind;
ub2 o_id_len;
ub2 o_id_rc;

sb2 c_discount_ind;
ub2 c_discount_len;
ub2 c_discount_rc;

sb2 c_credit_ind;
ub2 c_credit_len;
ub2 c_credit_rc;

sb2 c_last_ind;
ub2 c_last_len;
ub2 c_last_rc;

sb2 retries_ind;
ub2 retries_len;
ub2 retries_rc;

sb2 cr_date_ind;
ub2 cr_date_len;
ub2 cr_date_rc;


int cs;
int norow;

/* context holders */
int i_name_ctx;
int i_data_ctx;
int i_price_ctx;
int s_data_ctx;
int s_dist_info_ctx;
int s_quantity_ctx;

};
typedef struct _newctx newctx;

struct _ordctx {
   sb2 c_rowid_ind[100];
   sb2 ol_supply_w_id_ind[NITEMS];
   sb2 ol_i_id_ind[NITEMS];
   sb2 ol_quantity_ind[NITEMS];
   sb2 ol_amount_ind[NITEMS];
   sb2 ol_delivery_d_ind[NITEMS];
   sb2 ol_w_id_ind;
   sb2 ol_d_id_ind;
   sb2 ol_o_id_ind;
   sb2 c_id_ind;
   sb2 c_first_ind;
   sb2 c_middle_ind;
   sb2 c_balance_ind;
   sb2 c_last_ind;
   sb2 o_id_ind;
   sb2 o_entry_d_ind;
   sb2 o_carrier_id_ind;
```

```
   sb2 o_ol_cnt_ind;

   ub4 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;

   ub2 c_rowid_rcode[100];
   ub2 ol_supply_w_id_rcode[NITEMS];
   ub2 ol_i_id_rcode[NITEMS];
   ub2 ol_quantity_rcode[NITEMS];
   ub2 ol_amount_rcode[NITEMS];
   ub2 ol_delivery_d_rcode[NITEMS];
   ub2 ol_w_id_rcode;
   ub2 ol_d_id_rcode;
   ub2 ol_o_id_rcode;

   ub4 ol_supply_w_id_csize;
   ub4 ol_i_id_csize;
   ub4 ol_quantity_csize;
   ub4 ol_amount_csize;
   ub4 ol_delivery_d_csize;
   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 *w_id_bp0;
   OCIBind *w_id_bp2;
   OCIBind *w_id_bp3;
   OCIBind *w_id_bp4;
   OCIBind *d_id_bp0;
   OCIBind *d_id_bp2;
   OCIBind *d_id_bp3;
   OCIBind *d_id_bp4;
   OCIBind *c_id_bp;
   OCIBind *byln_bp;
   OCIBind *c_last_bp;
   OCIBind *c_last_bp4;
   OCIBind *c_first_bp;
   OCIBind *c_middle_bp;
   OCIBind *c_balance_bp;
   OCIBind *o_id_bp;
   OCIBind *o_entry_d_bp;
   OCIBind *o_cr_id_bp;
   OCIBind *o_ol_cnt_bp;
   OCIBind *ol_s_w_id_bp;
   OCIBind *ol_i_id_bp;
   OCIBind *ol_quantity_bp;
   OCIBind *ol_amount_bp;
   OCIBind *ol_d_d_bp;
   OCIBind *c_rowid_bp;
   OCIDefine *c_rowid_dp;
   OCIDefine *c_last_dp;
   OCIDefine *c_last_dp1;
   OCIDefine *c_id_dp;
   OCIDefine *c_id_dp1;
   OCIDefine *c_first_dp1;
   OCIDefine *c_first_dp2;
   OCIDefine *c_middle_dp1;
   OCIDefine *c_middle_dp2;
   OCIDefine *c_balance_dp1;
   OCIDefine *c_balance_dp2;
   OCIDefine *o_id_dp1;
   OCIDefine *o_id_dp2;
   OCIDefine *o_entry_d_dp1;
   OCIDefine *o_entry_d_dp2;
   OCIDefine *o_cr_id_dp1;
   OCIDefine *o_cr_id_dp2;
   OCIDefine *o_ol_cnt_dp1;
   OCIDefine *o_ol_cnt_dp2;
   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 *middle_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;
  OCIBind *w_id_bp1;
  sb2 w_id_ind;
  ub2 w_id_len;
  ub2 w_id_rc;

  OCIBind *d_id_bp;
  OCIBind *d_id_bp1;
  sb2 d_id_ind;
  ub2 d_id_len;
  ub2 d_id_rc;

  OCIBind *c_w_id_bp;
  OCIBind *c_w_id_bp1;
  sb2 c_w_id_ind;
  ub2 c_w_id_len;
  ub2 c_w_id_rc;

  OCIBind *c_d_id_bp;
  OCIBind *c_d_id_bp1;
  sb2 c_d_id_ind;
  ub2 c_d_id_len;
  ub2 c_d_id_rc;

  OCIBind *c_id_bp;
  OCIBind *c_id_bp1;
  sb2 c_id_ind;
  ub2 c_id_len;
  ub2 c_id_rc;

  OCIBind *h_amount_bp;
  OCIBind *h_amount_bp1;
  sb2 h_amount_ind;
  ub2 h_amount_len;
  ub2 h_amount_rc;

  OCIBind *c_last_bp;
  OCIBind *c_last_bp1;
  sb2 c_last_ind;
  ub2 c_last_len;
  ub2 c_last_rc;

  OCIBind *w_street_1_bp;
  OCIBind *w_street_1_bp1;
  sb2 w_street_1_ind;
  ub2 w_street_1_len;
  ub2 w_street_1_rc;

  OCIBind *w_street_2_bp;
  OCIBind *w_street_2_bp1;
  sb2 w_street_2_ind;
  ub2 w_street_2_len;
  ub2 w_street_2_rc;

  OCIBind *w_city_bp;
  OCIBind *w_city_bp1;
  sb2 w_city_ind;
  ub2 w_city_len;
  ub2 w_city_rc;

  OCIBind *w_state_bp;
  OCIBind *w_state_bp1;
  sb2 w_state_ind;
  ub2 w_state_len;
  ub2 w_state_rc;

  OCIBind *w_zip_bp;
  OCIBind *w_zip_bp1;
  sb2 w_zip_ind;
  ub2 w_zip_len;
  ub2 w_zip_rc;

  OCIBind *d_street_1_bp;
  OCIBind *d_street_1_bp1;
  sb2 d_street_1_ind;
  ub2 d_street_1_len;
  ub2 d_street_1_rc;

  OCIBind *d_street_2_bp;
  OCIBind *d_street_2_bp1;
  sb2 d_street_2_ind;
  ub2 d_street_2_len;
  ub2 d_street_2_rc;

  OCIBind *d_city_bp;
  OCIBind *d_city_bp1;
  sb2 d_city_ind;
  ub2 d_city_len;
  ub2 d_city_rc;

  OCIBind *d_state_bp;
  OCIBind *d_state_bp1;
  sb2 d_state_ind;
  ub2 d_state_len;
  ub2 d_state_rc;

  OCIBind *d_zip_bp;
  OCIBind *d_zip_bp1;
  sb2 d_zip_ind;
  ub2 d_zip_len;
  ub2 d_zip_rc;

  OCIBind *c_first_bp;
  OCIBind *c_first_bp1;
  sb2 c_first_ind;
  ub2 c_first_len;
  ub2 c_first_rc;
```

```
  OCIBind *c_middle_bp;
  OCIBind *c_middle_bp1;
  sb2 c_middle_ind;
  ub2 c_middle_len;
  ub2 c_middle_rc;

  OCIBind *c_street_1_bp;
  OCIBind *c_street_1_bp1;
  sb2 c_street_1_ind;
  ub2 c_street_1_len;
  ub2 c_street_1_rc;

  OCIBind *c_street_2_bp;
  OCIBind *c_street_2_bp1;
  sb2 c_street_2_ind;
  ub2 c_street_2_len;
  ub2 c_street_2_rc;

  OCIBind *c_city_bp;
  OCIBind *c_city_bp1;
  sb2 c_city_ind;
  ub2 c_city_len;
  ub2 c_city_rc;

  OCIBind *c_state_bp;
  OCIBind *c_state_bp1;
  sb2 c_state_ind;
  ub2 c_state_len;
  ub2 c_state_rc;

  OCIBind *c_zip_bp;
  OCIBind *c_zip_bp1;
  sb2 c_zip_ind;
  ub2 c_zip_len;
  ub2 c_zip_rc;

  OCIBind *c_phone_bp;
  OCIBind *c_phone_bp1;
  sb2 c_phone_ind;
  ub2 c_phone_len;
  ub2 c_phone_rc;

  OCIBind *c_since_bp;
  OCIBind *c_since_bp1;
  sb2 c_since_ind;
  ub2 c_since_len;
  ub2 c_since_rc;

  OCIBind *c_credit_bp;
  OCIBind *c_credit_bp1;
  sb2 c_credit_ind;
  ub2 c_credit_len;
  ub2 c_credit_rc;

  OCIBind *c_credit_lim_bp;
  OCIBind *c_credit_lim_bp1;
  sb2 c_credit_lim_ind;
  ub2 c_credit_lim_len;
  ub2 c_credit_lim_rc;

  OCIBind *c_discount_bp;
  OCIBind *c_discount_bp1;
  sb2 c_discount_ind;
  ub2 c_discount_len;
  ub2 c_discount_rc;

  OCIBind *c_balance_bp;
  OCIBind *c_balance_bp1;
  sb2 c_balance_ind;
  ub2 c_balance_len;
  ub2 c_balance_rc;

  OCIBind *c_data_bp;
  OCIBind *c_data_bp1;
  sb2 c_data_ind;
  ub2 c_data_len;
  ub2 c_data_rc;

  OCIBind *h_date_bp;
  OCIBind *h_date_bp1;
  sb2 h_date_ind;
  ub2 h_date_len;
  ub2 h_date_rc;

  OCIBind *retries_bp;
  OCIBind *retries_bp1;
  sb2 retries_ind;
  ub2 retries_len;
  ub2 retries_rc;

  OCIBind *cr_date_bp;
  OCIBind *cr_date_bp1;
  sb2 cr_date_ind;
  ub2 cr_date_len;
  ub2 cr_date_rc;

  OCIBind *byln_bp;
  sb2 byln_ind;
  ub2 byln_len;
  ub2 byln_rc;
};
typedef struct _payctx payctx;

struct _stoctx {
  OCIStmt *curs;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *threshold_bp;
  OCIDefine *low_stock_bp;
  int norow;
};
typedef struct _stoctx stoctx;


typedef struct _deltemp {
  char cvtcr_date[DATE_SIZ];
  OCIDate cr_date;
} deltemp;

typedef struct _newtemp {
  char entry_date[DATE_SIZ + 1];
  OCIDate cr_date;
  int nol_i_id[MAX_OL];
  int nol_supply_w_id[MAX_OL];
  int nol_quantity[MAX_OL];
  char i_name[MAX_OL][25];
  int s_quantity[MAX_OL];
  int i_price[MAX_OL];
  int nol_amount[MAX_OL];
  char brand_generic[MAX_OL];
  float c_discount;
  float w_tax;
  float d_tax;
  int n_retry;
} newtemp;

typedef struct _ordtemp {
  OCIDate entry_date;
  char entry_date_str[DATE_SIZ + 1];
  int loc_ol_i_id[MAX_OL];
  int loc_ol_supply_w_id[MAX_OL];
  int loc_ol_quantity[MAX_OL];
  int loc_ol_amount[MAX_OL];
  OCIDate loc_ol_delivery_date[MAX_OL];
  char ol_delivery_date_str[MAX_OL][11];
} ordtemp;

typedef struct _paytemp {
  char h_date[DATE_SIZ];
  OCIDate customer_sdate;
  char c_since_str[11];
  OCIDate cr_date;
  float c_discount;
  int h_amount;
  int c_credit_lim;
  int p_retry;
} paytemp;


typedef struct _oracontext {
  /*  handles for talking to Oracle */
  OCIEnv *tpcenv;
  OCIServer *tpcsrv;
  OCIError *errhp;
  OCIError *datecvterrhp;
  OCISvcCtx *tpcsvc;
  OCISession *tpcusr;
  OCIStmt *curi;
  dvoid *xmem;
  int del_init;
  int new_init;
  int pay_init;
  int ord_init;
  int sto_init;
  /* data areas where cursors will find data */
  TransactionData   bindvars;
  /* oracle structures for bind data information during a
transaction */
  ordctx octx;
  delctx dctx;
  delctx dctx2;
  newctx nctx;
  payctx pctx;
  stoctx sctx;
  defctx cbctx;
  amtctx actx;

  union {
    deltemp del;
    newtemp new;
    ordtemp ord;
    paytemp pay;
  } tempvars;
} OraContext;


#define OCIERROR(p,function)\
        ocierror(__FILE__,__LINE__,(p),(function))

#define OCIBND(stmp, bndp, p, sqlvar, progv, progvl, ftype)\
       ocierror(__FILE__,__LINE__, (p), \
             OCIBindByName((stmp), &(bndp), (p->errhp), \
                           (text *)(sqlvar), strlen((sqlvar)),\
                           (progv), (progvl),
(ftype),0,0,0,0,0,OCI_DEFAULT))

#define
OCIBNDRA(stmp,bndp,p,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
       ocierror(__FILE__,__LINE__,(p), \
            OCIBindByName((stmp),&(bndp),(p->errhp),(text
*)(sqlvar),strlen((sqlvar)),\

(progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT))
```

```
#define
OCIBNDRAD(stmp,bndp,p,sqlvar,progvl,ftype,indp,ctxp,cbf_nodata,cbf_
data) \
       ocierror(__FILE__,__LINE__,(p), \
            OCIBindByName((stmp),&(bndp),(p->errhp),(text
*)(sqlvar), \
                          strlen((sqlvar)),0,(progvl),(ftype), \
                          indp,0,0,0,0,OCI_DATA_AT_EXEC)); \
       ocierror(__FILE__,__LINE__,(p), \
            OCIBindDynamic((bndp),(p-
>errhp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)))



#define
OCIBNDR(stmp,bndp,p,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
       ocierror(__FILE__,__LINE__,(p), \
          OCIBindByName((stmp),&(bndp),(p->errhp),(text
*)(sqlvar),strlen((sqlvar)),\

(progv),(progvl),(ftype),(indp),(alen),(arcode),0,0,OCI_DEFAULT))


#define
OCIBNDRAA(stmp,bndp,p,sqlvar,progv,progvl,ftype,indp,alen,arcode,ms
,cu) \
       ocierror(__FILE__,__LINE__,(p), \
             OCIBindByName((stmp), &(bndp), (p->errhp), \
                           (text *)(sqlvar), strlen((sqlvar)),\
                           (progv), (progvl),
(ftype),(indp),(alen),(arcode),\
                          (ms),(cu),OCI_DEFAULT))



#define OCIDEFINE(stmp,dfnp,errp,pos,progv,progvl,ftype)\

OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),(ftype)
,\
                           0,0,0,OCI_DEFAULT)


#define OCIDEF(stmp,dfnp,errp,pos,progv,progvl,ftype) \

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) \
       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)));



/* prototypes */
extern int tkvcninit (NewOrderData *pNew,
                          OraContext *p);

extern int tkvcn (NewOrderData *pNew, OraContext *p);

extern void tkvcndone (newctx *pnctx);

extern int tkvcpinit (PaymentData *pPay,
                          OraContext *p);

extern int tkvcp (PaymentData *pPay, OraContext *p);

extern void tkvcpdone (payctx *ppctx);

extern int tkvcoinit(OrderStatusData *pOrd,
                          OraContext *p);

extern int tkvco (OrderStatusData *pOrd, OraContext *p);

extern void tkvcodone (ordctx *poctx);

extern int tkvcsinit(StockLevelData *pOrd,
                          OraContext *p);

extern int tkvcs (OraContext *p);

extern void tkvcsdone (stoctx *psctx);


extern int tkvcdinit (DeliveryData *pDel,
                          OraContext *p);

extern int tkvcd (DeliveryData *pDel, OraContext *p);

extern void tkvcddone (delctx *pdctx);


int ocierror(char *fname, int lineno, OraContext *p, sword status);
extern int ErrRpt(Lda_Def *pLda, int rc);
void TPCCErr( char *fmt, ...);
void TPCCLog( char *fmt, ...);



#endif /* ORACLE_DB_H */


--------------------------
oracle_txns8.c
--------------------------
/*+ file: oracle_txns8.c  based on Oracle files - plpay.c plnew.c
plord.c
                                                      pldel.c plsto.c
-*/
/*+=============================================================
=+
 |        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA
|
 |                    OPEN SYSTEMS PERFORMANCE GROUP
|
 |                         All Rights Reserved
|

+=============================================================
+
 | DESCRIPTION
 |    OCI version (using PL/SQL stored procedure) of
 |    PAYMENT transaction in TPC-C benchmark.
 |    OCI version (using PL/SQL stored procedure) of
 |    NEW ORDER transaction in TPC-C benchmark.
 |    OCI version (using PL/SQL anynomous block) of
 |    ORDER STATUS transaction in TPC-C benchmark.
 |    OCI version of DELIVERY transaction in TPC-C benchmark.
 |    OCI version of STOCK LEVEL transaction in TPC-C benchmark.

+=============================================================-
*/
/*+***********************************************************
**********
 *
*
 *  COPYRIGHT (c) 1998 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

/*+
 * Abstract: This file contains the transaction routines for
connection
 *           to the oracle database - for the tpcc benchmark.
 *
 * Modified history:
 *
 *
 */

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

#include <oci.h>
#include <ocidfn.h>
#include <ociapr.h>
```

```
#ifdef _WIN32
#include <windows.h>
#endif

#include <tpccerr.h>
#include <tpccstruct.h>
#include <oracle_db8.h>

#ifdef OL_CHECK
# include <httpext.h>
extern int iMaxWareHouses;
#endif

/* prototypes */
int getfile(char *filename, text *filebuf);


void vgetdate (unsigned char *oradt)
{
 struct tm *loctime;
 time_t  int_time;

 struct ORADATE {
 unsigned char      century;
 unsigned char      year;
 unsigned char      month;
 unsigned char      day;
 unsigned char      hour;
 unsigned char      minute;
 unsigned char      second;
 } Date;
int century;
int cnvrtOK;

/* assume convert is successful */
cnvrtOK = 1;

/* get the current date and time as an integer */
time( &int_time);

/* Convert the current date and time into local time */
loctime = localtime( &int_time);

century = (1900+loctime->tm_year) / 100;

Date.century = (unsigned char)(century + 100);
if (Date.century < 119 || Date.century > 120) cnvrtOK = 0;
Date.year  = (unsigned char)(loctime->tm_year%100+100);
if (Date.year < 100 || Date.year > 199) cnvrtOK = 0;
Date.month = (unsigned char)(loctime->tm_mon + 1);
if (Date.month < 1 || Date.month > 12) cnvrtOK = 0;
Date.day   = (unsigned char)loctime->tm_mday;
if (Date.day < 1 || Date.day > 31) cnvrtOK = 0;
Date.hour  = (unsigned char)(loctime->tm_hour + 1);
if (Date.hour < 1 || Date.hour > 24) cnvrtOK = 0;
Date.minute= (unsigned char)(loctime->tm_min + 1);
if (Date.minute < 1 || Date.minute > 60) cnvrtOK = 0;
Date.second= (unsigned char)(loctime->tm_sec + 1);
if (Date.second < 1 || Date.second > 60) cnvrtOK = 0;


if (cnvrtOK)
  memcpy(oradt,&Date,7);
  else
  *oradt = '\0';

 return;

}
void cvtdmy (unsigned char *oradt, char *outdate)
{

        struct ORADATE {
                unsigned char   century;
                unsigned char   year;
                unsigned char   month;
                unsigned char   day;
                unsigned char   hour;
                unsigned char   minute;
                unsigned char   second;
        } Date;


        int day,month,year;

        memcpy(&Date,oradt,7);

        year = (Date.century-100)*100 + Date.year-100;
        month = Date.month;
        day = Date.day;
        sprintf(outdate,"%02d-%02d-%4d",day,month,year);

        return;
}

void cvtdmyhms (unsigned char *oradt, char *outdate)
{

        struct ORADATE {
                unsigned char   century;
                unsigned char   year;
                unsigned char   month;
                unsigned char   day;
                unsigned char   hour;
                unsigned char   minute;
                unsigned char   second;
        } Date;


        int day,month,year;
        int hour,min,sec;

        memcpy(&Date,oradt,7);

        year = (Date.century-100)*100 + Date.year-100;
        month = Date.month;
        day = Date.day;
        hour = Date.hour - 1;
        min = Date.minute - 1;
        sec = Date.second - 1;

        sprintf(outdate,"%02d-%02d-%4d %02d:%02d:%02d",
                        day,month,year,hour,min,sec);

        return;
}

swapitemstock (int i, int j, int *i_price, char i_name[][25],
                        int *s_quantity, newctx *nctx)
{

   int tempi;
   int tempf;
   char tempstr[52];
   ub2 tempub2;
   sb2 tempsb2;
   OCIRowid  *tmprid;

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

   tempsb2 = nctx->s_rowid_ind[i];
   nctx->s_rowid_ind[i] = nctx->s_rowid_ind[j];
   nctx->s_rowid_ind[j] = tempsb2;
   tempub2 = nctx->s_rowid_len[i];
   nctx->s_rowid_len[i] = nctx->s_rowid_len[j];
   nctx->s_rowid_len[j] = tempub2;
   tempub2 = nctx->s_rowid_rcode[i];
   nctx->s_rowid_rcode[i] = nctx->s_rowid_rcode[j];
   nctx->s_rowid_rcode[j] = tempub2;
   tmprid = nctx->s_rowid_ptr[i];
   nctx->s_rowid_ptr[i]= nctx->s_rowid_ptr[j];
   nctx->s_rowid_ptr[j]=tmprid;

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

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

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

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

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

   tempsb2 = nctx->s_data_ind[i];
   nctx->s_data_ind[i] = nctx->s_data_ind[j];
   nctx->s_data_ind[j] = tempsb2;
   tempub2 = nctx->s_data_len[i];
   nctx->s_data_len[i] = nctx->s_data_len[j];
   nctx->s_data_len[j] = tempub2;
   tempub2 = nctx->s_data_rcode[i];
   nctx->s_data_rcode[i] = nctx->s_data_rcode[j];
   nctx->s_data_rcode[j] = tempub2;
   strncpy (tempstr, nctx->s_data[i], 51);
   strncpy (nctx->s_data[i], nctx->s_data[j], 51);
   strncpy (nctx->s_data[j], tempstr, 51);
   return 0;
}

/* the arrays are initialized based on a successful select from */
/* stock/item. We need to shift the values in the orderline array
*/
/* one position up to compensate when we have an invalid item */

shiftitemstock (int i, int j, newctx *nctx, OraContext *p)
{
  newtemp *ntemp = &(p->tempvars.new);

  /* shift up the values for the stock table */
  nctx->s_remote[i] =  nctx->s_remote[j];

  /* shift up the order_line values */

  nctx->nol_i_id_ind[i]=nctx->nol_i_id_ind[j];
  ntemp->nol_i_id[i] = ntemp->nol_i_id[j];

  nctx->nol_quantity_ind[i] =  nctx->nol_quantity_ind[j];
  ntemp->nol_quantity[i] = ntemp->nol_quantity[j];

  nctx->nol_supply_w_id_ind [i] =  nctx->nol_supply_w_id_ind[j];
  ntemp->nol_supply_w_id[i] = ntemp->nol_supply_w_id[j];
  return 0;
}

int SelItemStk (NewOrderData *pNew,
                    int *pstatus,
                    int retries, int proc_no, newctx *nctx,
OraContext *p)
{

   int i, j, rpc3,rcount;
   int errcode;
   int execstatus;

#ifdef OL_CHECK
   newtemp *ntemp = &(p->tempvars.new);
   for (i = 0; i < MAX_OL; i++) {
     if((TRUE == nctx->nol_supply_w_id_ind[i]) &&
     ( ntemp->nol_supply_w_id[i] > iMaxWareHouses )) {
       TPCCErr( "Bad supply warehouse ol: %d, s_w_id: %d, query:
%s",
                    i+1, ntemp->nol_supply_w_id[i],
                    ((EXTENSION_CONTROL_BLOCK *)pNew->pCC)-
>lpszQueryString );
     }
   }
#endif

    /* array select from item and stock tables */
   execstatus=OCIStmtExecute(p->tpcsvc,(nctx->curn3)[pNew->d_id-
1],p->errhp,
                                  pNew-
>o_ol_cnt,0,0,0,OCI_DEFAULT);
   if((execstatus != OCI_SUCCESS) && (execstatus != OCI_NO_DATA)) {
     errcode = OCIERROR(p,execstatus);
     if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
     {
     /* In case of NO_DATA this should NOT return, but simply fall
through */
        OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
        return (RECOVERR);
     }
     else
     {
        OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
        return (IRRECERR);
     }
   }
   /* mark invalid items */
   OCIAttrGet((nctx->curn3)[pNew->d_id-1],
OCI_HTYPE_STMT,&rcount,NULL,
               OCI_ATTR_ROW_COUNT, p->errhp);
   rpc3 = rcount;

   /* the result is in order, so we have to shift up to fill    */
   /* the slot for the line with the invalid item.              */

   if ((*pstatus = pNew->o_ol_cnt - rcount) >1)
   {
      TPCCErr ("TPC-C server %d: more than 1 invalid item?\n",
proc_no);
      return (rpc3);
   }
   if (*pstatus == 0) return (rpc3);

   /* find the invalid item, transfer the rowid information */

   for (i = 0; i < pNew->o_ol_cnt; i++) {
     if (nctx->cons[i] != i) break;   /* this item is invalid */
   }

   /* not the last item - shift up */

   for (j = i; j < pNew->o_ol_cnt-1; j++)
   {
     shiftitemstock (j, j+1, nctx, p);
   }
   /* zero the last item */
     i = pNew->o_ol_cnt-1;
     nctx->nol_i_id_ind[i] = NA;
     nctx->nol_supply_w_id_ind[i] = NA;
     nctx->nol_quantity_ind[i] = NA;
     nctx->nol_amount_ind[i] = NA;
     nctx->ol_w_id_ind[i] = NA;
     nctx->ol_d_id_ind[i] = NA;
     nctx->ol_o_id_ind[i] = NA;
     nctx->null_date_ind[i]= NA;
     nctx->ol_number_ind[i] = NA;
     nctx->ol_dist_info_ind[i] = NA;
     nctx->s_remote_ind[i] = NA;
     nctx->s_quant_ind[i] = NA;

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

   return (rpc3);
}


UpdStk (OraContext *p, NewOrderData *pNew, int proc_no)
{
  int rcount;
  int execstatus;
  int errcode;
  newctx *nctx = &(p->nctx);

#ifdef OL_CHECK
  int i;
   newtemp *ntemp = &(p->tempvars.new);
   for (i = 0; i < MAX_OL; i++) {
     if((TRUE == nctx->nol_supply_w_id_ind[i]) &&
     ( ntemp->nol_supply_w_id[i] > iMaxWareHouses )) {
       TPCCErr( "Bad supply warehouse ol in updstk: %d, s_w_id: %d,
query: %s",
                    i+1, ntemp->nol_supply_w_id[i],
                    ((EXTENSION_CONTROL_BLOCK *)pNew->pCC)-
>lpszQueryString );
     }
   }
#endif
   /* array update of stock table */

  execstatus = OCIStmtExecute(p->tpcsvc,nctx->curn2,p->errhp,pNew-
>o_ol_cnt,
                              0,0,0,OCI_DEFAULT);
   if(execstatus != OCI_SUCCESS) {
     OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
     errcode = OCIERROR(p, execstatus);
     if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)) {
         return (RECOVERR);
     } else {
       return (IRRECERR);
     }
   }
  OCIAttrGet(nctx->curn2,OCI_HTYPE_STMT,&rcount,NULL,
           OCI_ATTR_ROW_COUNT, p->errhp);
  if (rcount != (pNew->o_ol_cnt) ) {
     TPCCErr ("Error in TPC-C server %d: array update failed in
UpdStk()\n",
              proc_no);
     OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
     return (IRRECERR);
   }
   return (rcount);
}



UpdStk2 (OraContext *p, NewOrderData *pNew, int status, int
proc_no)
{

  int rcount;
  int execstatus;
  int errcode;
  newctx *nctx = &(p->nctx);
```

```
#ifdef OL_CHECK
  int i;
   newtemp *ntemp = &(p->tempvars.new);
   for (i = 0; i < MAX_OL; i++) {
     if((TRUE == nctx->nol_supply_w_id_ind[i]) &&
     ( ntemp->nol_supply_w_id[i] > iMaxWareHouses )) {
       TPCCErr( "Bad supply warehouse ol in updstk2: %d, s_w_id:
%d, query: %s",
                    i+1, ntemp->nol_supply_w_id[i],
                    ((EXTENSION_CONTROL_BLOCK *)pNew->pCC)-
>lpszQueryString );
     }
   }
#endif
  /* array update of stock table */

  execstatus = OCIStmtExecute(p->tpcsvc,nctx->curn2,p->errhp,
                                        pNew->o_ol_cnt-
status,0,0,0,OCI_DEFAULT);
   if(execstatus != OCI_SUCCESS) {
     OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
     errcode = OCIERROR(p,execstatus);
     if(errcode == NOT_SERIALIZABLE) {
          return (RECOVERR);
     } else if (errcode == RECOVERR) {
          return (RECOVERR);
     } else {
       return (IRRECERR);
     }
   }
   OCIAttrGet(nctx->curn2,OCI_HTYPE_STMT,&rcount,NULL,
OCI_ATTR_ROW_COUNT,
                p->errhp);

   if (rcount != (pNew->o_ol_cnt - status)) {
      TPCCErr("Error in TPC-C server %d: array update failed in
UpdStk2()\n",
                proc_no);
      OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
      return (NO_COMMIT);
   }

   return (rcount);

}


/* stock level transaction */

#define SLSQLTXT "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)"


#define SQLTXTTEST "BEGIN stocklevel.getstocklevel (:w_id, :d_id, \
:threshold); END;"

tkvcsinit (StockLevelData *pSL,
             OraContext *p)

{
  stoctx *sctx = &(p->sctx);
  text stmbuf[SQL_BUF_SIZE];

  sctx->curs = NULL;

  memset(sctx,(char)0,sizeof(stoctx));
  sctx->norow=0;

  OCIERROR(p, OCIHandleAlloc(p->tpcenv,(dvoid**)&(sctx-
>curs),OCI_HTYPE_STMT,0,
                                    (dvoid**)0));
  sprintf ((char *) stmbuf, SLSQLTXT);
  OCIERROR(p,OCIStmtPrepare(sctx->curs,p->errhp,stmbuf,strlen((char
*)stmbuf),
                                  OCI_NTV_SYNTAX,OCI_DEFAULT));
  OCIERROR(p, OCIAttrSet(sctx->curs,OCI_HTYPE_STMT,(dvoid*)&sctx-
>norow,0,
                                OCI_ATTR_PREFETCH_ROWS,p->errhp));

  /* bind variables */

  OCIBND(sctx->curs,sctx->w_id_bp,p, ":w_id", ADR(pSL-
>w_id),sizeof(int),
           SQLT_INT);
  OCIBND(sctx->curs,sctx->d_id_bp,p, ":d_id", ADR(pSL-
>ld_id),sizeof(int),
           SQLT_INT);
  OCIBND(sctx->curs,sctx->threshold_bp,p, ":threshold", ADR(pSL-
>threshold),
           sizeof(int),SQLT_INT);
  OCIDEF(sctx->curs,sctx->low_stock_bp,p->errhp, 1, ADR(pSL-
>low_stock),
      sizeof(int), SQLT_INT);

  return (ERR_DB_SUCCESS);

}

tkvcs (OraContext *p)
{
  stoctx *sctx = &(p->sctx);
  int execstatus = 0;
  int errcode = 0;

  execstatus = OCIStmtExecute(p->tpcsvc,sctx->curs,p-
>errhp,1,0,0,0,
                                        OCI_COMMIT_ON_SUCCESS |
OCI_DEFAULT);
  if(execstatus != OCI_SUCCESS) {
    OCITransCommit(p->tpcsvc,p->errhp,OCI_DEFAULT);
    errcode = OCIERROR(p,execstatus);
    if(errcode == NOT_SERIALIZABLE) {
      return (RECOVERR);
     } else if (errcode == RECOVERR) {
      return (RECOVERR);
     } else if (errcode == SNAPSHOT_TOO_OLD) {
      return (RECOVERR);
     } else {
      TPCCErr("DB error in stocklevel transaction %d\n",errcode);
      return (ERR_DB_ERROR);
     }
  }

  return (ERR_DB_SUCCESS);

}

void tkvcsdone (stoctx *psctx)

{
  stoctx sctx = *psctx;

  if(NULL != sctx.curs)
    OCIHandleFree((dvoid *)sctx.curs,OCI_HTYPE_STMT);

}


#define SQLTXT_PAY_ZERO "BEGIN apayment.adopayment(:w_id, :d_id,
:c_w_id, :c_d_id, \
   :c_id, 0, \
   :h_amount, :c_last, :w_street_1, :w_street_2, :w_city, :w_state,
\
   :w_zip, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip,
:c_first, \
   :c_middle, :c_street_1, :c_street_2, :c_city, :c_state, :c_zip,
:c_phone, \
   :c_since, :c_credit, :c_credit_lim, :c_discount, :c_balance,
:c_data, \
   :h_date, :retry, :cr_date); END;"

#define SQLTXT_PAY_NONZERO "BEGIN apayment.adopayment(:w_id, :d_id,
:c_w_id, :c_d_id, \
   :c_id, 1, \
   :h_amount, :c_last, :w_street_1, :w_street_2, :w_city, :w_state,
\
   :w_zip, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip,
:c_first, \
   :c_middle, :c_street_1, :c_street_2, :c_city, :c_state, :c_zip,
:c_phone, \
   :c_since, :c_credit, :c_credit_lim, :c_discount, :c_balance,
:c_data, \
   :h_date, :retry, :cr_date); END;"

#define SQLTXT_PAY_INIT "BEGIN initpay.pay_init; END;"


tkvcpinit (PaymentData *pPay,
            OraContext *p)

{
  payctx *pctx = &(p->pctx);
  paytemp *ptemp = &(p->tempvars.pay);

  text stmbuf[SQL_BUF_SIZE];

  pctx->curpi = NULL;
  pctx->curp0 = NULL;
  pctx->curp1 = NULL;

  memset(pctx,(char)0,sizeof(payctx));

/* cursor for init */
   OCIERROR(p,OCIHandleAlloc(p->tpcenv, (dvoid **)(&(pctx->curpi)),
            OCI_HTYPE_STMT,0,(dvoid**)0));

   OCIERROR(p,OCIHandleAlloc(p->tpcenv, (dvoid **)(&(pctx->curp0)),
            OCI_HTYPE_STMT,0,(dvoid**)0));
   OCIERROR(p,OCIHandleAlloc(p->tpcenv, (dvoid **)(&(pctx->curp1)),
            OCI_HTYPE_STMT,0,(dvoid**)0));

   /*  build the init statement  and execute it */

   sprintf ((char*)stmbuf, SQLTXT_PAY_INIT);
   OCIERROR(p,OCIStmtPrepare(pctx->curpi, p->errhp, stmbuf,
            strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));
   OCIERROR(p,
     OCIStmtExecute(p->tpcsvc,pctx->curpi,p-
>errhp,1,0,0,0,OCI_DEFAULT));

   /* customer id !=  0, go by customer id */
   if(ERR_DB_ERROR == getfile("paynz.sql",stmbuf))
  {
    TPCCErr("Error opening the file paynz.sql");
    return ERR_DB_ERROR;
  }

   OCIERROR(p,OCIStmtPrepare(pctx->curp0, p->errhp, stmbuf,
```

```
          strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

   /* customer id ==  0, go by last name */
   if(ERR_DB_ERROR == getfile("payz.sql",stmbuf))
  {
    TPCCErr("Error opening the file payz.sql");
    return ERR_DB_ERROR;
  }

   OCIERROR(p,OCIStmtPrepare(pctx->curp1, p->errhp, stmbuf,
          strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));

  pctx->w_id_ind = TRUE;
  pctx->w_id_len = SIZ(pPay->w_id);
  pctx->d_id_ind = TRUE;
  pctx->d_id_len = SIZ(pPay->d_id);
  pctx->c_w_id_ind = TRUE;
  pctx->c_w_id_len = SIZ(pPay->c_w_id);
  pctx->c_d_id_ind = TRUE;
  pctx->c_d_id_len = SIZ(pPay->c_d_id);
  pctx->c_id_ind = TRUE;
  pctx->c_id_len = 0;
  pctx->h_amount_len = SIZ(ptemp->h_amount);
  pctx->h_amount_ind = TRUE;
  pctx->c_last_ind = TRUE;
  pctx->c_last_len = 0;
  pctx->w_street_1_ind = TRUE;
  pctx->w_street_1_len = 0;
  pctx->w_street_2_ind = TRUE;
  pctx->w_street_2_len = 0;
  pctx->w_city_ind = TRUE;
  pctx->w_city_len = 0;
  pctx->w_state_ind = TRUE;
  pctx->w_state_len = 0;
  pctx->w_zip_ind = TRUE;
  pctx->w_zip_len = 0;
  pctx->d_street_1_ind = TRUE;
  pctx->d_street_1_len = 0;
  pctx->d_street_2_ind = TRUE;
  pctx->d_street_2_len = 0;
  pctx->d_city_ind = TRUE;
  pctx->d_city_len = 0;
  pctx->d_state_ind = TRUE;
  pctx->d_state_len = 0;
  pctx->d_zip_ind = TRUE;
  pctx->d_zip_len = 0;
  pctx->c_first_ind = TRUE;
  pctx->c_first_len = 0;
  pctx->c_middle_ind = TRUE;
  pctx->c_middle_len = 0;
  pctx->c_street_1_ind = TRUE;
  pctx->c_street_1_len = 0;
  pctx->c_street_2_ind = TRUE;
  pctx->c_street_2_len = 0;
  pctx->c_city_ind = TRUE;
  pctx->c_city_len = 0;
  pctx->c_state_ind = TRUE;
  pctx->c_state_len = 0;
  pctx->c_zip_ind = TRUE;
  pctx->c_zip_len = 0;
  pctx->c_phone_ind = TRUE;
  pctx->c_phone_len = 0;
  pctx->c_since_ind = TRUE;
  pctx->c_since_len = 0;
  pctx->c_credit_ind = TRUE;
  pctx->c_credit_len = 0;
  pctx->c_credit_lim_ind = TRUE;
  pctx->c_credit_lim_len = 0;
  pctx->c_discount_ind = TRUE;
  pctx->c_discount_len = 0;
  pctx->c_balance_ind = TRUE;
  pctx->c_balance_len = sizeof(double);
  pctx->c_data_ind = TRUE;
  pctx->c_data_len = 0;
  pctx->h_date_ind = TRUE;
  pctx->h_date_len = 0;
  pctx->retries_ind = TRUE;
  pctx->retries_len = 0;
  pctx->cr_date_ind = TRUE;
  pctx->cr_date_len = sizeof(ptemp->cr_date);



   /* bind variables */


   OCIBNDR(pctx->curp0, pctx->w_id_bp, p,":w_id",ADR(pPay->w_id),
            SIZ(int), SQLT_INT, &pctx->w_id_ind, NULL, NULL);
   OCIBNDR(pctx->curp0, pctx->d_id_bp, p,":d_id",ADR(pPay->d_id),
            SIZ(int), SQLT_INT, &pctx->d_id_ind, NULL, NULL);
   OCIBND(pctx->curp0, pctx->c_w_id_bp, p,":c_w_id",ADR(pPay-
>c_w_id),
            SIZ(int), SQLT_INT);
   OCIBND(pctx->curp0, pctx->c_d_id_bp, p,":c_d_id",ADR(pPay-
>c_d_id),
            SIZ(int), SQLT_INT);
   OCIBND(pctx->curp0, pctx->c_id_bp, p,":c_id",ADR(pPay->c_id),
            SIZ(int),  SQLT_INT);
   OCIBNDR(pctx->curp0, pctx->h_amount_bp, p,":h_amount",
             ADR(ptemp->h_amount), SIZ(ptemp->h_amount),SQLT_INT,
             &pctx->h_amount_ind, &pctx->h_amount_len, &pctx-
>h_amount_rc);
   OCIBNDR(pctx->curp0, pctx->c_last_bp, p,":c_last",pPay->c_last,
            SIZ(pPay->c_last), SQLT_STR, &pctx->c_last_ind, &pctx-
>c_last_len,
            &pctx->c_last_rc);
   OCIBNDR(pctx->curp0, pctx->w_street_1_bp, p,":w_street_1",
             pPay->w_street_1, SIZ(pPay->w_street_1),SQLT_STR,
             &pctx->w_street_1_ind, &pctx->w_street_1_len, &pctx-
>w_street_1_rc);
   OCIBNDR(pctx->curp0, pctx->w_street_2_bp, p,":w_street_2",
             pPay->w_street_2, SIZ(pPay->w_street_2),SQLT_STR,
             &pctx->w_street_2_ind, &pctx->w_street_2_len, &pctx-
>w_street_2_rc);
   OCIBNDR(pctx->curp0, pctx->w_city_bp, p,":w_city",pPay->w_city,
             SIZ(pPay->w_city),
           SQLT_STR, &pctx->w_city_ind, &pctx->w_city_len, &pctx-
>w_city_rc);
   OCIBNDR(pctx->curp0, pctx->w_state_bp, p,":w_state",pPay-
>w_state,
             SIZ(pPay->w_state), SQLT_STR, &pctx->w_state_ind,
             &pctx->w_state_len, &pctx->w_state_rc);
   OCIBNDR(pctx->curp0, pctx->w_zip_bp, p,":w_zip",pPay->w_zip,
             SIZ(pPay->w_zip),
           SQLT_STR, &pctx->w_zip_ind, &pctx->w_zip_len, &pctx-
>w_zip_rc);
   OCIBNDR(pctx->curp0, pctx->d_street_1_bp, p,":d_street_1",
             pPay->d_street_1,
           SIZ(pPay->d_street_1),SQLT_STR, &pctx->d_street_1_ind,
           &pctx->d_street_1_len, &pctx->d_street_1_rc);
   OCIBNDR(pctx->curp0, pctx->d_street_2_bp, p,":d_street_2",
             pPay->d_street_2,
           SIZ(pPay->d_street_2),SQLT_STR, &pctx->d_street_2_ind,
           &pctx->d_street_2_len, &pctx->d_street_2_rc);
   OCIBNDR(pctx->curp0, pctx->d_city_bp, p,":d_city",pPay->d_city,
             SIZ(pPay->d_city),
           SQLT_STR, &pctx->d_city_ind, &pctx->d_city_len, &pctx-
>d_city_rc);
   OCIBNDR(pctx->curp0, pctx->d_state_bp, p,":d_state",pPay-
>d_state,
             SIZ(pPay->d_state), SQLT_STR,
           &pctx->d_state_ind, &pctx->d_state_len, &pctx-
>d_state_rc);
   OCIBNDR(pctx->curp0, pctx->d_zip_bp, p,":d_zip",pPay->d_zip,
             SIZ(pPay->d_zip),
           SQLT_STR, &pctx->d_zip_ind, &pctx->d_zip_len, &pctx-
>d_zip_rc);
   OCIBNDR(pctx->curp0, pctx->c_first_bp, p,":c_first",pPay-
>c_first,
             SIZ(pPay->c_first), SQLT_STR,
             &pctx->c_first_ind, &pctx->c_first_len, &pctx-
>c_first_rc);
   OCIBNDR(pctx->curp0, pctx->c_middle_bp, p,":c_middle",
             pPay->c_middle,2,
           SQLT_AFC, &pctx->c_middle_ind, &pctx->c_middle_len,
           &pctx->c_middle_rc);
   OCIBNDR(pctx->curp0, pctx->c_street_1_bp, p,":c_street_1",
             pPay->c_street_1,
           SIZ(pPay->c_street_1),SQLT_STR, &pctx->c_street_1_ind,
           &pctx->c_street_1_len, &pctx->c_street_1_rc);
   OCIBNDR(pctx->curp0, pctx->c_street_2_bp, p,":c_street_2",
             pPay->c_street_2,
           SIZ(pPay->c_street_2),SQLT_STR, &pctx->c_street_2_ind,
           &pctx->c_street_2_len, &pctx->c_street_2_rc);
   OCIBNDR(pctx->curp0, pctx->c_city_bp, p,":c_city",pPay->c_city,
             SIZ(pPay->c_city),
           SQLT_STR, &pctx->c_city_ind, &pctx->c_city_len, &pctx-
>c_city_rc);
   OCIBNDR(pctx->curp0, pctx->c_state_bp, p,":c_state",pPay-
>c_state,
             SIZ(pPay->c_state),
           SQLT_STR,&pctx->c_state_ind, &pctx->c_state_len, &pctx-
>c_state_rc);
   OCIBNDR(pctx->curp0, pctx->c_zip_bp, p,":c_zip",pPay->c_zip,
             SIZ(pPay->c_zip),
           SQLT_STR, &pctx->c_zip_ind, &pctx->c_zip_len, &pctx-
>c_zip_rc);
   OCIBNDR(pctx->curp0, pctx->c_phone_bp, p,":c_phone",pPay-
>c_phone,
             SIZ(pPay->c_phone), SQLT_STR,
             &pctx->c_phone_ind, &pctx->c_phone_len, &pctx-
>c_phone_rc);
   OCIBNDR(pctx->curp0, pctx->c_since_bp, p,":c_since",
             ADR(ptemp->customer_sdate),SIZ(ptemp->customer_sdate),
SQLT_ODT,
             &pctx->c_since_ind, &pctx->c_since_len, &pctx-
>c_since_rc);
   OCIBNDR(pctx->curp0, pctx->c_credit_bp, p,":c_credit",pPay-
>c_credit,
           SIZ(pPay->c_credit),SQLT_CHR,
             &pctx->c_credit_ind, &pctx->c_credit_len, &pctx-
>c_credit_rc);
   OCIBNDR(pctx->curp0, pctx->c_credit_lim_bp, p,":c_credit_lim",
           ADR(ptemp->c_credit_lim),SIZ(ptemp->c_credit_lim),
SQLT_INT,
             &pctx->c_credit_lim_ind,
             &pctx->c_credit_lim_len, &pctx->c_credit_lim_rc);
   OCIBNDR(pctx->curp0, pctx->c_discount_bp, p,":c_discount",
           ADR(ptemp->c_discount),SIZ(ptemp->c_discount), SQLT_FLT,
             &pctx->c_discount_ind,
           &pctx->c_discount_len, &pctx->c_discount_rc);
   OCIBNDR(pctx->curp0, pctx->c_balance_bp, p,":c_balance",
             ADR(pPay->c_balance), SIZ(pPay->c_balance),SQLT_FLT,
             &pctx->c_balance_ind, &pctx->c_balance_len,
           &pctx->c_balance_rc);
   OCIBNDR(pctx->curp0, pctx->c_data_bp, p,":c_data",pPay->c_data,
             SIZ(pPay->c_data),
           SQLT_STR, &pctx->c_data_ind, &pctx->c_data_len, &pctx-
>c_data_rc);
   OCIBNDR(pctx->curp0, pctx->retries_bp, p,":retry",
             ADR(ptemp->p_retry),SIZ(ptemp->p_retry), SQLT_INT,
           &pctx->retries_ind, &pctx->retries_len, &pctx-
>retries_rc);
   OCIBNDR(pctx->curp0, pctx->cr_date_bp, p,":cr_date",
             ADR(ptemp->cr_date), SIZ(ptemp->cr_date),SQLT_ODT,
```

```
            &pctx->cr_date_ind, &pctx->cr_date_len,
          &pctx->cr_date_rc);


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


   OCIBNDR(pctx->curp1, pctx->w_id_bp1, p,":w_id",
            ADR(pPay->w_id),SIZ(pPay->w_id),
          SQLT_INT, &pctx->w_id_ind, &pctx->w_id_len, &pctx-
>w_id_rc);
   OCIBNDR(pctx->curp1, pctx->d_id_bp1, p,":d_id",ADR(pPay->d_id),
            SIZ(pPay->d_id),
          SQLT_INT, &pctx->d_id_ind, &pctx->d_id_len, &pctx-
>d_id_rc);
   OCIBND(pctx->curp1, pctx->c_w_id_bp1, p,":c_w_id",ADR(pPay-
>c_w_id),
           SIZ(pPay->c_w_id),
           SQLT_INT);
   OCIBND(pctx->curp1, pctx->c_d_id_bp1, p,":c_d_id",ADR(pPay-
>c_d_id),
           SIZ(pPay->c_d_id),
           SQLT_INT);
   OCIBNDR(pctx->curp1, pctx->c_id_bp1, p,":c_id",ADR(pPay->c_id),
            SIZ(int),
          SQLT_INT, &pctx->c_id_ind, &pctx->c_id_len, &pctx-
>c_id_rc);
   OCIBNDR(pctx->curp1, pctx->h_amount_bp1, p,":h_amount",
            ADR(ptemp->h_amount),
          SIZ(int),SQLT_INT, &pctx->h_amount_ind, &pctx-
>h_amount_len,
          &pctx->h_amount_rc);
   OCIBND(pctx->curp1, pctx->c_last_bp1, p,":c_last",pPay->c_last,
           SIZ(pPay->c_last),
           SQLT_STR);
   OCIBNDR(pctx->curp1, pctx->w_street_1_bp1, p,":w_street_1",
            pPay->w_street_1,
          SIZ(pPay->w_street_1),SQLT_STR, &pctx->w_street_1_ind,
          &pctx->w_street_1_len, &pctx->w_street_1_rc);
   OCIBNDR(pctx->curp1, pctx->w_street_2_bp1, p,":w_street_2",
            pPay->w_street_2,
          SIZ(pPay->w_street_2),SQLT_STR, &pctx->w_street_2_ind,
          &pctx->w_street_2_len, &pctx->w_street_2_rc);
   OCIBNDR(pctx->curp1, pctx->w_city_bp1, p,":w_city",pPay->w_city,
            SIZ(pPay->w_city),
          SQLT_STR, &pctx->w_city_ind, &pctx->w_city_len, &pctx-
>w_city_rc);
   OCIBNDR(pctx->curp1, pctx->w_state_bp1, p,":w_state",pPay-
>w_state,
            SIZ(pPay->w_state), SQLT_STR,
            &pctx->w_state_ind, &pctx->w_state_len, &pctx-
>w_state_rc);
   OCIBNDR(pctx->curp1, pctx->w_zip_bp1, p,":w_zip",pPay->w_zip,
            SIZ(pPay->w_zip),
          SQLT_STR, &pctx->w_zip_ind, &pctx->w_zip_len, &pctx-
>w_zip_rc);
   OCIBNDR(pctx->curp1, pctx->d_street_1_bp1, p,":d_street_1",
            pPay->d_street_1,
          SIZ(pPay->d_street_1),SQLT_STR, &pctx->d_street_1_ind,
          &pctx->d_street_1_len, &pctx->d_street_1_rc);
   OCIBNDR(pctx->curp1, pctx->d_street_2_bp1, p,":d_street_2",
            pPay->d_street_2,
          SIZ(pPay->d_street_2),SQLT_STR, &pctx->d_street_2_ind,
          &pctx->d_street_2_len, &pctx->d_street_2_rc);
   OCIBNDR(pctx->curp1, pctx->d_city_bp1, p,":d_city",
            pPay->d_city,SIZ(pPay->d_city),
          SQLT_STR, &pctx->d_city_ind, &pctx->d_city_len, &pctx-
>d_city_rc);
   OCIBNDR(pctx->curp1, pctx->d_state_bp1, p,":d_state",
            pPay->d_state, SIZ(pPay->d_state), SQLT_STR,
            &pctx->d_state_ind, &pctx->d_state_len,
          &pctx->d_state_rc);
   OCIBNDR(pctx->curp1, pctx->d_zip_bp1, p,":d_zip",pPay->d_zip,
            SIZ(pPay->d_zip),
          SQLT_STR, &pctx->d_zip_ind, &pctx->d_zip_len, &pctx-
>d_zip_rc);
   OCIBNDR(pctx->curp1, pctx->c_first_bp1, p,":c_first",pPay-
>c_first,
          SIZ(pPay->c_first), SQLT_STR,
            &pctx->c_first_ind, &pctx->c_first_len,
          &pctx->c_first_rc);
   OCIBNDR(pctx->curp1, pctx->c_middle_bp1, p,":c_middle",
            pPay->c_middle,2,
          SQLT_AFC, &pctx->c_middle_ind, &pctx->c_middle_len,
          &pctx->c_middle_rc);
   OCIBNDR(pctx->curp1, pctx->c_street_1_bp1, p,":c_street_1",
            pPay->c_street_1,
          SIZ(pPay->c_street_1),SQLT_STR, &pctx->c_street_1_ind,
          &pctx->c_street_1_len, &pctx->c_street_1_rc);
   OCIBNDR(pctx->curp1, pctx->c_street_2_bp1, p,":c_street_2",
            pPay->c_street_2,
          SIZ(pPay->c_street_2),SQLT_STR, &pctx->c_street_2_ind,
          &pctx->c_street_2_len, &pctx->c_street_2_rc);
   OCIBNDR(pctx->curp1, pctx->c_city_bp1, p,":c_city",pPay->c_city,
            SIZ(pPay->c_city),SQLT_STR,
            &pctx->c_city_ind, &pctx->c_city_len, &pctx-
>c_city_rc);
   OCIBNDR(pctx->curp1, pctx->c_state_bp1, p,":c_state",pPay-
>c_state,
            SIZ(pPay->c_state),SQLT_STR, &pctx->c_state_ind,
&pctx->c_state_len,
            &pctx->c_state_rc);
   OCIBNDR(pctx->curp1, pctx->c_zip_bp1, p,":c_zip",pPay->c_zip,
            SIZ(pPay->c_zip),
          SQLT_STR, &pctx->c_zip_ind, &pctx->c_zip_len, &pctx-
>c_zip_rc);
   OCIBNDR(pctx->curp1, pctx->c_phone_bp1, p,":c_phone",pPay-
>c_phone,
            SIZ(pPay->c_phone), SQLT_STR,
            &pctx->c_phone_ind, &pctx->c_phone_len,
            &pctx->c_phone_rc);
   OCIBNDR(pctx->curp1, pctx->c_since_bp1, p,":c_since",
            ADR(ptemp->customer_sdate), SIZ(ptemp-
>customer_sdate), SQLT_ODT,
            &pctx->c_since_ind, &pctx->c_since_len, &pctx-
>c_since_rc);
   OCIBNDR(pctx->curp1, pctx->c_credit_bp1, p,":c_credit",
            pPay->c_credit,
          SIZ(pPay->c_credit),SQLT_CHR,
            &pctx->c_credit_ind, &pctx->c_credit_len,
          &pctx->c_credit_rc);
   OCIBNDR(pctx->curp1, pctx->c_credit_lim_bp1, p,":c_credit_lim",
          ADR(ptemp->c_credit_lim),SIZ(int), SQLT_INT,
            &pctx->c_credit_lim_ind,
          &pctx->c_credit_lim_len, &pctx->c_credit_lim_rc);
   OCIBNDR(pctx->curp1, pctx->c_discount_bp1, p,":c_discount",
            ADR(ptemp->c_discount),SIZ(int), SQLT_FLT, &pctx-
>c_discount_ind,
          &pctx->c_discount_len, &pctx->c_discount_rc);
   OCIBNDR(pctx->curp1, pctx->c_balance_bp1, p,":c_balance",
            ADR(pPay->c_balance),
          SIZ(double),SQLT_FLT, &pctx->c_balance_ind, &pctx-
>c_balance_len,
          &pctx->c_balance_rc);
   OCIBNDR(pctx->curp1, pctx->c_data_bp1, p,":c_data",pPay->c_data,
            SIZ(pPay->c_data),
          SQLT_STR, &pctx->c_data_ind, &pctx->c_data_len, &pctx-
>c_data_rc);
   OCIBNDR(pctx->curp1, pctx->retries_bp1, p,":retry",
            ADR(ptemp->p_retry),SIZ(int), SQLT_INT,
            &pctx->retries_ind, &pctx->retries_len, &pctx-
>retries_rc);
   OCIBNDR(pctx->curp1, pctx->cr_date_bp1, p,":cr_date",
            ADR(ptemp->cr_date), SIZ(ptemp->cr_date), SQLT_ODT,
            &pctx->cr_date_ind, &pctx->cr_date_len,
          &pctx->cr_date_rc);

 return (ERR_DB_SUCCESS);

}


tkvcp (PaymentData *pPay, OraContext *p)

{
  int execstatus;
  int errcode;
  payctx *pctx = &(p->pctx);
  paytemp *ptemp = &(p->tempvars.pay);
  unsigned char localcr_date[7];
  OCIError *datecvterrhp = p->datecvterrhp;

  vgetdate(localcr_date);
  cvtdmyhms(localcr_date,ptemp->h_date);
  OCIDateFromText(datecvterrhp,ptemp->h_date,strlen(ptemp-
>h_date),"DD-MM-YYYY HH24:MI:SS",21,(text *) 0, 0,&ptemp->cr_date);


  pctx->w_id_ind = TRUE;
  pctx->w_id_len = SIZ(pPay->w_id);
  pctx->d_id_ind = TRUE;
  pctx->d_id_len = SIZ(pPay->d_id);
  pctx->c_w_id_ind = TRUE;
  pctx->c_w_id_len = 0;
  pctx->c_d_id_ind = TRUE;
  pctx->c_d_id_len = 0;
  pctx->c_id_ind = TRUE;
  pctx->c_id_len = 0;
  pctx->h_amount_len = SIZ(ptemp->h_amount);
  pctx->h_amount_ind = TRUE;
  pctx->c_last_ind = TRUE;
  pctx->c_last_len = SIZ(pPay->c_last);
  pctx->w_street_1_ind = NA;
  pctx->w_street_1_len = 0;
  pctx->w_street_2_ind = NA;
  pctx->w_street_2_len = 0;
  pctx->w_city_ind = NA;
  pctx->w_city_len = 0;
  pctx->w_state_ind = NA;
  pctx->w_state_len = 0;
  pctx->w_zip_ind = NA;
  pctx->w_zip_len = 0;
  pctx->d_street_1_ind = NA;
  pctx->d_street_1_len = 0;
  pctx->d_street_2_ind = NA;
  pctx->d_street_2_len = 0;
  pctx->d_city_ind = NA;
  pctx->d_city_len = 0;
  pctx->d_state_ind = NA;
  pctx->d_state_len = 0;
  pctx->d_zip_ind = NA;
  pctx->d_zip_len = 0;
  pctx->c_first_ind = NA;
  pctx->c_first_len = 0;
  pctx->c_middle_ind = NA;
  pctx->c_middle_len = 0;
  pctx->c_street_1_ind = NA;
  pctx->c_street_1_len = 0;
  pctx->c_street_2_ind = NA;
  pctx->c_street_2_len = 0;
  pctx->c_city_ind = NA;
  pctx->c_city_len = 0;
  pctx->c_state_ind = NA;
  pctx->c_state_len = 0;
  pctx->c_zip_ind = NA;
  pctx->c_zip_len = 0;
```

```
  pctx->c_phone_ind = NA;
  pctx->c_phone_len = 0;
  pctx->c_since_ind = NA;
  pctx->c_since_len = 0;
  pctx->c_credit_ind = NA;
  pctx->c_credit_len = 0;
  pctx->c_credit_lim_ind = NA;
  pctx->c_credit_lim_len = 0;
  pctx->c_discount_ind = NA;
  pctx->c_discount_len = 0;
  pctx->c_balance_ind = NA;
  pctx->c_balance_len = sizeof(double);
  pctx->c_data_ind = NA;
  pctx->c_data_len = 0;
  pctx->h_date_ind = TRUE;
  pctx->h_date_len = 0;
  pctx->retries_ind = TRUE;
  pctx->retries_len = 0;
  pctx->cr_date_ind = TRUE;
  pctx->cr_date_len = sizeof(ptemp->cr_date);



  if(pPay->byname)
  {

    pctx->c_id_ind = NA;
    execstatus=OCIStmtExecute(p->tpcsvc,pctx->curp1,p->errhp,

1,0,0,0,OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
  }
  else
  {
    pctx->c_last_ind = NA;
    execstatus=OCIStmtExecute(p->tpcsvc,pctx->curp0,p->errhp,

1,0,0,0,OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
  }


  if(execstatus != OCI_SUCCESS) {
     OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
     errcode = OCIERROR(p,execstatus);
     if(errcode == NOT_SERIALIZABLE) {
        return(RECOVERR);
     } else if (errcode == RECOVERR) {
        return(RECOVERR);
     } else if (errcode == SNAPSHOT_TOO_OLD) {
        return(RECOVERR);
     } else {
       TPCCErr("DB error in payment transaction cursor p0
%d\n",errcode);
        return ERR_DB_ERROR;
     }
  }

  return (ERR_DB_SUCCESS);

}



void tkvcpdone (payctx *ppctx)
{
   payctx pctx = *ppctx;

   if(NULL != pctx.curpi)
     OCIHandleFree((dvoid *)pctx.curpi,OCI_HTYPE_STMT);
   if(NULL != pctx.curp0)
     OCIHandleFree((dvoid *)pctx.curp0,OCI_HTYPE_STMT);
   if(NULL != pctx.curp1)
     OCIHandleFree((dvoid *)pctx.curp1,OCI_HTYPE_STMT);

}


/*
---------------------------------------------------------------------
-----------
Orderstatus transaction
*/

#define SQL_ORD_CUR0 "SELECT /*+ no_index(ICUST1) index (cust
icust2) */ rowid FROM cust \
                WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last
= :c_last \
                ORDER BY c_w_id, c_d_id, c_last, c_first"

#define SQL_ORD_CUR1 "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_w_id, o_d_id, o_c_id, o_id DESC"


#define SQL_ORD_CUR2 "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, c_id \
                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, o_d_id, o_w_id, o_id DESC"


#define SQL_ORD_CUR3 "SELECT
ol_i_id,ol_supply_w_id,ol_quantity,ol_amount, \
                ol_delivery_d\
                FROM ordl \
                WHERE ol_d_id = :d_id AND ol_w_id = :w_id AND
ol_o_id = :o_id"

#define SQL_ORD_CUR4 "SELECT count (c_last) FROM cust \
                WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last
= :c_last "



sb4 rid_data(dvoid *ctxp, OCIDefine *dp, ub4 iter, dvoid **bufpp,
ub4 **alenp,
         ub1 *piecep, dvoid **indpp, ub2 **rcodepp)
{
ub4 i;

ordctx *octx = &(((OraContext *)ctxp)->octx);
defctx *cbctx = &(((OraContext *)ctxp)->cbctx);

if (cbctx->reexec)  /* if this is the second execute - use entry 0
*/
{
  i=0;
  cbctx->count--;   /* count down */
}
else
  i=iter;

*bufpp = octx->c_rowid_ptr[i];
*indpp = &octx->c_rowid_ind[i] ;
*alenp = &octx->c_rowid_len[i] ;
*rcodepp = &octx->c_rowid_rcode[i] ;
*piecep = OCI_ONE_PIECE;

return (OCI_CONTINUE);
}

tkvcoinit (OrderStatusData *pOrd,
            OraContext *p)
{

  int i;
  text stmbuf[8192];
  ordtemp *otemp = &(p->tempvars.ord);
  ordctx *octx = &(p->octx);

  memset(octx,(char)0,sizeof(ordctx));
  octx->cs = 1;
  octx->norow = 0;
  octx->somerows = 10;
  octx->curo0 = NULL;
  octx->curo1 = NULL;
  octx->curo2 = NULL;
  octx->curo3 = NULL;
  octx->curo4 = NULL;

/* get the rowid handles */
  for(i=0;i<100;i++) {
    OCIERROR(p, OCIDescriptorAlloc(p->tpcenv,(dvoid**)&octx-
>c_rowid_ptr[i],

OCI_DTYPE_ROWID,0,(dvoid**)0));
  }


   OCIERROR(p,OCIHandleAlloc(p->tpcenv,(dvoid**)&octx-
>curo0,OCI_HTYPE_STMT,
                             0,(dvoid**)0));
   OCIERROR(p,OCIHandleAlloc(p->tpcenv,(dvoid**)&octx-
>curo1,OCI_HTYPE_STMT,
                                       0,(dvoid**)0));
   OCIERROR(p,OCIHandleAlloc(p->tpcenv,(dvoid**)&octx-
>curo2,OCI_HTYPE_STMT,
                                       0,(dvoid**)0));
   OCIERROR(p,OCIHandleAlloc(p->tpcenv,(dvoid**)&octx-
>curo3,OCI_HTYPE_STMT,
                                       0,(dvoid**)0));
   OCIERROR(p,OCIHandleAlloc(p->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*/
   sprintf((char *) stmbuf, SQL_ORD_CUR0);
   OCIERROR(p,OCIStmtPrepare(octx->curo0,p->errhp,stmbuf,
                                       strlen((char *)stmbuf),
                                       OCI_NTV_SYNTAX,OCI_DEFAULT));
   OCIERROR(p,OCIAttrSet(octx->curo0,OCI_HTYPE_STMT,(dvoid*)&octx-
>norow,0,
                                 OCI_ATTR_PREFETCH_ROWS,p->errhp));

/* get order/customer info back based on rowid */
   sprintf((char *) stmbuf, SQL_ORD_CUR1);
   OCIERROR(p,OCIStmtPrepare(octx->curo1,p->errhp,stmbuf,
                                       strlen((char *)stmbuf),
                                       OCI_NTV_SYNTAX,OCI_DEFAULT));
   OCIERROR(p,OCIAttrSet(octx->curo1,OCI_HTYPE_STMT,(dvoid*)&octx-
>norow,0,
                                 OCI_ATTR_PREFETCH_ROWS,p->errhp));

/*  c_id != 0, use id to find customer  */
   sprintf((char *) stmbuf, SQL_ORD_CUR2);
```

```
   OCIERROR(p,OCIStmtPrepare(octx->curo2,p->errhp,stmbuf,
                                 strlen((char *)stmbuf),
                                 OCI_NTV_SYNTAX,OCI_DEFAULT));
   OCIERROR(p,OCIAttrSet(octx->curo2,OCI_HTYPE_STMT,(dvoid*)&octx-
>norow,0,
                              OCI_ATTR_PREFETCH_ROWS,p->errhp));

   sprintf((char *) stmbuf, SQL_ORD_CUR3);
   OCIERROR(p,OCIStmtPrepare(octx->curo3,p->errhp,stmbuf,
                                 strlen((char *)stmbuf),
                                 OCI_NTV_SYNTAX,OCI_DEFAULT));
   OCIERROR(p,OCIAttrSet(octx->curo3,OCI_HTYPE_STMT,(dvoid*)&octx-
>norow,0,
                              OCI_ATTR_PREFETCH_ROWS,p->errhp));

   sprintf((char *) stmbuf, SQL_ORD_CUR4);
   OCIERROR(p,OCIStmtPrepare(octx->curo4,p->errhp,stmbuf,
                                 strlen((char *)stmbuf),
                                 OCI_NTV_SYNTAX,OCI_DEFAULT));
   OCIERROR(p,OCIAttrSet(octx->curo4,OCI_HTYPE_STMT,(dvoid*)&octx-
>norow,0,
                              OCI_ATTR_PREFETCH_ROWS,p->errhp));

   for (i = 0; i < NITEMS; i++) {
      octx->ol_supply_w_id_ind[i] = TRUE;
      octx->ol_i_id_ind[i] = TRUE;
      octx->ol_quantity_ind[i] = TRUE;
      octx->ol_amount_ind[i] = TRUE;
      octx->ol_delivery_d_ind[i] = TRUE;

      octx->ol_supply_w_id_len[i] = sizeof(int);
      octx->ol_i_id_len[i] = sizeof(int);
      octx->ol_quantity_len[i] = sizeof(int);
      octx->ol_amount_len[i] = sizeof(int);
      octx->ol_delivery_d_len[i] = sizeof(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;
   octx->ol_w_id_csize = NITEMS;
   octx->ol_o_id_csize = NITEMS;
   octx->ol_d_id_csize = NITEMS;
   octx->ol_w_id_ind = TRUE;
   octx->ol_d_id_ind = TRUE;
   octx->ol_o_id_ind = TRUE;
   octx->ol_w_id_len = sizeof(int);
   octx->ol_d_id_len = sizeof(int);
   octx->ol_o_id_len = sizeof(int);

   /* bind variables */

   /* c_id (customer id) is not known */
   /* cursor 0 */
   OCIBND(octx->curo0,octx->w_id_bp0,p,":w_id",ADR(pOrd-
>w_id),SIZ(int),
            SQLT_INT);
   OCIBND(octx->curo0,octx->d_id_bp0,p,":d_id",ADR(pOrd-
>d_id),SIZ(int),
            SQLT_INT);
   OCIBND(octx->curo0,octx->c_last_bp,p,":c_last",pOrd->c_last,
            SIZ(pOrd->c_last),SQLT_STR);

       ocierror(__FILE__,__LINE__,p, \
       OCIDefineByPos(octx->curo0,&octx->c_rowid_dp,p->errhp,1, \
          octx->c_rowid_ptr, sizeof(OCIRowid*),SQLT_RDD,octx-
>c_rowid_ind,\
          NULL,NULL, OCI_DYNAMIC_FETCH));
       ocierror(__FILE__,__LINE__,p, OCIDefineDynamic(octx-
>c_rowid_dp,\
          p->errhp,\
          p,rid_data));
   OCIBND(octx->curo1,octx->c_rowid_bp,p,":cust_rowid",
          &octx->middle_cust,sizeof(octx->middle_cust),SQLT_RDD);

   OCIDEF(octx->curo1,octx->c_id_dp,p->errhp,1,ADR(pOrd-
>c_id),SIZ(int),
            SQLT_INT);
   OCIDEF(octx->curo1,octx->c_balance_dp1,p->errhp,2,ADR(pOrd-
>c_balance),
            SIZ(double),SQLT_FLT);
   OCIDEF(octx->curo1,octx->c_first_dp1,p->errhp,3,pOrd->c_first,
            SIZ(pOrd->c_first)-1,
            SQLT_CHR);
   OCIDEF(octx->curo1,octx->c_middle_dp1,p->errhp,4,pOrd->c_middle,
            SIZ(pOrd->c_middle)-1,SQLT_AFC);

   OCIDEF(octx->curo1,octx->c_last_dp1,p->errhp,5,pOrd->c_last,
            SIZ(pOrd->c_last)-1, SQLT_CHR);
   OCIDEF(octx->curo1,octx->o_id_dp1,p->errhp,6,ADR(pOrd-
>o_id),SIZ(int),
            SQLT_INT);
   OCIDEF(octx->curo1,octx->o_entry_d_dp1,p->errhp,7,
            &otemp->entry_date,SIZ(otemp->entry_date),SQLT_ODT);
   OCIDEF(octx->curo1,octx->o_cr_id_dp1,p->errhp,8,ADR(pOrd-
>o_carrier_id),
            SIZ(int),SQLT_INT);
   OCIDEF(octx->curo1,octx->o_ol_cnt_dp1,p->errhp,9,ADR(pOrd-
>o_ol_cnt),
            SIZ(int),SQLT_INT);



/* Bind for cursor 2 , no-zero customer id */
   OCIBND(octx->curo2,octx->w_id_bp2,p,":w_id",ADR(pOrd-
>w_id),SIZ(int),
            SQLT_INT);
   OCIBND(octx->curo2,octx->d_id_bp2,p,":d_id",ADR(pOrd-
>d_id),SIZ(int),
            SQLT_INT);
   OCIBND(octx->curo2,octx->c_id_bp,p,":c_id",ADR(pOrd-
>c_id),SIZ(int),
            SQLT_INT);
   OCIDEF(octx->curo2,octx->c_balance_dp2,p->errhp,1,ADR(pOrd-
>c_balance),
            SIZ(double),SQLT_FLT);
   OCIDEF(octx->curo2,octx->c_first_dp2,p->errhp,2,pOrd->c_first,
            SIZ(pOrd->c_first)-1,
            SQLT_CHR);
   OCIDEF(octx->curo2,octx->c_middle_dp2,p->errhp,3,pOrd->c_middle,
            SIZ(pOrd->c_middle)-1,SQLT_AFC);
   OCIDEF(octx->curo2,octx->c_last_dp,p->errhp,4,pOrd->c_last,
            SIZ(pOrd->c_last)-1,
            SQLT_CHR);
   OCIDEF(octx->curo2,octx->o_id_dp2,p->errhp,5,ADR(pOrd-
>o_id),SIZ(int),
            SQLT_INT);
   OCIDEF(octx->curo2,octx->o_entry_d_dp2,p->errhp,6,
           &otemp->entry_date,SIZ(otemp->entry_date),SQLT_ODT);
   OCIDEF(octx->curo2, octx->o_cr_id_dp2,p->errhp,7,ADR(pOrd-
>o_carrier_id),
            SIZ(int), SQLT_INT);
   OCIDEF(octx->curo2,octx->o_ol_cnt_dp2,p->errhp,8,ADR(pOrd-
>o_ol_cnt),
            SIZ(int),SQLT_INT);
   OCIDEF(octx->curo2,octx->c_id_dp1,p->errhp,9,ADR(pOrd-
>c_id),SIZ(int),
            SQLT_INT);

/* Bind for last cursor - 3 */

   OCIBND(octx->curo3,octx->w_id_bp3,p,":w_id",ADR(pOrd-
>w_id),SIZ(int),
            SQLT_INT);
   OCIBND(octx->curo3,octx->d_id_bp3,p,":d_id",ADR(pOrd-
>d_id),SIZ(int),
            SQLT_INT);
   OCIBND(octx->curo3,octx->o_id_bp,p,":o_id",ADR(pOrd-
>o_id),SIZ(int),
            SQLT_INT);

   OCIDFNRA(octx->curo3, octx->ol_i_id_dp, p->errhp, 1, otemp-
>loc_ol_i_id,
              SIZ(int), SQLT_INT,
              octx->ol_i_id_ind,octx->ol_i_id_len, octx-
>ol_i_id_rcode);
   OCIDFNRA(octx->curo3,octx->ol_supply_w_id_dp,p->errhp,2,
              otemp->loc_ol_supply_w_id,
             SIZ(int),SQLT_INT, octx->ol_supply_w_id_ind,
             octx->ol_supply_w_id_len, octx->ol_supply_w_id_rcode);
   OCIDFNRA(octx->curo3, octx->ol_quantity_dp,p->errhp,3,
              otemp->loc_ol_quantity, SIZ(int),
             SQLT_INT, octx->ol_quantity_ind,octx->ol_quantity_len,
              octx->ol_quantity_rcode);
   OCIDFNRA(octx->curo3,octx->ol_amount_dp,p->errhp,4,otemp-
>loc_ol_amount,
              SIZ(int),
             SQLT_INT,octx->ol_amount_ind, octx->ol_amount_len,
             octx->ol_amount_rcode);
   OCIDFNRA(octx->curo3,octx->ol_d_base_dp,p->errhp,5,
              otemp->loc_ol_delivery_date,SIZ(OCIDate), SQLT_ODT,
             octx->ol_delivery_d_ind,octx->ol_delivery_d_len,
              octx->ol_delivery_d_rcode);

   OCIBND(octx->curo4, octx->w_id_bp4, p, ":w_id", ADR(pOrd->w_id),
SIZ(int),
                  SQLT_INT);
   OCIBND(octx->curo4,octx->d_id_bp4,p,":d_id",ADR(pOrd-
>d_id),SIZ(int),
                  SQLT_INT);
   OCIBND(octx->curo4,octx->c_last_bp4,p,":c_last",ADR(pOrd-
>c_last),
              SIZ(pOrd->c_last), SQLT_STR);
   OCIDEF(octx->curo4,
                    octx->c_count_dp,
                    p,
                    1,
                    ADR(octx->rcount),
                    SIZ(int),
              SQLT_INT);

  return (ERR_DB_SUCCESS);

 }


tkvco (OrderStatusData *pOrd, OraContext *p)
{
  ordctx *octx = &(p->octx);
  defctx *cbctx = &(p->cbctx);
  ordtemp *otemp = &(p->tempvars.ord);
  int i;
  int execstatus;
  int errcode;
  int entry_date_str_len = sizeof (otemp->entry_date_str);


  int rcount;

  for (i = 0; i < NITEMS; i++) {
      octx->ol_supply_w_id_ind[i] = TRUE;
      octx->ol_i_id_ind[i] = TRUE;
      octx->ol_quantity_ind[i] = TRUE;
      octx->ol_amount_ind[i] = TRUE;
      octx->ol_delivery_d_ind[i] = TRUE;
```

```
      octx->ol_supply_w_id_len[i] = sizeof(int);
      octx->ol_i_id_len[i] = sizeof(int);
      octx->ol_quantity_len[i] = sizeof(int);
      octx->ol_amount_len[i] = sizeof(int);
      octx->ol_delivery_d_len[i] = sizeof(otemp-
>loc_ol_delivery_date[i]);
   }
   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;

   /* initialize bound output variables to null for oracle */
   if(pOrd->byname)
     {
     cbctx->reexec = FALSE;
     execstatus=OCIStmtExecute(p->tpcsvc,octx->curo0,p->errhp,
                                  100,0,0,0,OCI_DEFAULT);
     if ((execstatus != OCI_NO_DATA) && (execstatus !=
OCI_SUCCESS))
            /* will get OCI_NO_DATA if <100 found */
     {
         errcode = OCIERROR(p,execstatus);
         if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
||
            (errcode == SNAPSHOT_TOO_OLD))
         {
             OCITransCommit(p->tpcsvc,p->errhp,OCI_DEFAULT);
               return RECOVERR;
         } else {
       TPCCErr("DB error in order status transaction cursor o0
%d\n",errcode);
           return ERR_DB_ERROR;
         }
     }
     if (execstatus == OCI_NO_DATA)  /* there are no more rows */
     {
      /* get rowcount, find middle one */
     OCIAttrGet(octx-
>curo0,OCI_HTYPE_STMT,&rcount,NULL,OCI_ATTR_ROW_COUNT,
                  p->errhp);
     if (rcount <1)
     {
     TPCCErr("No Data Found");
     return ERR_DB_ERROR;
     }
     octx->cust_idx=(rcount-1)/2 ;
     }
     else
     {
     /* count  the number of rows */
     execstatus = OCIStmtExecute(p->tpcsvc,octx->curo4,p->errhp,
                                      1,0,0,0,OCI_DEFAULT);
     if ((execstatus != OCI_NO_DATA) && (execstatus !=
OCI_SUCCESS))
     {
      errcode = OCIERROR(p,execstatus);
      if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
         || (errcode == SNAPSHOT_TOO_OLD))
       {
         OCITransCommit(p->tpcsvc,p->errhp,OCI_DEFAULT);
           return RECOVERR;
       } else {
       TPCCErr("DB error in order status transaction cursor o4
%d\n",errcode);
           return ERR_DB_ERROR;
       }
     }
     if (octx->rcount+1 < 200)
         octx->cust_idx=(octx->rcount-1)/2;
     else
     {
     cbctx->reexec = TRUE;
     cbctx->count = (octx->rcount+1)/2;
     execstatus = OCIStmtExecute(p->tpcsvc,octx->curo0,p->errhp,
                                     cbctx-
>count,0,0,0,OCI_DEFAULT);
     /* will get OCI_NO_DATA if <100 found */
     if (cbctx->count>0)
     {
     TPCCErr("Did not get all rows.");
     return ERR_DB_ERROR;
     }
     if ((execstatus != OCI_NO_DATA) && (execstatus !=
OCI_SUCCESS))
     {
      errcode=OCIERROR(p,execstatus);
      if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
         || (errcode == SNAPSHOT_TOO_OLD))
      {
         OCITransCommit(p->tpcsvc,p->errhp,OCI_DEFAULT);
           return RECOVERR;
       } else {
       TPCCErr("DB error in order status transaction cursor o0
%d\n",errcode);
         return ERR_DB_ERROR;
       }
     }
     octx->cust_idx=0;
   }
  }

  octx->middle_cust=octx->c_rowid_ptr[octx->cust_idx];
  execstatus = OCIStmtExecute(p->tpcsvc,octx->curo1,p->errhp,
                                     1,0,0,0,OCI_DEFAULT);
  if (execstatus != OCI_SUCCESS)
   {
      errcode = OCIERROR(p,execstatus);
      OCITransCommit(p->tpcsvc,p->errhp,OCI_DEFAULT);
      if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR) ||
            (errcode == SNAPSHOT_TOO_OLD))
      {
       return RECOVERR;
      } else {
       TPCCErr("DB error in order status transaction cursor o1
%d\n",errcode);
        return ERR_DB_ERROR;
      }
   }
}
else
{
     execstatus = OCIStmtExecute(p->tpcsvc,octx->curo2,p->errhp,
                                          1,0,0,0,OCI_DEFAULT);
     if (execstatus != OCI_SUCCESS)
      {
         errcode = OCIERROR(p,execstatus);
         OCITransCommit(p->tpcsvc,p->errhp,OCI_DEFAULT);
         if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
              || (errcode == SNAPSHOT_TOO_OLD))
         {
             return RECOVERR;
         } else {
       TPCCErr("DB error in order status transaction cursor o2
%d\n",errcode);
           return ERR_DB_ERROR;
         }
      }
   }
   octx->ol_w_id_ind = TRUE;
   octx->ol_d_id_ind = TRUE;
   octx->ol_o_id_ind = TRUE;
   octx->ol_w_id_len = sizeof(int);
   octx->ol_d_id_len = sizeof(int);
   octx->ol_o_id_len = sizeof(int);

   execstatus =   OCIStmtExecute(p->tpcsvc,octx->curo3,p-
>errhp,pOrd->o_ol_cnt,
                                          0,0,0, OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
   if (execstatus != OCI_SUCCESS)
     {
         errcode = OCIERROR(p,execstatus);
         OCITransCommit(p->tpcsvc,p->errhp,OCI_DEFAULT);
         if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
              || (errcode == SNAPSHOT_TOO_OLD))
         {
             return RECOVERR;
         } else {
       TPCCErr("DB error in order status transaction cursor o3
%d\n",errcode);
           return ERR_DB_ERROR;
         }
     }


   /* clean up and convert the delivery dates */

  for (i = 0; i < pOrd->o_ol_cnt; i++) {
      if (octx->ol_delivery_d_ind[i] == -1) /* null date in field
*/
          strncpy(otemp->ol_delivery_date_str[i],"01-01-1811",10);
      else
      {
        octx->ol_delivery_d_len[i]=sizeof(otemp-
>ol_delivery_date_str[i]);
        OCIERROR(p, OCIDateToText(p->errhp,&otemp-
>loc_ol_delivery_date[i],
           (text*)"dd-mm-yyyy",strlen("dd-mm-yyyy"),(text*)0,0,
            &octx->ol_delivery_d_len[i],&otemp-
>ol_delivery_date_str[i]));
      }

   }

  /* convert the order entry date */
    OCIERROR(p, OCIDateToText(p->errhp,&otemp->entry_date,
           (text*)"dd-mm-yyyy HH24:MI:SS",strlen("dd-mm-yyyy
HH:MI:SS"),(text*)0,0,
            &entry_date_str_len,&otemp->entry_date_str));

  return (ERR_DB_SUCCESS);
  }


void tkvcodone (ordctx *poctx)
{
  ordctx octx = *poctx;

  if(NULL != octx.curo0)
    OCIHandleFree((dvoid *)octx.curo0,OCI_HTYPE_STMT);
  if(NULL != octx.curo1)
    OCIHandleFree((dvoid *)octx.curo1,OCI_HTYPE_STMT);
  if(NULL != octx.curo2)
    OCIHandleFree((dvoid *)octx.curo2,OCI_HTYPE_STMT);
  if(NULL != octx.curo3)
    OCIHandleFree((dvoid *)octx.curo3,OCI_HTYPE_STMT);
  if(NULL != octx.curo4)
    OCIHandleFree((dvoid *)octx.curo4,OCI_HTYPE_STMT);

}


/**** delivery transaction  */
```

```
#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
#define SQLTXT0 "SELECT substr(value,1,5) FROM v$parameter \
   WHERE name = 'instance_number'"
#endif



#define SQLTXT1 "DELETE FROM nord WHERE no_d_id = :d_id \
        AND no_w_id=:w_id and rownum <=1 \
          RETURNING no_o_id into :o_id "

#define SQLTXT3 "UPDATE ordr SET o_carrier_id = :carrier_id \
   WHERE o_id=:o_id and o_d_id=:d_id and o_w_id=:w_id \
   returning o_c_id into :o_c_id"

#define SQLTXT4 "UPDATE /*+ buffer */ 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 ol_amount into :ol_amount "


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



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)
{

  delctx *dctx;
  dctx=(delctx*)ctxp;

  *bufpp = &dctx->del_o_id[iter];
  *indpp= &dctx->del_o_id_ind[iter];
  dctx->del_o_id_len[iter]=sizeof(dctx->del_o_id[0]);
  *alenp= &dctx->del_o_id_len[iter];
  *rcodepp = &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)
{

  delctx *dctx;
  dctx=(delctx*)ctxp;

  *bufpp = &dctx->c_id[iter];
  *indpp= &dctx->c_id_ind[iter];
  dctx->c_id_len[iter]=sizeof(dctx->c_id[0]);
  *alenp= &dctx->c_id_len[iter];
  *rcodepp = &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;
  actx->ol_cnt[iter]=actx->ol_cnt[iter]+1;
  *bufpp = &actx->ol_amt[iter][index];
  *indpp= &actx->ol_amt_ind[iter][index];
  actx->ol_amt_len[iter][index]=sizeof(actx->ol_amt[0][0]);
  *alenp= &actx->ol_amt_len[iter][index];
  *rcodepp = &actx->ol_amt_rcode[iter][index];
  *piecep =OCI_ONE_PIECE;
  return (OCI_CONTINUE);
}



void shiftdata(from,dctx)
int from;
delctx *dctx;
{
   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];
   }
}    /* end shiftdata */


tkvcdinit (DeliveryData *pDel,
            OraContext *p)
{
   delctx *dctx = &(p->dctx);
   amtctx *actx = &(p->actx);
   text stmbuf[8192];
   memset(dctx,(char)0,sizeof(delctx));
   memset(actx,(char)0,sizeof(amtctx));

   dctx->norow = 0;
   dctx->curd1 = NULL;
   dctx->curd2 = NULL;
   dctx->curd3 = NULL;
   dctx->curd4 = NULL;
   dctx->curd5 = NULL;
   dctx->curd6 = NULL;


#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
   OCIHandleAlloc(tpcenv, (dvoid **)(&dctx->curd0), OCI_HTYPE_STMT,
0, (dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT0);
   OCIStmtPrepare(dctx->curd0, p->errhp, stmbuf, strlen((char
*)stmbuf),OCI_NTV_SYNTAX, OCI_DEFAULT);

   OCIDFNRA(dctx->curd0, dctx->inum_dp,p->errhp,1,dctx-
>inum,SIZ(dctx->inum),SQLT_STR,
         dctx->inum_ind,dctx->inum_len,dctx->inum_rcode);
#endif


   /* open first cursor */

   OCIHandleAlloc(p->tpcenv, (dvoid **)(&dctx->curd1),
OCI_HTYPE_STMT,
                        0, (dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT1);
   OCIStmtPrepare(dctx->curd1, p->errhp, stmbuf, strlen((char
*)stmbuf),
                        OCI_NTV_SYNTAX, OCI_DEFAULT);



   /* bind variables */

   OCIBND(dctx->curd1, dctx->w_id_bp,p,":w_id",dctx->w_id,SIZ(int),
            SQLT_INT);
   OCIBNDRA(dctx->curd1, dctx->d_id_bp,p,":d_id",dctx-
>d_id,SIZ(int),
            SQLT_INT,NULL,NULL,NULL);
   OCIBNDRAD(dctx->curd1, dctx->del_o_id_bp,p,":o_id",SIZ(int),
            SQLT_INT,NULL,dctx,no_data,TPC_oid_data);

   /* open third cursor */

   OCIHandleAlloc(p->tpcenv, (dvoid **)(&dctx->curd3),
OCI_HTYPE_STMT, 0,
                        (dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT3);
   OCIStmtPrepare(dctx->curd3, p->errhp, stmbuf, strlen((char
*)stmbuf),
                        OCI_NTV_SYNTAX, OCI_DEFAULT);


   /* bind variables */

   OCIBNDRA(dctx->curd3, dctx->carrier_id_bp,p,":carrier_id",dctx-
>carrier_id,
            SIZ(int),SQLT_INT,dctx->carrier_id_ind,
            dctx->carrier_id_len,dctx->carrier_id_rcode);
   OCIBNDRA(dctx->curd3, dctx->w_id_bp3,p,":w_id",dctx->w_id,
            SIZ(int),SQLT_INT,NULL,NULL,NULL);
   OCIBNDRA(dctx->curd3, dctx->d_id_bp3,p,":d_id",dctx->d_id,
            SIZ(int),SQLT_INT,NULL,NULL,NULL);
   OCIBNDRA(dctx->curd3, dctx->del_o_id_bp3,p,":o_id",dctx-
>del_o_id,
            SIZ(int),SQLT_INT,NULL,NULL,NULL);

   OCIBNDRAD(dctx->curd3, dctx->c_id_bp3,p,":o_c_id",SIZ(int),
            SQLT_INT,NULL,dctx,no_data,cid_data);


   /* open fourth cursor */

   OCIHandleAlloc(p->tpcenv, (dvoid **)(&dctx->curd4),
OCI_HTYPE_STMT, 0,
                        (dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT4);
   OCIStmtPrepare(dctx->curd4, p->errhp, stmbuf, strlen((char
*)stmbuf),
                        OCI_NTV_SYNTAX, OCI_DEFAULT);

   /* bind variables */

   OCIBND(dctx->curd4, dctx->w_id_bp4,p,":w_id",dctx-
>w_id,SIZ(dctx->w_id[0]),
            SQLT_INT);
   OCIBND(dctx->curd4, dctx->d_id_bp4,p,":d_id",dctx-
>d_id,SIZ(dctx->d_id[0]),
            SQLT_INT);
   OCIBND(dctx->curd4, dctx->o_id_bp,p,":o_id",dctx-
>del_o_id,SIZ(int),
            SQLT_INT);
   OCIBND(dctx->curd4, dctx->cr_date_bp,p,":cr_date",dctx-
>del_date,
```

```
            SIZ(OCIDate), SQLT_ODT);

   OCIBNDRAD(dctx->curd4, dctx->olamt_bp,p,":ol_amount",SIZ(int),
            SQLT_INT,NULL,actx,no_data,amt_data);


   /* open sixth cursor */

   OCIHandleAlloc(p->tpcenv, (dvoid **)(&dctx->curd6),
OCI_HTYPE_STMT, 0,
                  (dvoid**)0);
   sprintf ((char *) stmbuf, SQLTXT6);
   OCIStmtPrepare(dctx->curd6, p->errhp, stmbuf, strlen((char
*)stmbuf),
                  OCI_NTV_SYNTAX, OCI_DEFAULT);

   /* bind variables */

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


   return (ERR_DB_SUCCESS);

}


tkvcd (DeliveryData *pDel, OraContext *p)
{
  delctx *dctx = &(p->dctx);
  amtctx *actx = &(p->actx);
  deltemp *dtemp = &(p->tempvars.del);
  int i, j;
  int rpc,rcount,count;
  int invalid;
  unsigned char localcr_date[7];
  OCIError *datecvterrhp = p->datecvterrhp;
  int execstatus;
  int errcode;
  int proc_no = 0;

  invalid = 0;

  vgetdate(localcr_date);
  cvtdmyhms(localcr_date,dtemp->cvtcr_date);
  OCIDateFromText(datecvterrhp,dtemp->cvtcr_date,strlen(dtemp-
>cvtcr_date),"DD-MM-YYYY HH24:MI:SS",21,(text *) 0, 0,&dtemp-
>cr_date);



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

   OCIStmtExecute(p->tpcsvc,dctx->curd0,p-
>errhp,1,0,0,0,OCI_DEFAULT);
   sysdate(sdate);
   printf ("Delivery started at %s on node %s\n",sdate,dctx->inum);
#endif


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

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
iso:
#endif

   /* initialization for array operations */

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

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

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

      actx->ol_cnt[i]=0;
   }


   /* array select from new_order and orders tables */

   execstatus=OCIStmtExecute(p->tpcsvc,dctx->curd1,p-
>errhp,NDISTS,0,0,0,
                                    OCI_DEFAULT);
   if((execstatus != OCI_SUCCESS) && (execstatus != OCI_NO_DATA))
   {
     OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
     errcode = OCIERROR(p,execstatus);
     if(errcode == NOT_SERIALIZABLE)
     {
       return(RECOVERR);
     }
     else if (errcode == RECOVERR)
     {
       return(RECOVERR);
     }
     else if (errcode == SNAPSHOT_TOO_OLD)
     {
       return(RECOVERR);
     }
     else
     {
       TPCCErr("DB error in delivery transaction cursor d1
%d\n",errcode);
       return (ERR_DB_ERROR);
     }
   }


   /* mark districts with no new order */
   OCIAttrGet(dctx-
>curd1,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROW_COUNT,p->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,dctx);
     }
   }

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



   execstatus=OCIStmtExecute(p->tpcsvc,dctx->curd3,p-
>errhp,rpc,0,0,0,
                                    OCI_DEFAULT);
   if(execstatus != OCI_SUCCESS)
   {
     OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
     errcode = OCIERROR(p,execstatus);
     if(errcode == NOT_SERIALIZABLE)
     {
       return(RECOVERR);
     }
     else if (errcode == RECOVERR)
     {
       return (RECOVERR);
     }
     else if (errcode == SNAPSHOT_TOO_OLD)
     {
       return (RECOVERR);
     }
     else
     {
```

```
      TPCCErr("DB error in delivery transaction cursor d3
%d\n",errcode);
       return (ERR_DB_ERROR);
     }
   }


   OCIAttrGet(dctx-
>curd3,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROW_COUNT,p->errhp);

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

   /* array update of order_line table */
   execstatus=OCIStmtExecute(p->tpcsvc,dctx->curd4,p-
>errhp,rpc,0,0,0,
                                  OCI_DEFAULT);
   if(execstatus != OCI_SUCCESS)
   {
     OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
     errcode = OCIERROR(p,execstatus);
     if(errcode == NOT_SERIALIZABLE)
     {
       return(RECOVERR);
     }
     else if (errcode == RECOVERR)
     {
       return(RECOVERR);
     }
     else if (errcode == SNAPSHOT_TOO_OLD)
     {
       return(RECOVERR);
     }
     else
     {
       TPCCErr("DB error in delivery transaction cursor d4
%d\n",errcode);
        return (ERR_DB_ERROR);
     }
   }

   OCIAttrGet(dctx-
>curd4,OCI_HTYPE_STMT,&rcount,NULL,OCI_ATTR_ROW_COUNT,p->errhp);
   /* add up amounts */
   count=0;
   for (i=0;i<rpc;i++)
   {
     dctx->amt[i]=0;
     for (j=0;j<actx->ol_cnt[i];j++)
       if (actx->ol_amt_rcode[i][j] == 0)
       {
         dctx->amt[i] = dctx->amt[i] + actx->ol_amt[i][j];
         count = count+1;
       }
   }
   if (rcount > rpc*NITEMS)
   {
     TPCCErr( "Error in TPC-C server %d: %d ordnrs updated, %d
ordl updated\n",
              proc_no, rpc, rcount);
   }


#if defined(ISO5) || defined(ISO6)
   printf 9"d_id:amount\n";
   for (i=0;i<rpc;i++)
     printf ("%d:%.2f ", dctx->d_id[i], (float)dctx->amt[i]/100);
   printf ("\n");
#endif


   /* array update of customer table */

#if defined(ISO5) || defined(ISO6)
   execstatus=OCIStmtExecute(p->tpcsvc,dctx->curd6,p-
>errhp,rpc,0,0,0,
                                  OCI_DEFAULT);
#else
   execstatus=OCIStmtExecute(p->tpcsvc,dctx->curd6,p-
>errhp,rpc,0,0,0,
                                  OCI_COMMIT_ON_SUCCESS |
OCI_DEFAULT);
#endif



   if(execstatus != OCI_SUCCESS)
   {
     OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
     errcode = OCIERROR(p,execstatus);
     if(errcode == NOT_SERIALIZABLE)
     {
       return (RECOVERR);
     }
     else if (errcode == RECOVERR)
     {
       return (RECOVERR);
     }
     else if (errcode == SNAPSHOT_TOO_OLD)
     {
       return (RECOVERR);
     }
     else
     {
       TPCCErr("DB error in delivery transaction cursor d6
%d\n",errcode);
        return (ERR_DB_ERROR);
     }
   }

   OCIAttrGet(dctx-
>curd6,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROW_COUNT,p->errhp);

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


#if defined(ISO5) || defined(ISO6)
   sysdate;

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

#ifdef ISO6
   printf ("Delivery ISO6 is rolling back.\n";
   OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
#endif

#ifdef ISO5
   OCITransCommit(p->tpcsvc,p->errhp,OCI_DEFAULT);
#endif

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


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

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

   return (ERR_DB_SUCCESS);

}



void tkvcddone (delctx *pdctx)
{
  delctx dctx = *pdctx;

#if defined(ISO) || defined(ISO5) || defined(ISO6) || defined(ISO8)
  OCIHandleFree((dvoid *)dctx->curd0,OCI_HTYPE_STMT);
#endif

  if(NULL != dctx.curd1)
    OCIHandleFree((dvoid *)dctx.curd1,OCI_HTYPE_STMT);
  if(NULL != dctx.curd2)
    OCIHandleFree((dvoid *)dctx.curd2,OCI_HTYPE_STMT);
  if(NULL != dctx.curd3)
    OCIHandleFree((dvoid *)dctx.curd3,OCI_HTYPE_STMT);
  if(NULL != dctx.curd4)
    OCIHandleFree((dvoid *)dctx.curd4,OCI_HTYPE_STMT);
  if(NULL != dctx.curd5)
    OCIHandleFree((dvoid *)dctx.curd5,OCI_HTYPE_STMT);
  if(NULL != dctx.curd6)
    OCIHandleFree((dvoid *)dctx.curd6,OCI_HTYPE_STMT);
}

/*
---------------------------------------------------------------------
----------
NEW ORDER TRANSACTION
---------------------------------------------------------------------
----------
*/

#define NOSQLTXT2ops "UPDATE stok SET s_order_cnt = s_order_cnt +
1, \
   s_ytd = s_ytd + :ol_quantity, s_remote_cnt = s_remote_cnt +
:s_remote, \
   s_quantity = s_quantity - :ol_quantity + \
   DECODE (SIGN (s_quantity - :ol_quantity - 10), -1, 91, 0) \
   WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id"


#define NOSQLTXT2 "BEGIN initnew.new_init(:idx1arr); END;"



int tkvcninit (NewOrderData *pNew,
               OraContext *p)
{
```

```
  newctx *nctx = &(p->nctx);
  newtemp *ntemp = &(p->tempvars.new);
  int i;
  int execstatus;
  int errcode;
  text stmbuf[16384];

  memset(nctx,(char)0,sizeof(newctx));
  nctx->cs = 1;
  nctx->norow=0;
  nctx->curn1 = NULL;
  nctx->curn2 = NULL;
  for (i = 0; i < 100; i++)
    (nctx->curn3)[i] = NULL;
  nctx->curn4 = NULL;

  for(i=0;i<NITEMS;i++) {
    OCIERROR(p, OCIDescriptorAlloc(p->tpcenv,
                                          (dvoid**)&nctx-
>s_rowid_ptr[i],

OCI_DTYPE_ROWID,0,(dvoid**)0));
  }
  nctx->w_id_ind = TRUE;
  nctx->w_id_len = sizeof(pNew->w_id);
  nctx->d_id_ind = TRUE;
  nctx->d_id_len = sizeof(pNew->d_id);
  nctx->c_id_ind = TRUE;
  nctx->c_id_len = sizeof(pNew->c_id);
  nctx->o_all_local_ind = TRUE;
  nctx->o_all_local_len = sizeof(pNew->o_all_local);
  nctx->o_ol_cnt_ind = TRUE;
  nctx->o_ol_cnt_len = sizeof(pNew->o_ol_cnt);
  nctx->w_tax_ind = TRUE;
  nctx->w_tax_len = 0;
  nctx->d_tax_ind = TRUE;
  nctx->d_tax_len = 0;
  nctx->o_id_ind = TRUE;
  nctx->o_id_len = sizeof(pNew->o_id);
  nctx->c_discount_ind = TRUE;
  nctx->c_discount_len = 0;
  nctx->c_credit_ind = TRUE;
  nctx->c_credit_len = 0;
  nctx->c_last_ind = TRUE;
  nctx->c_last_len = 0;
  nctx->retries_ind = TRUE;
  nctx->retries_len = sizeof(ntemp->n_retry);
  nctx->cr_date_ind = TRUE;
  nctx->cr_date_len = sizeof(ntemp->cr_date);

  /* open first cursor */
  OCIERROR(p,OCIHandleAlloc(p->tpcenv,(dvoid **)(&nctx->curn1),
                                          OCI_HTYPE_STMT, 0,
(dvoid**)0));
  if(ERR_DB_ERROR == getfile("tkvcpnew.sql",stmbuf))
  {
    TPCCErr("Error opening the file tkvcpnew.sql");
    return ERR_DB_ERROR;
  }

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

  /* bind variables */

  OCIBNDR(nctx->curn1, nctx->w_id_bp, p, ":w_id",ADR(pNew->w_id),
           SIZ(pNew->w_id),
          SQLT_INT, &nctx->w_id_ind, &nctx->w_id_len, &nctx-
>w_id_rc);
  OCIBNDR(nctx->curn1, nctx->d_id_bp, p, ":d_id",ADR(pNew->d_id),
           SIZ(pNew->d_id),
          SQLT_INT, &nctx->d_id_ind, &nctx->d_id_len, &nctx-
>d_id_rc);
  OCIBNDR(nctx->curn1, nctx->c_id_bp, p, ":c_id",ADR(pNew->c_id),
           SIZ(pNew->c_id),
          SQLT_INT, &nctx->c_id_ind, &nctx->c_id_len, &nctx-
>c_id_rc);
  OCIBNDR(nctx->curn1, nctx->o_all_local_bp, p, ":o_all_local",
          ADR(pNew->o_all_local), SIZ(pNew->o_all_local),SQLT_INT,
            &nctx->o_all_local_ind,
          &nctx->o_all_local_len, &nctx->o_all_local_rc);
  OCIBNDR(nctx->curn1, nctx->o_all_cnt_bp, p, ":o_ol_cnt",
          ADR(pNew->o_ol_cnt), SIZ(pNew->o_ol_cnt),SQLT_INT,
            &nctx->o_ol_cnt_ind, &nctx->o_ol_cnt_len, &nctx-
>o_ol_cnt_rc);


  OCIBNDR(nctx->curn1, nctx->w_tax_bp, p, ":w_tax",ADR(ntemp-
>w_tax),
           SIZ(ntemp->w_tax),
          SQLT_FLT, &nctx->w_tax_ind, &nctx->w_tax_len, &nctx-
>w_tax_rc);
  OCIBNDR(nctx->curn1, nctx->d_tax_bp, p, ":d_tax",ADR(ntemp-
>d_tax),
           SIZ(ntemp->d_tax),
          SQLT_FLT, &nctx->d_tax_ind, &nctx->d_tax_len, &nctx-
>d_tax_rc);
  OCIBNDR(nctx->curn1, nctx->o_id_bp, p, ":o_id",ADR(pNew->o_id),
           SIZ(pNew->o_id),
          SQLT_INT, &nctx->o_id_ind, &nctx->o_id_len, &nctx-
>o_id_rc);
  OCIBNDR(nctx->curn1, nctx->c_discount_bp, p, ":c_discount",
          ADR(ntemp->c_discount), SIZ(ntemp->c_discount),SQLT_FLT,
            &nctx->c_discount_ind, &nctx->c_discount_len, &nctx-
>c_discount_rc);

  OCIBNDR(nctx->curn1, nctx->c_credit_bp, p, ":c_credit",pNew-
>c_credit,
           SIZ(pNew->c_credit),SQLT_CHR,
            &nctx->c_credit_ind, &nctx->c_credit_len, &nctx-
>c_credit_rc);
  OCIBNDR(nctx->curn1, nctx->c_last_bp, p, ":c_last",pNew->c_last,
            SIZ(pNew->c_last),
          SQLT_STR, &nctx->c_last_ind, &nctx->c_last_len, &nctx-
>c_last_rc);
  OCIBNDR(nctx->curn1, nctx->retries_bp, p, ":retry",ADR(ntemp-
>n_retry),
           SIZ(ntemp->n_retry),SQLT_INT,
            &nctx->retries_ind, &nctx->retries_len, &nctx-
>retries_rc);
  OCIBNDR(nctx->curn1, nctx->cr_date_bp, p, ":cr_date",ADR(ntemp-
>cr_date),
           SIZ(ntemp->cr_date), SQLT_ODT,
            &nctx->cr_date_ind, &nctx->cr_date_len, &nctx-
>cr_date_rc);

  OCIBNDRAA(nctx->curn1, nctx->ol_i_id_bp, p, ":ol_i_id",ntemp-
>nol_i_id,
           SIZ(int), SQLT_INT, nctx->nol_i_id_ind, nctx-
>nol_i_id_len,
             nctx->nol_i_id_rcode, NITEMS,&nctx->nol_i_count);

  OCIBNDRAA(nctx->curn1, nctx->ol_supply_w_id_bp, p,
":ol_supply_w_id",
             ntemp->nol_supply_w_id, SIZ(int), SQLT_INT,
             nctx->nol_supply_w_id_ind, nctx->nol_supply_w_id_len,
             nctx->nol_supply_w_id_rcode, NITEMS,&nctx-
>nol_s_count);

  OCIBNDRAA(nctx->curn1, nctx->ol_quantity_bp, p, ":ol_quantity",
             ntemp->nol_quantity, SIZ(int), SQLT_INT, nctx-
>nol_quantity_ind,
             nctx->nol_quantity_len, nctx->nol_quantity_rcode,
NITEMS,
             &nctx->nol_q_count);

  OCIBNDRAA(nctx->curn1, nctx->i_price_bp, p, ":i_price",
             ntemp->i_price, SIZ(int), SQLT_INT, nctx->i_price_ind,
             nctx->i_price_len, nctx->i_price_rcode, NITEMS,
             &nctx->nol_item_count);

  OCIBNDRAA(nctx->curn1, nctx->i_name_bp, p, ":i_name",
             ntemp->i_name, SIZ(pNew->o_ol[0].i_name), SQLT_STR,
             nctx->i_name_ind,
             nctx->i_name_len, nctx->i_name_rcode, NITEMS,
             &nctx->nol_name_count);

  OCIBNDRAA(nctx->curn1, nctx->s_quantity_bp, p, ":s_quantity",
             ntemp->s_quantity, SIZ(int), SQLT_INT, nctx-
>s_quant_ind,
             nctx->s_quant_len, nctx->s_quant_rcode, NITEMS,
             &nctx->nol_qty_count);

  OCIBNDRAA(nctx->curn1, nctx->s_bg_bp, p, ":brand_generic",
             ntemp->brand_generic, SIZ(char), SQLT_CHR, nctx-
>s_bg_ind,
             nctx->s_bg_len, nctx->s_bg_rcode, NITEMS,
             &nctx->nol_bg_count);

  OCIBNDRAA(nctx->curn1, nctx->ol_amount_bp, p, ":ol_amount",
             ntemp->nol_amount, SIZ(int), SQLT_INT, nctx-
>nol_amount_ind,
             nctx->nol_amount_len, nctx->nol_amount_rcode, NITEMS,
             &nctx->nol_am_count);

  OCIBNDRAA(nctx->curn1, nctx->s_remote_bp, p, ":s_remote",
             nctx->s_remote, SIZ(int), SQLT_INT, nctx->s_remote_ind,
             nctx->s_remote_len, nctx->s_remote_rcode, NITEMS,
             &nctx->s_remote_count);




  /* open second cursor */
  OCIERROR(p,OCIHandleAlloc(p->tpcenv, (dvoid **)(&nctx->curn2),
                                          OCI_HTYPE_STMT, 0,
(dvoid**)0));
  sprintf ((char *) stmbuf, NOSQLTXT2);
  OCIERROR(p,OCIStmtPrepare(nctx->curn2, p->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];
  ub2 idx1arr_rcode[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;
  pNew->o_ol_cnt=NITEMS;

/* Bind array */
  OCIBNDRAA(nctx-
>curn2,idx1arr_bp,p,":idx1arr",idx1arr,SIZ(int),SQLT_INT,
```

```
    idx1arr_ind,idx1arr_len,idx1arr_rcode,NITEMS,&idx1arr_count);

  execstatus = OCIStmtExecute(p->tpcsvc,nctx->curn2,p-
>errhp,1,0,0,0,
                                    OCI_DEFAULT);
  if(execstatus != OCI_SUCCESS)
  {
   OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
   errcode = OCIERROR(p,execstatus);
   TPCCErr("DB error in neworder transaction cursor n2
%d\n",errcode);
   return ERR_DB_ERROR;
  }
}


return (ERR_DB_SUCCESS);

}



int tkvcn (NewOrderData *pNew, OraContext *p)
{
  int statusCnt;
  int execstatus;
  int errcode;
  newctx *nctx = &(p->nctx);
  newtemp *ntemp = &(p->tempvars.new);

  int proc_no = 0;
  int retries = 0;
  int i;
  int rcount;

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

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

  for (i = 0; i < pNew->o_ol_cnt; i++) {
    if (ntemp->nol_supply_w_id[i] != pNew->w_id) {
      nctx->s_remote[i] = 1;
      pNew->o_all_local = 0;
    }
    else
      nctx->s_remote[i] = 0;
  }

   nctx->w_id_ind = TRUE;
   nctx->w_id_len = sizeof(pNew->w_id);
   nctx->d_id_ind = TRUE;
   nctx->d_id_len = sizeof(pNew->d_id);
   nctx->c_id_ind = TRUE;
   nctx->c_id_len = sizeof(pNew->c_id);
   nctx->o_all_local_ind = TRUE;
   nctx->o_all_local_len = sizeof(pNew->o_all_local);
   nctx->o_ol_cnt_ind = TRUE;
   nctx->o_ol_cnt_len = sizeof(pNew->o_ol_cnt);
   nctx->w_tax_ind = TRUE;
   nctx->w_tax_len = 0;
   nctx->d_tax_ind = TRUE;
   nctx->d_tax_len = 0;
   nctx->o_id_ind = TRUE;
   nctx->o_id_len = sizeof(pNew->o_id);
   nctx->c_discount_ind = TRUE;
   nctx->c_discount_len = 0;
   nctx->c_credit_ind = TRUE;
   nctx->c_credit_len = 0;
   nctx->c_last_ind = TRUE;
   nctx->c_last_len = 0;
   nctx->retries_ind = TRUE;
   nctx->retries_len = sizeof(retries);
   nctx->cr_date_ind = TRUE;
   nctx->cr_date_len = sizeof(ntemp->cr_date);



   /* this is the row count */
   rcount = pNew->o_ol_cnt;
   nctx->nol_i_count = pNew->o_ol_cnt;
   nctx->nol_q_count = pNew->o_ol_cnt;
   nctx->nol_s_count = pNew->o_ol_cnt;
   nctx->s_remote_count = pNew->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;
   nctx->s_data_count = pNew->o_ol_cnt;
   nctx->i_data_count = pNew->o_ol_cnt;

   /* initialization for array operations */
   for (i = 0; i < pNew->o_ol_cnt; i++) {
      nctx->ol_w_id[i] = pNew->w_id;
      nctx->ol_d_id[i] = pNew->d_id;
      nctx->ol_number[i] = i + 1;
      nctx->null_date_ind[i]= TRUE;
      nctx->nol_i_id_ind[i] = 0;
      nctx->nol_supply_w_id_ind[i] = TRUE;
      nctx->nol_quantity_ind[i] = TRUE;
      nctx->nol_amount_ind[i] = TRUE;
      nctx->ol_w_id_ind[i] = TRUE;
      nctx->ol_d_id_ind[i] = TRUE;
      nctx->ol_o_id_ind[i] = TRUE;
      nctx->ol_number_ind[i] = TRUE;
      nctx->ol_dist_info_ind[i] = TRUE;
      nctx->s_remote_ind[i] = TRUE;
      nctx->s_data_ind[i] = TRUE;
      nctx->i_data_ind[i] = TRUE;
      nctx->s_quant_ind[i] = TRUE;
      nctx->s_bg_ind[i] = TRUE;
      nctx->cons_ind[i] = TRUE;
      nctx->s_rowid_ind[i] = TRUE;
      nctx->nol_i_id_len[i] = sizeof(int);
      nctx->nol_supply_w_id_len[i] = sizeof(int);
      nctx->nol_quantity_len[i] = sizeof(int);
      nctx->nol_amount_len[i] = sizeof(int);
      nctx->ol_w_id_len[i] = sizeof(int);
      nctx->ol_d_id_len[i] = sizeof(int);
      nctx->ol_o_id_len[i] = sizeof(int);
      nctx->ol_number_len[i] = sizeof(int);
      nctx->ol_dist_info_len[i] = nctx->s_dist_info_len[i];
      nctx->null_date_len[i]=sizeof(OCIDate);
      nctx->s_remote_len[i] = sizeof(int);
      nctx->s_data_len[i] = sizeof(int);
      nctx->i_data_len[i] = sizeof(int);
      nctx->s_quant_len[i] = sizeof(int);
      nctx->s_rowid_len[i] = sizeof(nctx->s_rowid_ptr[0]);
      nctx->cons_len[i] = sizeof(int);
      nctx->i_name_len[i]=0;
      nctx->s_bg_len[i] = 0;
   }
   for (i = pNew->o_ol_cnt; i < NITEMS; i++) {
      nctx->nol_i_id_ind[i] = NA;
      nctx->nol_supply_w_id_ind[i] = NA;
      nctx->nol_quantity_ind[i] = NA;
      nctx->nol_amount_ind[i] = NA;
      nctx->ol_w_id_ind[i] = NA;
      nctx->ol_d_id_ind[i] = NA;
      nctx->ol_o_id_ind[i] = NA;
      nctx->ol_number_ind[i] = NA;
      nctx->ol_dist_info_ind[i] = NA;
      nctx->null_date_ind[i]= NA;
      nctx->s_remote_ind[i] = NA;
      nctx->s_data_ind[i] = NA;
      nctx->i_data_ind[i] = NA;
      nctx->s_quant_ind[i] = NA;
      nctx->s_bg_ind[i] = NA;
      nctx->cons_ind[i] = NA;
      nctx->s_rowid_ind[i] = NA;

      nctx->nol_i_id_len[i] = 0;
      nctx->nol_supply_w_id_len[i] = 0;
      nctx->nol_quantity_len[i] = 0;
      nctx->nol_amount_len[i] = 0;
      nctx->ol_w_id_len[i] = 0;
      nctx->ol_d_id_len[i] = 0;
      nctx->ol_o_id_len[i] = 0;
      nctx->ol_number_len[i] = 0;
      nctx->ol_dist_info_len[i] = 0;
      nctx->null_date_len[i]= 0;
      nctx->s_remote_len[i] = 0;
      nctx->i_data_len[i] = 0;
      nctx->s_data_len[i] = 0;
      nctx->s_quant_len[i] = 0;
      nctx->s_rowid_len[i] = 0;
      nctx->cons_len[i] = 0;
      nctx->i_name_len[i]=0;
      nctx->s_bg_len[i] = 0;
   }

   execstatus = OCIStmtExecute(p->tpcsvc,nctx->curn1,p-
>errhp,1,0,0,0,
                                       OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
   if(execstatus != OCI_SUCCESS) {
     OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
     errcode = OCIERROR(p,execstatus);
     if(errcode == NOT_SERIALIZABLE)
     {
          retries++;
          return (RECOVERR);
     }
     else if (errcode == RECOVERR)
     {
          retries++;
          return (RECOVERR);
     }
     else if (errcode == SNAPSHOT_TOO_OLD)
     {
          retries++;
          return (RECOVERR);
     }
     else
     {
          return (ERR_DB_ERROR);
     }
   }


   /* did the txn succeed? */
   /* sth added return of ERR_DB_NOT_COMMITED for Invalid Item */
   if (rcount != pNew->o_ol_cnt)
   {
    statusCnt = rcount - pNew->o_ol_cnt;
    pNew->o_ol_cnt = rcount;
    return (ERR_DB_NOT_COMMITED);
   }



#ifdef ISO1
   else {
     OCITransCommit(p->tpcsvc, p->errhp, OCI_DEFAULT);
```

```
   }
#endif

#if defined(ISO1) || defined(ISO7)
   sysdate (sdate);
   printf ("New Order completed at: %s\n", sdate);
#endif


/* calculate total amount */
  pNew->total_amount = 0.0;
  for (i=0;i<pNew->o_ol_cnt;i++)
  {
   if (nctx->nol_amount_ind[i] != NA)
   {
     pNew->total_amount += ntemp->nol_amount[i];
   }
  }
  pNew->total_amount *= ((float)(1-ntemp->c_discount)) *
(float)(1.0 + ((float)(ntemp->d_tax))+((float)(ntemp->w_tax)));
  pNew->total_amount = pNew->total_amount/100;


  return (ERR_DB_SUCCESS);

}



void tkvcndone (newctx *pnctx)

{
   int i;
   newctx nctx = *pnctx;

   if(NULL != nctx.curn1)
     OCIHandleFree((dvoid *)nctx.curn1,OCI_HTYPE_STMT);
   if(NULL != nctx.curn2)
     OCIHandleFree((dvoid *)nctx.curn2,OCI_HTYPE_STMT);
   for (i = 0; i < 10; i++)
     if(NULL != ((nctx.curn3)[i]))
       OCIHandleFree((dvoid *)(nctx.curn3)[i],OCI_HTYPE_STMT);
   if(NULL != nctx.curn4)
     OCIHandleFree((dvoid *)nctx.curn4,OCI_HTYPE_STMT);

}


--------------------------
tpcc.c
--------------------------
/*+       FILE:     TPCC.C
 *                  Microsoft TPC-C Kit Ver. 3.00.000
 *                  Audited 08/23/96    By Francois Raab
 *
 *                  Copyright Microsoft, 1996
 *                  Copyright Digital Equipment Corp., 1997
 *
 *        PURPOSE:  Main module for TPCC.DLL which is an ISAPI
service dll.
 *        Author:             Philip Durr
 *                                  philipdu@Microsoft.com
 *
 *        MODIFICATIONS:
 *
 *                  Routines substantially modified by:
 *                            Anne Bradley        Digital Equipment
Corp.
 *                            Bill Carr Digital Equipment Corp.
 *
 */
/*+*****************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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


#include <windows.h>
#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
#include <io.h>

#define TPCC_C

#include <tpccerr.h>
#include <tpccstruct.h>
#include <tpccapi.h>
#include <httpext.h>

#include <tpcc.h>
#include <web_ui.h>

/* FUNCTION: void FormatString(char *szDest, char *szPic, char
*szSrc)
 *
 * PURPOSE:         This function formats a character string for
inclusion in the
 *                  HTML formatted page being constructed.
 *
 * ARGUMENTS:       char      *szDest
 *          Destination buffer where
 *
 *          formatted string is to be
 *                                                              placed
 *                  char      *szPic                          picture
string which describes
 *                                                              how
character value is to be
 *
 *          formatted.
 *                  char      *szSrc
 *          character string value.
 *
 * RETURNS:         None
 *
 * COMMENTS:        This functions is used to format TPC-C phone
and zip value
 *                  strings.
 *
 */

void FormatString(char *szDest, char *szPic, char *szSrc)
{
  while( *szPic )
  {
    if ( *szPic == 'X' )
    {
      if ( *szSrc )
          *szDest++ = *szSrc++;
      else
          *szDest++ = ' ';
    }
    else
      *szDest++ = *szPic;
    szPic++;
  }
  *szDest = 0;

  return;
}

/* FUNCTION: int ParseNewOrderQuery( char *pProcessedQuery[],
 *                                        NewOrderData
*pNewOrderData )
 *
 * PURPOSE:         This function extracts and validates the new
order query
 *                  from an http command string.
 *
 * ARGUMENTS:       char      *pProcessedQuery[]  array of char*
that points to
 *                                                              the
value of each name-value
 *                                                              pair.
 *                  NewOrderData  *pNewOrderData  pointer to new
order data
 *          structure
 *
 * RETURNS:         int       ERR_SUCCESS                     input
data successfully parsed
 *                            error_code                      reason
for failure
 *
 * COMMENTS:        None
 *
 */
```

```
int ParseNewOrderQuery(char *pQueryString, NewOrderData
*pNewOrderData)
{
  char    *ptr;
  int             i;
  short   items;
  char    *pProcessedQuery[MAXNEWORDERVALS];


  PARSE_QUERY_STRING(pQueryString, MAXNEWORDERVALS,
                       newOrderStrs, pProcessedQuery);

  if ( !GetValuePtr(pProcessedQuery, DID, &ptr) )
    return ERR_NEWORDER_FORM_MISSING_DID;

  GetNumeric(ptr, &pNewOrderData->d_id);
  if(0 == pNewOrderData->d_id)
    return ERR_NEWORDER_DISTRICT_INVALID;


  if ( !GetValuePtr(pProcessedQuery, CID, &ptr) )
    return ERR_NEWORDER_CUSTOMER_KEY;

  if( !GetNumeric(ptr, &pNewOrderData->c_id))
    return ERR_NEWORDER_CUSTOMER_INVALID;

  pNewOrderData->o_all_local = 1;


  for(i=0, items=0; i<15; i++)
  {
    if( !GetValuePtr(pProcessedQuery, i*3+IID00, &ptr))
      return ERR_NEWORDER_MISSING_IID_KEY;
    if(*ptr != '&' && *ptr)
    {
      if(!GetNumeric(ptr, &pNewOrderData->o_ol[items].ol_i_id))
          return ERR_NEWORDER_ITEMID_INVALID;

      if(!GetValuePtr(pProcessedQuery, i*3+SP00, &ptr))
          return ERR_NEWORDER_MISSING_SUPPW_KEY;
      if(!GetNumeric(ptr, &pNewOrderData-
>o_ol[items].ol_supply_w_id))
          return ERR_NEWORDER_SUPPW_INVALID;
      if ( pNewOrderData->o_all_local &&
           pNewOrderData->o_ol[items].ol_supply_w_id !=
           pNewOrderData->w_id )
          pNewOrderData->o_all_local = 0;
      if(!GetValuePtr(pProcessedQuery, i*3+QTY00, &ptr))
          return ERR_NEWORDER_MISSING_QTY_KEY;
      if(!GetNumeric(ptr, &pNewOrderData->o_ol[items].ol_quantity))
          return ERR_NEWORDER_QTY_INVALID;
      if ( pNewOrderData->o_ol[items].ol_i_id >= 1000000 ||
           pNewOrderData->o_ol[items].ol_i_id < 1 )
          return ERR_NEWORDER_ITEMID_RANGE;
      if ( pNewOrderData->o_ol[items].ol_quantity >= 100 ||
           pNewOrderData->o_ol[items].ol_quantity < 1 )
          return ERR_NEWORDER_QTY_RANGE;
      items++;
    }
    else
    {
      if(!GetValuePtr(pProcessedQuery, i*3+SP00, &ptr))
          return ERR_NEWORDER_MISSING_SUPPW_KEY;
      if(*ptr != '&' && *ptr)
          return ERR_NEWORDER_SUPPW_WITHOUT_ITEMID;

      if(!GetValuePtr(pProcessedQuery, i*3+QTY00, &ptr))
          return ERR_NEWORDER_MISSING_QTY_KEY;
      if(*ptr != '&' && *ptr)
          return ERR_NEWORDER_QTY_WITHOUT_ITEMID;
    }
  }
  if ( items == 0 )
    return ERR_NEWORDER_NOITEMS_ENTERED;

  pNewOrderData->o_ol_cnt = items;

  return ERR_SUCCESS;
}

/* FUNCTION: int ParseOrderStatusQuery( char *pProcessedQuery[],
 *                                                 OrderStatusData
*pOrderStatusData )
 *
 * PURPOSE:         This function extracts and validates the order
status query
 *                  from an http command string.
 *
 * ARGUMENTS:       char      *pProcessedQuery[]  array of char*
that points to
 *                                                        the
value of each name-value
 *                                                        pair.
 *                  OrderStatusData *pOrderStatusData pointer to
new order data
 *
structure
 *
 * RETURNS:         int       ERR_SUCCESS                   input
data successfully parsed
 *                            error_code                    reason
for failure
 *
 * COMMENTS:        None
 *
 */
int ParseOrderStatusQuery(char *pQueryString,
                                          OrderStatusData *pOrderStatusData)
{
  char    szTmp[26];
  char    *ptr;
  char    *pSzTmp;
  char    *pProcessedQuery[MAXORDERSTATUSVALS];


  PARSE_QUERY_STRING(pQueryString, MAXORDERSTATUSVALS,
                       orderStatusStrs, pProcessedQuery);

  if ( !GetValuePtr(pProcessedQuery, DID, &ptr) )
    return ERR_ORDERSTATUS_MISSING_DID_KEY;
  if ( !GetNumeric(ptr, &pOrderStatusData->d_id) )
    return ERR_ORDERSTATUS_DID_INVALID;

  if ( !GetValuePtr(pProcessedQuery, CID, &ptr) )
    return ERR_ORDERSTATUS_MISSING_CID_KEY;

  if ( *ptr == '&' || !(*ptr))
  {
    pSzTmp = szTmp;
    pOrderStatusData->c_id = 0;
    if ( !GetValuePtr(pProcessedQuery, CLT_O, &ptr) )
      return ERR_ORDERSTATUS_MISSING_CLT_KEY;
    while(*ptr != '&' && *ptr)
    {
      *pSzTmp = *ptr;
      pSzTmp++;
      ptr++;
    }
    *pSzTmp = '\0';
    _strupr( szTmp );
    strcpy(pOrderStatusData->c_last, szTmp);
    if ( strlen(pOrderStatusData->c_last) > 16 )
      return ERR_ORDERSTATUS_CLT_RANGE;
  }
  else
  {
    if (!GetNumeric(ptr, &pOrderStatusData->c_id))
      return ERR_ORDERSTATUS_CID_INVALID;
    if ( !GetValuePtr(pProcessedQuery, CLT_O, &ptr) )
      return ERR_ORDERSTATUS_MISSING_CLT_KEY;
    if ( *ptr != '&' && *ptr)
      return ERR_ORDERSTATUS_CID_AND_CLT;
  }

  return ERR_SUCCESS;
}

/* FUNCTION: int ParsePaymentQuery( char *pProcessedQuery[],
 *                                                 PaymentData
*pPaymentData )
 *
 * PURPOSE:         This function extracts and validates the
payment query
 *                  from an http command string.
 *
 * ARGUMENTS:       char      *pProcessedQuery[]  array of char*
that points to
 *                                                        the
value of each name-value
 *                                                        pair.
 *                  PaymentData  *pPaymentData    pointer to
payment data
 *
         structure
 *
 * RETURNS:         int       ERR_SUCCESS                   input
data successfully parsed
 *                            error_code                    reason
for failure
 *
 * COMMENTS:        None
 *
 */

int ParsePaymentQuery(char *pQueryString, PaymentData
*pPaymentData)
{
  char    szTmp[26];
  char    *ptr;
  char    *pPtr;
  char    *pSzTmp;
  char    *pProcessedQuery[MAXPAYMENTVALS];


  PARSE_QUERY_STRING(pQueryString, MAXPAYMENTVALS,
                       paymentStrs, pProcessedQuery);

  if ( !GetValuePtr(pProcessedQuery, DID, &ptr) )
    return ERR_PAYMENT_MISSING_DID_KEY;
  if ( !GetNumeric(ptr, &pPaymentData->d_id) )
    return ERR_PAYMENT_DISTRICT_INVALID;

  if ( !GetValuePtr(pProcessedQuery, CID, &ptr) )
    return ERR_PAYMENT_MISSING_CID_KEY;


  if(*ptr == '&' || !(*ptr))
  {
    pPaymentData->c_id = 0;
    pSzTmp = szTmp;
    if ( !GetValuePtr(pProcessedQuery, CLT_P, &ptr) )
      return ERR_PAYMENT_MISSING_CLT;
    if (*ptr == '&' || !(*ptr))
      return ERR_PAYMENT_MISSING_CID_CLT;
    while(*ptr != '&' && *ptr)
```

```
    {
      *pSzTmp = *ptr;
      pSzTmp++;
      ptr++;
    }
    *pSzTmp = '\0';
    _strupr( szTmp );

    strcpy(pPaymentData->c_last, szTmp);
    if ( strlen(pPaymentData->c_last) > 16 )
      return ERR_PAYMENT_LAST_NAME_TO_LONG;
  }
  else
  {
    if (!GetNumeric(ptr, &pPaymentData->c_id))
       return ERR_PAYMENT_CUSTOMER_INVALID;
    if ( !GetValuePtr(pProcessedQuery, CLT_P, &ptr) )
      return ERR_PAYMENT_MISSING_CLT_KEY;
    if(*ptr != '&' && *ptr)
      return ERR_PAYMENT_CID_AND_CLT;
  }

  if ( !GetValuePtr(pProcessedQuery, CDI, &ptr) )
    return ERR_PAYMENT_MISSING_CDI_KEY;
  if ( !GetNumeric(ptr, &pPaymentData->c_d_id) )
    return ERR_PAYMENT_CDI_INVALID;

  if ( !GetValuePtr(pProcessedQuery, CWI, &ptr) )
    return ERR_PAYMENT_MISSING_CWI_KEY;

  if ( !GetNumeric(ptr, &pPaymentData->c_w_id) )
    return ERR_PAYMENT_CWI_INVALID;



  if ( !GetValuePtr(pProcessedQuery, HAM, &ptr) )
    return ERR_PAYMENT_MISSING_HAM_KEY;

  pPtr = ptr;
  while( *pPtr != '&' && *pPtr)
  {
    if ( *pPtr == '.' )
    {
      pPtr++;
      if ( !*pPtr )
          break;
      if ( *pPtr < '0' || *pPtr > '9' )
          return ERR_PAYMENT_HAM_INVALID;
      pPtr++;
      if ( !*pPtr )
          break;
      if ( *pPtr < '0' || *pPtr > '9' )
          return ERR_PAYMENT_HAM_INVALID;
      if ( !*pPtr )
          return ERR_PAYMENT_HAM_INVALID;
    }
    else if ( *pPtr < '0' || *pPtr > '9' )
      return ERR_PAYMENT_HAM_INVALID;
    pPtr++;
  }

  pPaymentData->h_amount = atof(ptr);
  if ( pPaymentData->h_amount >= 10000.00 || pPaymentData->h_amount
< 0 )
    return ERR_PAYMENT_HAM_RANGE;

  return ERR_SUCCESS;
}

/* FUNCTION: BOOL ReadRegistrySettings(void)
 *
 * PURPOSE:         This function reads the NT registry for startup
parameters.
 *                  There parameters are under the TPCC key.
 *
 * ARGUMENTS:   None
 *
 * RETURNS:         None
 *
 * COMMENTS:        This function also sets up required operation
variables to
 *                  their default value so if registry is not setup
the default
 *                  values will be used.
 *
 */

int ReadRegistrySettings(void)
{
  HKEY    hKey;
  DWORD   size;
  DWORD   type;
  char    szTmp[FILENAMESIZE];
  int   status;
  int     iTmp;


  status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                        0, KEY_READ, &hKey);
  if( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_TPCC_KEY;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "PATH", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_PATH_VALUE;
  strcpy(szTpccLogPath, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Server", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_SERVER_VALUE;
  strcpy(gszServer, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Database", 0, &type, szTmp,
&size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_DATABASE_VALUE;
  strcpy(gszDatabase, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "User", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_USER_VALUE;
  strcpy(gszUser, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Password", 0, &type, szTmp,
&size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_PASSWORD_VALUE;
  strcpy(gszPassword, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "CkptUser", 0, &type, szTmp,
&size);
  if ( status != ERROR_SUCCESS )
    strcpy(gszCkptUser, gszUser);
  else
    strcpy(gszCkptUser, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "CkptPassword", 0, &type, szTmp,
&size);
  if ( status != ERROR_SUCCESS )
    strcpy(gszCkptPassword, gszPassword);
  else
    strcpy(gszCkptPassword, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "LOG", 0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS  && 0 == stricmp(szTmp, "ON") )
    bLog = TRUE;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "MaximumWarehouses", 0, &type,
szTmp, &size);
  if ( status == ERROR_SUCCESS  && 0 != (iTmp = atoi(szTmp)) )
    iMaxWareHouses = iTmp;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "MaxConnections", 0, &type, szTmp,
&size);
  if ( status == ERROR_SUCCESS && 0 != (iTmp = atoi(szTmp)) )
    iMaxConnections = iTmp;

  RegCloseKey(hKey);

  if( RegOpenKeyEx(HKEY_LOCAL_MACHINE,

"SYSTEM\\CurrentControlSet\\Services\\InetInfo\\Parameters",
                        0, KEY_READ, &hKey) != ERROR_SUCCESS )
    return ERR_CANT_FIND_INETINFO_KEY;

  size = sizeof(gdwPoolThreadLimit);
  if ( RegQueryValueEx(hKey, "PoolThreadLimit", 0, &type,
                       (LPBYTE)&gdwPoolThreadLimit, &size) !=
ERROR_SUCCESS )
    return ERR_CANT_FIND_POOLTHREADLIMIT;

  RegCloseKey(hKey);

  return ERR_SUCCESS;

}



--------------------------
tpcc.h
--------------------------
#ifndef TPCC_H
#define TPCC_H

/*+**********************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
```

```
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

/*+
 * Abstract: This is the header file for web_ui.c.  it contains the
 *        function prototypes for the routines that are called
outside web_ui.c
 *
 * Author: A Bradley
 * Creation Date: May 1997
 *
 * Modified history:
 *
 *
 */


#define FILENAMESIZE 256

#if defined WEB_UI_C || defined TPCC_C

void FormatString(char *szDest, char *szPic, char *szSrc);

int ParseNewOrderQuery(char *pQueryString, NewOrderData
*pNewOrderData);
int ParsePaymentQuery(char *pQueryString, PaymentData
*pPaymentData);
int ParseOrderStatusQuery(char *pQueryString,
                          OrderStatusData *pOrderStatusData);
#endif /* defined WEB_UI_C || defined TPCC_C */

BOOL ReadRegistrySettings(void);

/* global variables */
#ifdef TPCC_C
#define GLOBAL(thing,initializer) thing = initializer
#else
#define GLOBAL(thing,initializer) extern thing
#endif /* TPCC_C */

#if defined WEB_UI_C || defined TPCC_C
#endif /* defined WEB_UI_C || defined TPCC_C */
GLOBAL(int iMaxConnections,25);
GLOBAL(BOOL bLog,FALSE);
GLOBAL(int iDeadlockRetry,3);
GLOBAL(char szTpccLogPath[FILENAMESIZE],{'\0'});
GLOBAL(int iMaxWareHouses,500);
GLOBAL(char gszServer[32],{'\0'});
GLOBAL(char gszDatabase[32],"tpcc");
GLOBAL(char gszUser[32],"sa");
GLOBAL(char gszPassword[32],{'\0'});
GLOBAL(char gszCkptUser[32],"sa");
GLOBAL(char gszCkptPassword[32],{'\0'});
GLOBAL(pTransactionPoolStruct gpTransactionPool,{0});

#endif /* TPCC_H */


--------------------------
tpcc_tux.h
--------------------------
/*+*****************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

#ifndef TPCC_TUX_H
#define TPCC_TUX_H

#include <tpccstruct.h>


#define TYPE_NO 1
#define TYPE_OS 2
#define TYPE_PT 3
#define TYPE_SL 4
#define TYPE_GC 5
#define TYPE_DY 6

struct int_gc_wksp {
   long trans;
}


#ifndef _STDL_CALL_def
#define _STDL_CALL_def
#ifdef _WIN32
#define _STDL_CALL_ __stdcall
#else
#define _STDL_CALL_
#endif
#endif


#endif /* TPCC_TUX_H */


--------------------------
tpccapi.h
--------------------------
#ifndef TPCCAPI_H
#define TPCCAPI_H
/*+*****************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1996 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

```
/*+**************************************************************
**********
*****************************  tpccapi.h
*********************************
**************************************************************************
**********
*
**  tpccapi.h:  This header file declares function calls between
TPCC
**            application and server
*
*
*  Authors: Tareef Kawaf and Bill Carr
**
**
**  02-05-97 FWM Added bQueueDelivery flag to startup call.
**  18-Feb-98 WCarr Introduced TPCCAPI V2.0
**
*/


#define DELIVERY_RESPONSE_COUNT 2

int TPCCGetTransportData( pTransportData pTransport );

int TPCCStartup( int iMaxUsers, pTransportData pTransport );
int TPCCStartupDB( int iMaxDBConnections );

int TPCCConnect( pLoginData pLogin );
int TPCCConnectDB( DBContext *pDBC, pLoginData pLogin );

int TPCCDelivery( pDeliveryData pDelivery,
                  pDeliveryData
CompletedDeliveries[DELIVERY_RESPONSE_COUNT] );
int TPCCDeliveryDeferred( pDeliveryData pDelivery );
int TPCCDeliveryDB( DBContext DBC, pDeliveryData pDelivery );

int TPCCNewOrder( pNewOrderData pNewOrder );
int TPCCNewOrderDB( DBContext DBC, pNewOrderData pNewOrder );

int TPCCOrderStatus( pOrderStatusData pOrderStatus );
int TPCCOrderStatusDB( DBContext DBC, pOrderStatusData pOrderStatus
);

int TPCCPayment( pPaymentData pPayment );
int TPCCPaymentDB( DBContext DBC, pPaymentData pPayment );

int TPCCStockLevel( pStockLevelData pStockLevel );
int TPCCStockLevelDB( DBContext DBC, pStockLevelData pStockLevel );

int TPCCCheckpoint( pCheckpointData pCheckpoint );
int TPCCCheckpointDB( DBContext DBC, pCheckpointData pCheckpoint );

int TPCCDisconnect( pCallersContext pCC );
int TPCCDisconnectDB( DBContext DBC, pCallersContext pCC );

int TPCCShutdown( void );
int TPCCShutdownDB( void );

void TPCCDeliveryResponse( int retcode, pDeliveryData pDelivery,
                  pDeliveryData
CompletedDeliveries[DELIVERY_RESPONSE_COUNT] );

void TPCCDeliveryDeferredResponse( int retcode, pDeliveryData
pDelivery );

void TPCCNewOrderResponse( int retcode, pNewOrderData pNewOrder );

void TPCCOrderStatusResponse( int retcode, pOrderStatusData
pOrderStatus );

void TPCCPaymentResponse( int retcode, pPaymentData pPayment );

void TPCCStockLevelResponse( int retcode, pStockLevelData
pStockLevel );

void TPCCResponseComplete( CallersContext *pCC );

void ErrorMessage( CallersContext *pCC, int iError, int iErrorType,
                   char *pszMesasge );

int TPCCGetTransportErrorString( int iErrorCode, int iBufSize, char
*pBuffer );
int TPCCGetDBErrorString( int iErrorCode, int iBufSize, char
*pBuffer );

BOOL TPCCOpenLog( void );

BOOL TPCCCloseLog( void );

void TPCCLog( char *fmt, ... );

void TPCCErr( char *fmt, ... );

void TPCCTransactionErr( pConnData pConn, char *fmt, ... );
#endif /* TPCCAPI_H */


--------------------------
tpccerr.h
--------------------------
#ifndef TPCCERR_H
#define TPCCERR_H

/*        FILE:               TPCCERR.H
 *
 *                            Copyright Microsoft, 1996
 *                            Copyright Digital Equipment Corp.,
1997
 *
 *        PURPOSE:  Header file for ISAPI TPCC.DLL, defines
structures
 *                            and error messages used by tpcc
benchmark code.
 *        Author:             Philip Durr
 *                                    philipdu@Microsoft.com
 *
 *        Modified by:        William D. Carr
 *                                    carr@perfom.enet.dec.com
 *
 */

#pragma message ("FIXME: the error types need to be made DB non-
specific")
#define   ERR_TYPE_WEBDLL
                            1
#define   ERR_TYPE_SQL
                            2
#define   ERR_TYPE_DBLIB
                            3

#define ERR_DB_SUCCESS                                          0
#define ERR_DB_ERROR                                            1
#define ERR_TRANSPORT_ERROR                          2
#define ERR_DB_INTERFACE                             3
#define ERR_DB_DEADLOCK_LIMIT                        4
#define ERR_DB_NOT_COMMITED                          5
#define ERR_DB_DEAD                                  6
#define ERR_DB_PENDING                                          7
#define ERR_DB_NOT_LOGGED_IN                         8
#define ERR_DB_LOGIN_FAILED                          9
#define ERR_DB_USE_FAILED                           10
#define ERR_DB_LOGOUT_FAILED                        11
/* NOTE: Be sure to update MAX_ERR if new error code is added. */
#define ERR_DB_MAX_ERR                                         11

#define VALID_DB_ERR(err) (((err) >= ERR_DB_SUCCESS)&&((err) <=
ERR_DB_MAX_ERR))

#define   ERR_SUCCESS                                        1000
#define   ERR_COMMAND_UNDEFINED                              1001
#define   ERR_NOT_IMPLEMENTED_YET                            1002
#define   ERR_CANNOT_INIT_TERMINAL               1003
#define   ERR_OUT_OF_MEMORY                      1004
#define   ERR_NEW_ORDER_NOT_PROCESSED            1005
#define   ERR_PAYMENT_NOT_PROCESSED              1006
#define   ERR_NO_SERVER_SPECIFIED                            1007
#define   ERR_ORDER_STATUS_NOT_PROCESSED                     1008
#define   ERR_W_ID_INVALID                       1009
#define   ERR_CAN_NOT_SET_MAX_CONNECTIONS                    1010
#define   ERR_NOSUCH_CUSTOMER                    1011
#define   ERR_D_ID_INVALID                       1012
#define   ERR_MAX_CONNECT_PARAM                              1013
#define   ERR_INVALID_SYNC_CONNECTION            1014
#define   ERR_INVALID_TERMID                     1015
#define   ERR_PAYMENT_INVALID_CUSTOMER           1016
#define   ERR_SQL_OPEN_CONNECTION                            1017
#define   ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY   1018
#define   ERR_STOCKLEVEL_THRESHOLD_INVALID       1019
#define   ERR_STOCKLEVEL_THRESHOLD_RANGE                     1020
#define   ERR_STOCKLEVEL_NOT_PROCESSED           1021
#define   ERR_NEWORDER_FORM_MISSING_DID          1022
#define   ERR_NEWORDER_DISTRICT_INVALID          1023
#define   ERR_NEWORDER_DISTRICT_RANGE            1024
#define   ERR_NEWORDER_CUSTOMER_KEY              1025
#define   ERR_NEWORDER_CUSTOMER_INVALID          1026
#define   ERR_NEWORDER_CUSTOMER_RANGE            1027
#define   ERR_NEWORDER_MISSING_IID_KEY           1028
#define   ERR_NEWORDER_ITEM_BLANK_LINES          1029
#define   ERR_NEWORDER_ITEMID_INVALID            1030
#define   ERR_NEWORDER_MISSING_SUPPW_KEY                     1031
#define   ERR_NEWORDER_SUPPW_INVALID             1032
#define   ERR_NEWORDER_MISSING_QTY_KEY           1033
#define   ERR_NEWORDER_QTY_INVALID               1034
#define   ERR_NEWORDER_SUPPW_RANGE               1035
#define   ERR_NEWORDER_ITEMID_RANGE              1036
#define   ERR_NEWORDER_QTY_RANGE                             1037
#define   ERR_PAYMENT_DISTRICT_INVALID           1038
#define   ERR_NEWORDER_SUPPW_WITHOUT_ITEMID      1039
#define   ERR_NEWORDER_QTY_WITHOUT_ITEMID                    1040
#define   ERR_NEWORDER_NOITEMS_ENTERED           1041
#define   ERR_PAYMENT_MISSING_DID_KEY            1042
#define   ERR_PAYMENT_DISTRICT_RANGE             1043
#define   ERR_PAYMENT_MISSING_CID_KEY            1044
#define   ERR_PAYMENT_CUSTOMER_INVALID           1045
#define   ERR_PAYMENT_MISSING_CLT                            1046
#define   ERR_PAYMENT_LAST_NAME_TO_LONG          1047
#define   ERR_PAYMENT_CUSTOMER_RANGE             1048
#define   ERR_PAYMENT_CID_AND_CLT                            1049
#define   ERR_PAYMENT_MISSING_CDI_KEY            1050
#define   ERR_PAYMENT_CDI_INVALID                            1051
#define   ERR_PAYMENT_CDI_RANGE                              1052
#define   ERR_PAYMENT_MISSING_CWI_KEY            1053
#define   ERR_PAYMENT_CWI_INVALID                            1054
#define   ERR_PAYMENT_CWI_RANGE                              1055
#define   ERR_PAYMENT_MISSING_HAM_KEY            1056
#define   ERR_PAYMENT_HAM_INVALID                            1057
#define   ERR_PAYMENT_HAM_RANGE                              1058
#define   ERR_ORDERSTATUS_MISSING_DID_KEY                    1059
#define   ERR_ORDERSTATUS_DID_INVALID            1060
#define   ERR_ORDERSTATUS_DID_RANGE              1061
#define   ERR_ORDERSTATUS_MISSING_CID_KEY                    1062
#define   ERR_ORDERSTATUS_MISSING_CLT_KEY                    1063
#define   ERR_ORDERSTATUS_CLT_RANGE              1064
#define   ERR_ORDERSTATUS_CID_INVALID            1065
```

```
#define   ERR_ORDERSTATUS_CID_RANGE              1066
#define   ERR_ORDERSTATUS_CID_AND_CLT            1067
#define   ERR_DELIVERY_MISSING_OCD_KEY           1068
#define   ERR_DELIVERY_CARRIER_INVALID           1069
#define   ERR_DELIVERY_CARRIER_ID_RANGE          1070
#define ERR_PAYMENT_MISSING_CLT_KEY              1071
#define ERR_CANT_FIND_TPCC_KEY                                 1072
#define ERR_CANT_FIND_INETINFO_KEY               1073
#define ERR_CANT_FIND_POOLTHREADLIMIT            1074
#define ERR_DB_DELIVERY_NOT_QUEUED               1075
#define ERR_DELIVERY_NOT_PROCESSED               1076
#define ERR_TERM_ALLOCATE_FAILED                 1077
#define ERR_PENDING                              1078
#define ERR_CANT_START_FRCDINIT_THREAD           1079
#define ERR_CANT_START_DELIVERY_THREAD           1080
#define ERR_GOVERNOR_VALUE_NOT_FOUND           1081
#define ERR_SERVER_MISMATCH                    1082
#define ERR_DATABASE_MISMATCH                  1083
#define ERR_USER_MISMATCH                      1084
#define ERR_PASSWORD_MISMATCH                  1085
#define ERR_CANT_CREATE_ALL_THREADS_EVENT      1086
#define ERR_CANT_CREATE_FORCE_THRED_STRT_EVENT  1087
#define ERR_CANT_ALLOCATE_THREAD_LOCAL_STORAGE  1088
#define ERR_CANT_SET_THREAD_LOCAL_STORAGE      1089
#define ERR_FORCE_CONNECT_THREAD_FAILED          1090
#define ERR_CANT_FIND_SERVER_VALUE               1091
#define ERR_NO_MESSAGE                                         1092
#define ERR_CANT_FIND_PATH_VALUE                 1093
#define   ERR_CANNOT_CREATE_RESULTS_FILE                       1094
#define ERR_DELIVERY_PIPE_SECURITY               1095
#define ERR_DELIVERY_PIPE_CREATE                 1096
#define ERR_DELIVERY_PIPE_OPEN                                 1097
#define ERR_DELIVERY_PIPE_READ                                 1098
#define ERR_DELIVERY_PIPE_DISCONNECT             1099
#define ERR_CANT_FIND_DATABASE_VALUE             1100
#define ERR_CANT_FIND_USER_VALUE                 1101
#define ERR_CANT_FIND_PASSWORD_VALUE             1102
#define ERR_DELIVERY_OUTPUT_PIPE_WRITE           1103
#define ERR_DELIVERY_OUTPUT_PIPE_READ            1104
#define ERR_DELIVERY_MISSING_QUEUETIME_KEY       1105
#define ERR_DELIVERY_QUEUETIME_INVALID           1106
#define ERR_ALREADY_LOGGED_IN                    1107
#define ERR_INVALID_FORM                         1109
#define ERR_DELIVERY_MUST_CONNECTDB              1110
#define ERR_INVALID_FORM_AND_CMD_NOT_BEGIN       1111
#define ERR_MAX_CONNECTIONS_EXCEEDED             1112
#define ERR_CANNOT_FIND_CONNECTION               1113
#define ERR_CKPT_NOT_INITIALIZED                 1114
#define   ERR_PAYMENT_MISSING_CID_CLT            1115
#define ERR_CANT_FIND_MAXDBCONNECTIONS_VALUE    1116

/* error message structure used in ErrorMessage API */
typedef struct _SERRORMSG
{
  int               iError;             /* error id of message */
  char              szMsg[80];          /* message to sent to
browser */
} SERRORMSG;


#ifdef TPCC_C
SERRORMSG errorMsgs[] =
{
  { ERR_SUCCESS, "Success, no error." },
  { ERR_NO_MESSAGE, "No message string available for the specified
error code." },
  { ERR_COMMAND_UNDEFINED, "Command undefined." },
  { ERR_NOT_IMPLEMENTED_YET, "Not Implemented Yet." },
  { ERR_CANNOT_INIT_TERMINAL, "Cannot initialize client
connection." },
  { ERR_OUT_OF_MEMORY, "Insufficient memory." },
  { ERR_NEW_ORDER_NOT_PROCESSED, "Cannot process new Order form."
},
  { ERR_PAYMENT_NOT_PROCESSED, "Cannot process payment form." },
  { ERR_NO_SERVER_SPECIFIED, "No Server name specified." },
  { ERR_ORDER_STATUS_NOT_PROCESSED, "Cannot process order status
form." },
  { ERR_W_ID_INVALID, "Invalid Warehouse ID." },
  { ERR_CAN_NOT_SET_MAX_CONNECTIONS, "Insufficient memory to
allocate # connections." },
  { ERR_NOSUCH_CUSTOMER, "No such customer." },
  { ERR_D_ID_INVALID, "Invalid District ID Must be 1 to 10." },
  { ERR_MAX_CONNECT_PARAM, "Max client connections exceeded, run
install to increase." },
  { ERR_INVALID_SYNC_CONNECTION, "Invalid Terminal Sync ID." },
  { ERR_INVALID_TERMID, "Invalid Terminal ID." },
  { ERR_PAYMENT_INVALID_CUSTOMER, "Payment Form, No such Customer."
},
  { ERR_SQL_OPEN_CONNECTION, "SQLOpenConnection API Failed." },
  { ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY, "Stock Level missing
Threshold key \"TT*\"." },
  { ERR_STOCKLEVEL_THRESHOLD_INVALID, "Stock Level Threshold
invalid data type range = 1 - 99." },
  { ERR_STOCKLEVEL_THRESHOLD_RANGE, "Stock Level Threshold out of
range, range must be 1 - 99." },
  { ERR_STOCKLEVEL_NOT_PROCESSED, "Stock Level not processed." },
  { ERR_NEWORDER_FORM_MISSING_DID, "New Order missing District key
\"DID*\"." },
  { ERR_NEWORDER_DISTRICT_INVALID, "New Order District ID Invalid
range 1 - 10." },
  { ERR_NEWORDER_DISTRICT_RANGE, "New Order District ID out of
Range. Range = 1 - 10." },
  { ERR_NEWORDER_CUSTOMER_KEY, "New Order missing Customer key
\"CID*\"." },
  { ERR_NEWORDER_CUSTOMER_INVALID, "New Order customer id invalid
data type, range = 1 to 3000." },
  { ERR_NEWORDER_CUSTOMER_RANGE, "New Order customer id out of
range, range = 1 to 3000." },
  { ERR_NEWORDER_MISSING_IID_KEY, "New Order missing Item Id key
\"IID*\"." },
  { ERR_NEWORDER_ITEM_BLANK_LINES, "New Order blank order lines all
orders must be continuous." },
  { ERR_NEWORDER_ITEMID_INVALID, "New Order Item Id is wrong data
type, must be numeric." },
  { ERR_NEWORDER_MISSING_SUPPW_KEY, "New Order missing Supp_W key
\"SP##*\"." },
  { ERR_NEWORDER_SUPPW_INVALID, "New Order Supp_W invalid data type
must be numeric." },
  { ERR_NEWORDER_MISSING_QTY_KEY, "New Order Missing Qty key
\"Qty##*\"." },
  { ERR_NEWORDER_QTY_INVALID, "New Order Qty invalid must be
numeric range 1 - 99." },
  { ERR_NEWORDER_SUPPW_RANGE, "New Order Supp_W value out of range
range = 1 - Max Warehouses." },
  { ERR_NEWORDER_ITEMID_RANGE, "New Order Item Id is out of range.
Range = 1 to 999999." },
  { ERR_NEWORDER_QTY_RANGE, "New Order Qty is out of range. Range =
1 to 99." },
  { ERR_PAYMENT_DISTRICT_INVALID, "Payment District ID is invalid
must be 1 - 10." },
  { ERR_NEWORDER_SUPPW_WITHOUT_ITEMID, "New Order Supp_W field
entered without a corrisponding Item_Id." },
  { ERR_NEWORDER_QTY_WITHOUT_ITEMID, "New Order Qty entered without
a corrisponding Item_Id." },
  { ERR_NEWORDER_NOITEMS_ENTERED, "New Order Blank Items between
items, items must be continuous." },
  { ERR_PAYMENT_MISSING_DID_KEY, "Payment missing District Key
\"DID*\"." },
  { ERR_PAYMENT_DISTRICT_RANGE, "Payment District Out of range,
range = 1 - 10." },
  { ERR_PAYMENT_MISSING_CID_KEY, "Payment missing Customer Key
\"CID*\"." },
  { ERR_PAYMENT_CUSTOMER_INVALID, "Payment Customer data type
invalid, must be numeric." },
  { ERR_PAYMENT_MISSING_CLT, "Payment missing Customer Last Name
Key \"CLT*\"." },
  { ERR_PAYMENT_MISSING_CID_CLT, "Payment entered without Customer
ID or last Name. " },
  { ERR_PAYMENT_LAST_NAME_TO_LONG, "Payment Customer last name
longer than 16 characters." },
  { ERR_PAYMENT_CUSTOMER_RANGE, "Payment Customer ID out of range,
must be 1 to 3000." },
  { ERR_PAYMENT_CID_AND_CLT, "Payment Customer ID and Last Name
entered must be one or other." },
  { ERR_PAYMENT_MISSING_CDI_KEY, "Payment missing Customer district
key \"CDI*\"." },
  { ERR_PAYMENT_CDI_INVALID, "Payment Customer district invalid
must be numeric." },
  { ERR_PAYMENT_CDI_RANGE, "Payment Customer district out of range
must be 1 - 10." },
  { ERR_PAYMENT_MISSING_CWI_KEY, "Payment missing Customer
Warehouse key \"CWI*\"." },
  { ERR_PAYMENT_CWI_INVALID, "Payment Customer Warehouse invalid
must be numeric." },
  { ERR_PAYMENT_CWI_RANGE, "Payment Customer Warehouse out of
range, 1 to Max Warehouses." },
  { ERR_PAYMENT_MISSING_HAM_KEY, "Payment missing Amount key
\"HAM*\"." },
  { ERR_PAYMENT_HAM_INVALID, "Payment Amount invalid data type must
be numeric." },
  { ERR_PAYMENT_HAM_RANGE, "Payment Amount out of range, 0 -
9999.99." },
  { ERR_ORDERSTATUS_MISSING_DID_KEY, "Order Status missing District
key \"DID*\"." },
  { ERR_ORDERSTATUS_DID_INVALID, "Order Status District invalid,
value must be numeric 1 - 10." },
  { ERR_ORDERSTATUS_DID_RANGE, "Order Status District out of range
must be 1 - 10." },
  { ERR_ORDERSTATUS_MISSING_CID_KEY, "Order Status missing Customer
key \"CID*\"." },
  { ERR_ORDERSTATUS_MISSING_CLT_KEY, "Order Status missing Customer
Last Name key \"CLT*\"." },
  { ERR_ORDERSTATUS_CLT_RANGE, "Order Status Customer last name
longer than 16 characters." },
  { ERR_ORDERSTATUS_CID_INVALID, "Order Status Customer ID invalid,
range must be numeric 1 - 3000." },
  { ERR_ORDERSTATUS_CID_RANGE, "Order Status Customer ID out of
range must be 1 - 3000." },
  { ERR_ORDERSTATUS_CID_AND_CLT, "Order Status Customer ID and
LastName entered must be only one." },
  { ERR_DELIVERY_MISSING_OCD_KEY, "Delivery missing Carrier ID key
\"OCD*\"." },
  { ERR_DELIVERY_CARRIER_INVALID, "Delivery Carrier ID invalid must
be numeric 1 - 10." },
  { ERR_DELIVERY_CARRIER_ID_RANGE, "Delivery Carrier ID out of
range must be 1 - 10." },
  { ERR_PAYMENT_MISSING_CLT_KEY, "Payment missing Customer Last
Name key \"CLT*\"." },
  { ERR_DB_ERROR, "A Database error has occurred." },
  { ERR_DELIVERY_NOT_PROCESSED, "Delivery not processed." },
  { ERR_DB_DELIVERY_NOT_QUEUED, "Delivery not queued." },
  { ERR_CANT_FIND_TPCC_KEY, "TPCC key not found in registry." },
  { ERR_CANT_FIND_INETINFO_KEY, "inetinfo key not found in
registry." },
  { ERR_CANT_FIND_POOLTHREADLIMIT, "PoolThreadLimit value not set
in inetinfo\\Parameters key." },
  { ERR_TERM_ALLOCATE_FAILED, "Failed to allocate terminal data
structure." },
  { ERR_DELIVERY_PIPE_SECURITY, "Failed to initialize delivery pipe
security." },
  { ERR_DELIVERY_PIPE_CREATE, "Failed to create delivery pipe." },
  { ERR_DELIVERY_PIPE_OPEN, "Failed to open delivery pipe." },
  { ERR_DELIVERY_PIPE_READ, "Failed to read delivery pipe." },
  { ERR_DELIVERY_PIPE_DISCONNECT, "Failed to start delivery pipe
disconnect thread."},
  { ERR_PENDING, "Transaction pending."},
```

```
  { ERR_CANT_START_FRCDINIT_THREAD, "Can't start Forced
Initialization thread." },
  { ERR_CANT_START_DELIVERY_THREAD, "Can't start delivery thread."
},
  { ERR_GOVERNOR_VALUE_NOT_FOUND, "Governor value not found in
Registry." },
  { ERR_SERVER_MISMATCH, "Server does not match registry value." },
  { ERR_DATABASE_MISMATCH, "Database name does not match registry
value." },
  { ERR_USER_MISMATCH, "User name does not match registry value."
},
  { ERR_PASSWORD_MISMATCH, "Password does not match registry
value." },
  { ERR_CANT_CREATE_ALL_THREADS_EVENT, "Can't create All Threads
Event." },
  { ERR_CANT_CREATE_FORCE_THRED_STRT_EVENT, "Can't create Force
Thread Start Event." },
  { ERR_CANT_ALLOCATE_THREAD_LOCAL_STORAGE, "Can't allocate thread
local storage" },
  { ERR_CANT_SET_THREAD_LOCAL_STORAGE, "Can't set thread local
storage." },
  { ERR_FORCE_CONNECT_THREAD_FAILED, "At least one database connect
call failed, check log files for specific error." },
  { ERR_CANT_FIND_SERVER_VALUE, "Server value not set in TPCC key."
},
  { ERR_CANT_FIND_PATH_VALUE, "PATH value not set in TPCC key." },
  { ERR_CANNOT_CREATE_RESULTS_FILE, "Cannot create results file."
},
  { ERR_CANT_FIND_DATABASE_VALUE, "Database value not set in TPCC
key." },
  { ERR_CANT_FIND_USER_VALUE, "User value not set in TPCC key." },
  { ERR_CANT_FIND_PASSWORD_VALUE, "Password value not set in TPCC
key." },
  { ERR_DELIVERY_OUTPUT_PIPE_WRITE, "Failed to write output
delivery pipe." },
  { ERR_DELIVERY_OUTPUT_PIPE_READ, "Failed to read output delivery
pipe." },
  { ERR_DELIVERY_MISSING_QUEUETIME_KEY, "Delivery queue time
missing from query." },
  { ERR_DELIVERY_QUEUETIME_INVALID, "Delivery queue time is
invalid." },
  { ERR_ALREADY_LOGGED_IN, "TPCCConnectDB has already been called."
},
  { ERR_DB_NOT_LOGGED_IN, "TPCCConnectDB has not yet been called."
},
  { ERR_INVALID_FORM, "The FORM field is missing or invalid." },
  { ERR_DELIVERY_MUST_CONNECTDB, "Synchronous transport requires
delivery server connect to database." },
  { ERR_INVALID_FORM_AND_CMD_NOT_BEGIN, "The FORM field is missing
and CMD is not Begin." },
  { ERR_MAX_CONNECTIONS_EXCEEDED, "The maximum number of
connections has been exceeded." },
  { ERR_CANT_FIND_MAXDBCONNECTIONS_VALUE, "MaxDBConnections value
not set in TPCC key." },
  { ERR_CANNOT_FIND_CONNECTION, "Transport layer unable to find a
DBContext coresponding to the CallersContext." },
  { ERR_CKPT_NOT_INITIALIZED, "The checkpoint subsystem has not
been started." },
  {       0, "" }
};
#else
extern SERRORMSG errorMsgs[];
#endif /* TPCC_C */

#endif /* TPCCERR_H */


--------------------------
tpccstruct.h
--------------------------
#ifndef TPCCSTRUCT_H
#define TPCCSTRUCT_H

/*****************************************************************
**********
*****************************  tpccstruct.h
*****************************
*****************************************************************
**********/
/*
**  tpccstruct.h:  This header file declares data structures for
use in
**            application and server
*/
/* Copyright 1996 Digital Equipment Corporation */
/*
**  Author: Bill Carr
**             (Majority of content from previous work by Ruth
Morgenstein)
**
**
*/

#include <time.h>

#ifdef _WIN32
# ifndef BOOLEAN
#  define BOOLEAN BOOL
# endif
#else
# include <sys/types.h>

# define BOOLEAN int
# define VMS    0
# define LINEMAX 256

# define FALSE   0
# define TRUE    1
#endif

#define MAX_OL  15


#ifdef FFE_DEBUG

# define CALLING_LH 0x0001
# define IN_LH                0x0002
# define IN_RH                0x0004
# define IN_DB                0x0008
# define LEAVING_DB 0x0010
# define LEAVING_RH 0x0020
# define LEAVING_LH 0x0040
# define CALLING_RESP         0x0080
# define UNRESERVING          0x0100

# define ALL_STAGES 0x01ff

  /*                           users * scale * hours * min * txn/no
*/
# define HISTORY_SIZE ((int)( 5000 *   1.2 *     2 *  60 *
2.22222))

# define TRANSACTION_DEBUG_INFO\
  int iStage;\
  DWORD dwThreadId;\
  DWORD dwXPThreadId;\
  int iSynchronous;\
  int iType;\
  int iReserveHistoryId;\
  int iUnreserveHistoryId;\

# define INIT_TRANSACTION(type,pData)\
  gpTransactionPool->iHistoryId++;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iFailure = 0;\
  _ASSERT( gpTransactionPool->iNextFree <= gpTransactionPool-
>iMaxIndex );\
  memset( pData, 0x01, gpTransactionPool->iTransactionSize );\
  pData->iStage = 0;\
  pData->dwThreadId = GetCurrentThreadId();\
  pData->dwXPThreadId = 0;\
  pData->iType = type;\
  pData->iReserveHistoryId = gpTransactionPool->iHistoryId;\
  pData->iUnreserveHistoryId = 0;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iOpCode
= 1;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iReserveHistoryId = gpTransactionPool->iHistoryId;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iUnreserveHistoryId = 0;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iType =
type;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].dwThreadId = pData->dwThreadId;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].dwXPThreadId = pData->dwXPThreadId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].pTrans
= pData;

# define CHECK_TRANSACTION(type,pData)\
  gpTransactionPool->iHistoryId++;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iFailure++;\
  _ASSERT( gpTransactionPool->iNextFree > 0 );\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iFailure++;\
  _ASSERT(((pData->iStage) | ALL_STAGES) == ALL_STAGES);\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iFailure++;\
  if( pData->iSynchronous == 1 )\
    _ASSERT((pData->dwThreadId == GetCurrentThreadId( )));\
  else if( pData->iSynchronous == 0 )\
    _ASSERT((pData->dwXPThreadId == GetCurrentThreadId( )));\
  else\
    _ASSERT(FALSE);\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iFailure++;\
  _ASSERT((pData->iType==type));\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iFailure++;\
  _ASSERT((gpTransactionPool->History[pData-
>iReserveHistoryId].pTrans) == pData);\
  pData->iUnreserveHistoryId = gpTransactionPool->iHistoryId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iOpCode
= 2;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iReserveHistoryId = pData->iReserveHistoryId;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iUnreserveHistoryId = gpTransactionPool->iHistoryId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iType =
type;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].dwThreadId = pData->dwThreadId;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].dwXPThreadId = pData->dwXPThreadId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].pTrans
= pData;

#else /* FFE_DEBUG */

# define TRANSACTION_DEBUG_INFO

# define INIT_TRANSACTION(type,pData)

# define CHECK_TRANSACTION(type,pData)
```

```
#endif /* FFE_DEBUG */

# define NUMBER_POOL_TRANS_TYPES 5
# define DELIVERY_TRANS 0
# define NEW_ORDER_TRANS 1
# define ORDER_STATUS_TRANS 2
# define PAYMENT_TRANS 3
# define STOCK_LEVEL_TRANS 4

#define RESERVE_TRANSACTION_STRUCT(type,pData)\
  EnterCriticalSection( &gpTransactionPool->critSec );\
  pData = gpTransactionPool->index[gpTransactionPool->iNextFree];\
  INIT_TRANSACTION(type,pData);\
  gpTransactionPool->iNextFree++;\
  LeaveCriticalSection( &gpTransactionPool->critSec );

#define UNRESERVE_TRANSACTION_STRUCT(type,pData)\
  EnterCriticalSection( &gpTransactionPool->critSec );\
  CHECK_TRANSACTION(type,pData);\
  gpTransactionPool->index[--gpTransactionPool->iNextFree] =
pData;\
  LeaveCriticalSection( &gpTransactionPool->critSec );

typedef struct
{
  CRITICAL_SECTION critSec;
  int     iNextFree;
#ifdef FFE_DEBUG
  int     iMaxIndex;
  int     iTransactionSize;
  int     iHistoryId;
  struct
  {
    int             iOpCode;
    int             iFailure;
    int             iReserveHistoryId;
    int             iUnreserveHistoryId;
    int             iType;
    DWORD dwThreadId;
    DWORD dwXPThreadId;
    void  *pTrans;
  }                 History[HISTORY_SIZE];
#endif
  void    *index[1];
  char    data[1];
} TransactionPoolStruct, *pTransactionPoolStruct;
```

```
/*
**  Data structures descriptions for IO data for each transaction
type
**
*/

typedef void CallersContext;
typedef void *pCallersContext;
typedef void *DBContext;

#define INVALID_DB_CONTEXT NULL

typedef struct _DBDate {
          int year;     /* 1900 - 2100 */
          int month;              /* 1 - 12 */
          int day;      /* 1 - 31 */
          int hour;     /* 0 - 23 */
          int minute;             /* 0 - 59 */
          int second;             /* 0 - 59 */
} DBDateData, *pDBDateData;

/* Data common to all transactions that represents the connection
to the UI */
/* and the database are built as a macro to reduce duplication. */
#define CONN_DATA \
    TRANSACTION_DEBUG_INFO\
    int     w_id;\
    int     ld_id;\
    CallersContext *pCC;\
    int     status;\
    int     dbstatus;


typedef struct _ConnData
{
    CONN_DATA
} ConnData, *pConnData;

/* DELIVERY is built as a macro so that i_delivery struct is
consistant with */
/* the io_delivery struct.  Note also that the input portion of the
delivery */
/* data can be simply memcpyed from the input to the input/output
struct. */
#define I_DELIVERY \
    CONN_DATA\
    time_t  queue_time;\
    unsigned delta_time;      /* in milliseconds */\
    int     o_carrier_id;

typedef struct _DeliveryDataInput {
    I_DELIVERY
} DeliveryDataInput, *pDeliveryDataInput;

typedef struct _DeliveryData {
    I_DELIVERY                          /* see comment above */
    int     o_id[10];
} DeliveryData, *pDeliveryData;

struct io_order_line {
    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 _NewOrderData {
    CONN_DATA
    int     d_id;
    int     c_id;
    int     o_ol_cnt;
    int     o_all_local;
    struct io_order_line o_ol[MAX_OL];
    DBDateData o_entry_d;
    char    c_last[17];
    char    c_credit[3];
    float  c_discount;
    float  w_tax;
    float  d_tax;
    int     o_id;
    double  tax_n_discount;
    double  total_amount;
} NewOrderData, *pNewOrderData;

struct status_order_line {
    int     ol_supply_w_id;
    int     ol_i_id;
    int     ol_quantity;
    double  ol_amount;
    DBDateData    ol_delivery_d;
};

typedef struct _OrderStatusData {
    CONN_DATA
    BOOLEAN byname;
    int     d_id;
    int     c_id;
    char    c_last[17];
    char    c_first[17];
    char    c_middle[3];
    double  c_balance;
    int     o_id;
    DBDateData    o_entry_d;
    int     o_carrier_id;
    int     o_ol_cnt;
    struct status_order_line    s_ol[MAX_OL];
} OrderStatusData, *pOrderStatusData;

typedef struct _PaymentData {
    CONN_DATA
    BOOLEAN byname;
    int     d_id;
    int     c_id;
    char    c_last[17];
    int     c_w_id;
    int     c_d_id;
    double  h_amount;
    DBDateData 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];
    DBDateData    c_since;
    char    c_credit[3];
    double  c_credit_lim;
    float  c_discount;
    double  c_balance;
    char    c_data[201];
} PaymentData, *pPaymentData;

typedef struct _StockLevelData {
    CONN_DATA
    int     threshold;
    int     low_stock;
} StockLevelData, *pStockLevelData;

typedef struct _CheckpointData {
    CONN_DATA
    int     how_many;
    int     interval;
} CheckpointData, *pCheckpointData;

/*
**  Data structure for input & output data
*/

typedef struct _TransactionData {
    int type;
    union {
          DeliveryData delivery;
          NewOrderData newOrder;
          OrderStatusData orderStatus;
          PaymentData payment;
          StockLevelData stockLevel;
        CheckpointData checkpoint;
   }  info;
} TransactionData, *pTransactionData;


typedef struct _TransportData {
  BOOL              asynchronous;
  BOOL              generic;
  int               num_gc;
  int               num_dy;
  int               num_no;
  int               num_os;
  int               num_pt;
  int               num_sl;
  BOOL              dy_use_transport;
  int               num_dy_servers;
  int               num_queued_deliveries;
  int               num_queued_responses;
} TransportData, *pTransportData;

/* Data structure for passing connection information */
typedef struct _LoginData {
  CONN_DATA
  char                        szServer[32];
  char                        szDatabase[32];
  char                        szUser[32];
  char                        szPassword[32];
  char                        szApplication[32];
} LoginData, *pLoginData;

#endif /* TPCCSTRUCT_H */


--------------------------
tux_cli.c
--------------------------
/*+**************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
```

```
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*
 *  Updated November 20, 2001 - Susan Georgson
*
 *  Converted tpcc_fct.c file to tux_cli.c
*
 *  Changed transaction monitor from DB Web Connector to Tuxedo
*

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

#include <stdlib.h>   /* stg - added for change to Tuxedo */
#include <string.h>
#include <stdio.h>
#include <windows.h>
#include <winbase.h>
#include <process.h>
/* stg - next line not needed for web ora tux app code
#include <stdldefs.h>
#define STDL_SYNCHRONOUS ((void*)0xffffffff)
*/
#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpccerr.h>

/* stg - next two lines not needed for web ora tux app code
#include <tpcc_acmsxp.h>
#include <tpcc_acmsxp_pp.h> */

#include <tpcc.h>

#include <deli_cli.h>
#include <deli_srv.h>

/* tuxedo include files */
#include <atmi.h>

#ifdef FFE_DEBUG
# include <crtdbg.h>
#endif

#define TOTAL_ADMIN_CONNECTIONS 1

#define FILENAMESIZE 256

/* Module globals */
/* Added for Tuxedo */
static DWORD tlsIndex;
static CRITICAL_SECTION TLS_crit_sec;
/* End added section for Tuxedo */


static HANDLE gForceAllThreadsEvent;
static int    gForceAllThreadsCtr;
static int    gInitRetStatus;
static int    giLoginDelay = 0;
static BOOL   gbGeneric = FALSE;

/* stg - this is for acmsxp and is not needed for Tuxedo */
#if 0
{
void _STDL_CALL_ dy_callback( void *);
void _STDL_CALL_ no_callback( void *);
void _STDL_CALL_ os_callback( void *);
void _STDL_CALL_ pt_callback( void *);
void _STDL_CALL_ sl_callback( void *);

struct int_gc_wksp dy = { TYPE_DY };
struct int_gc_wksp no = { TYPE_NO };
struct int_gc_wksp os = { TYPE_OS };
struct int_gc_wksp pt = { TYPE_PT };
struct int_gc_wksp sl = { TYPE_SL };
struct int_gc_wksp gc = { TYPE_GC };

typedef struct {
  int type;
} force_connect_args;

void __cdecl force_connect( void *arglist );
BOOL getGovernorValue( char *group, long *ctr );
#ifdef USE_MUX
stdl_call_dll_init( DWORD why_called );
#endif

#ifdef FFE_DEBUG
pDeliveryData gpDelivery = NULL;
pNewOrderData gpNewOrder = NULL;
pOrderStatusData gpOrderStatus = NULL;
pPaymentData gpPayment = NULL;
pStockLevelData gpStockLevel = NULL;
#endif

void _STDL_CALL_
dy_callback( void *data )
{
  pDeliveryData pDelivery = data;

#ifdef FFE_DEBUG
  _ASSERT( VALID_DB_ERR(pDelivery->status) );
#endif

  TPCCDeliveryDeferredResponse( pDelivery->status, pDelivery );

  /* batch transactions do not return to the UI, so no response
complete */
  /* call is necessary.
                                */
}
void _STDL_CALL_
no_callback( void *data )
{
  pNewOrderData pNewOrder = data;
  pCallersContext pCC;

#ifdef FFE_DEBUG
  _ASSERT( VALID_DB_ERR(pNewOrder->status) );
#endif

  pCC = pNewOrder->pCC;

  TPCCNewOrderResponse( pNewOrder->status, pNewOrder );

  TPCCResponseComplete( pCC );
}
void _STDL_CALL_
os_callback( void *data )
{
  pOrderStatusData pOrderStatus = data;
  pCallersContext pCC;

#ifdef FFE_DEBUG
  _ASSERT( VALID_DB_ERR(pOrderStatus->status) );
#endif

  pCC = pOrderStatus->pCC;

  TPCCOrderStatusResponse( pOrderStatus->status, pOrderStatus );

  TPCCResponseComplete( pCC );
}
void _STDL_CALL_
pt_callback( void *data )
{
  pPaymentData pPayment = data;
  pCallersContext pCC;

#ifdef FFE_DEBUG
  _ASSERT( VALID_DB_ERR(pPayment->status) );
#endif

  pCC = pPayment->pCC;

  TPCCPaymentResponse( pPayment->status, pPayment );

  TPCCResponseComplete( pCC );
}
void _STDL_CALL_
sl_callback( void *data )
{
  pStockLevelData pStockLevel = data;
  pCallersContext pCC;

#ifdef FFE_DEBUG
  _ASSERT( VALID_DB_ERR(pStockLevel->status) );
#endif

  pCC = pStockLevel->pCC;

  TPCCStockLevelResponse( pStockLevel->status, pStockLevel );

  TPCCResponseComplete( pCC );
}
#endif
/* end of #if 0 section commented out for acmsxp...stg */


/* stg - IsTuxInit is added to check if Tuxedo has been initialized
*/
/* If Tuxedo has not been initialized, then Tuxedo is initialized
during */
/* this function.  */
```

```
/*
 * FUNCTION int IsTuxInit
 */
int
IsTuxInit()
{
  TPINIT *tpinitbuf;

  int x = 1;
  int retcode = -1;
  int count = 0;
  static int num_tpinits = 0;

  if(!(TlsGetValue(tlsIndex)))
    {
      EnterCriticalSection(&TLS_crit_sec);
      while(count < 20)
      {
            if(NULL == (tpinitbuf = (TPINIT *) tpalloc("TPINIT",
NULL,

sizeof(TPINIT))))
            {
              TPCCErr("error with tpalloc - %d - %d",
tperrno,count);
            }
            else
            {
              tpinitbuf->flags |= TPMULTICONTEXTS;
              itoa(++num_tpinits, tpinitbuf->cltname, 10);
              retcode = tpinit(tpinitbuf);
              if(-1 != retcode)
              {
                TlsSetValue(tlsIndex, &x);
                tpfree((char*)tpinitbuf);
                break;
              }
              else
              {
                TPCCErr("error with TPINIT - %s (%d) -
%d\n\t\t..%s..",
                        tpstrerror(tperrno),
                        tperrno,
                        count,
                        tpstrerrordetail( tperrordetail( 0 ), 0
));
                tpfree((char*)tpinitbuf);
              }
            }

            count++;
            if(count > 50)
              {
                retcode = -1;
                TPCCErr("exceeded 50 trys in TPINIT");
              }

            Sleep(50);
      }
      LeaveCriticalSection(&TLS_crit_sec);
      Sleep(50);
      if( -1 != retcode)
          return ERR_DB_SUCCESS;
      else
          return(retcode);
    }
  return ERR_DB_SUCCESS;
}

/* stg - end IsTuxInit function */



/* stg - begin Tuxedo change of TPCCDelivery Deferred */
/*
 * FUNCTION int TPCCDelivery
 */

int
TPCCDeliveryDeferred( pDeliveryData ppDelivery )
{
  int retcode = ERR_DB_SUCCESS;

  pDeliveryData retptr;
  int dysiz = sizeof(DeliveryData);

  /* check to see that the database is connected. */
  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
      TPCCErr("IsTuxInit - delivery ");
      return ERR_DB_ERROR;
    }

  /* allocate memory and copy over data */
  if(NULL == ( retptr= (pDeliveryData) tpalloc("CARRAY", NULL,
dysiz)))
    {
      TPCCErr("tp alloc in delivery");
      return ERR_DB_ERROR;
    }
  memcpy( retptr, ppDelivery, dysiz);

  /* Call tuxedo for Delivery */
  retcode = tpcall("dy_transaction", (char *)retptr, dysiz,
                   (char**)&retptr, &dysiz, TPNOTIME |
TPSIGRSTRT );
  if( -1 == retcode )
    {
      TPCCErr("tpcall - delivery: %d", tperrno);
      tpfree((char*) retptr);
      return ERR_DB_ERROR;
    }
  memcpy(ppDelivery, retptr, dysiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}

/* stg - end Tuxedo change of TPCCDelivery Deferred */



/* stg - begin Tuxedo change of TPCCNewOrder */
/*
 * FUNCTION int TPCCNewOrder
 */
int
TPCCNewOrder( pNewOrderData ppNewOrder )
{
  int retcode = ERR_DB_SUCCESS;
  pNewOrderData retptr;
  int nosiz = sizeof(NewOrderData);
  HANDLE bryh;
  char Brybuffer[100];
  char *szBry[1];

  szBry[0]=Brybuffer;

  /* check to see that the database is connected. */
  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
      TPCCErr("IsTuxInit - new order: %d ", tperrno);
      return ERR_DB_ERROR;
    }
  /* allocate memory and copy over data */
  if(NULL == ( retptr= (pNewOrderData) tpalloc("CARRAY", NULL,
nosiz)))
    {
      TPCCErr("tp alloc in neworder: %d ", tperrno);
      return ERR_DB_ERROR;
    }
  memcpy( retptr, ppNewOrder, nosiz);
  /* Call tuxedo for New Order */
  retcode = tpcall("no_transaction", (char *)retptr, nosiz,
                   (char**)&retptr, &nosiz, TPNOTIME |
TPSIGRSTRT );
  if( -1 == retcode )
    {
      /* Bry added to write event log for NewOrder transaction
completion */
       bryh=RegisterEventSource(NULL, TEXT("Bryon TEST"));
       sprintf(szBry[0],"newOrder transaction complete w: %d, d:
%d, c: %d, retcode: %d, tpurcode: %d", retptr->w_id, retptr->d_id,
retptr->c_id, tperrno, tpurcode);
       ReportEvent(bryh, EVENTLOG_INFORMATION_TYPE, 100, 0, NULL,
1, nosiz, szBry, retptr);
       DeregisterEventSource(bryh);
     /* End Bry added to write event log for NewOrder Transaction
completion */
      tpfree((char*) retptr);
      return ERR_DB_ERROR;
    }
  memcpy(ppNewOrder, retptr, nosiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}

/* stg - end Tuxedo change of TPCCNewOrder */



/* stg - begin Tuxedo change of TPCCOrderStatus */
/*
 * FUNCTION int TPCCOrderStatus
 */
int
TPCCOrderStatus( pOrderStatusData ppOrderStatus )
{
  int retcode = ERR_DB_SUCCESS;

  pOrderStatusData retptr;
  int ossiz = sizeof(OrderStatusData);

  /* check to see that the database is connected. */
  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
      TPCCErr("IsTuxInit - order status");
      return ERR_DB_ERROR;
    }

  /* allocate memory and copy over data */
  if(NULL == ( retptr= (pOrderStatusData) tpalloc("CARRAY", NULL,
ossiz)))
    {
      TPCCErr("tp alloc in order status: %d", tperrno);
      return ERR_DB_ERROR;
    }
  memcpy( retptr, ppOrderStatus, ossiz);

  /* Call tuxedo for Order Status */
  retcode = tpcall("os_transaction", (char *)retptr, ossiz,
                   (char**)&retptr, &ossiz, TPNOTIME |
TPSIGRSTRT );
  if( -1 == retcode )
    {
      TPCCErr("tpcall - order status");
      tpfree((char*) retptr);
```

```
      return ERR_DB_ERROR;
    }
  memcpy(ppOrderStatus, retptr, ossiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}

/* stg - end Tuxedo change of TPCCOrderStatus */


/* stg - begin Tuxedo change of TPCCPayment */
/*
 * FUNCTION int TPCCPayment
 */
int
TPCCPayment( pPaymentData ppPayment )
{
  int retcode = ERR_DB_SUCCESS;

  pPaymentData retptr;
  long ptsiz = sizeof(PaymentData);

  /* check to see that the database is connected. */
  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
      TPCCErr("IsTuxInit - payment ");
      return ERR_DB_ERROR;
    }

  /* allocate memory and copy over data */
  if(NULL == ( retptr= (pPaymentData) tpalloc("CARRAY", NULL,
ptsiz)))
    {
      TPCCErr("tp alloc in payment");
      return ERR_DB_ERROR;
    }
  memcpy( retptr, ppPayment, ptsiz);

  /* Call tuxedo for Payment */
  retcode = tpcall("pt_transaction", (char *)retptr, ptsiz,
                   (char**)&retptr, &ptsiz, TPNOTIME |
TPSIGRSTRT );
  if( -1 == retcode )
    {
      TPCCErr("tpcall - payment: %d ", tperrno);
      tpfree((char*) retptr);
      return ERR_DB_ERROR;
    }
  memcpy(ppPayment, retptr, ptsiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}

/* stg - end Tuxedo change of TPCCPayment */


/* stg - begin Tuxedo change of TPCCStockLevel */
/*
 * FUNCTION int TPCCStockLevel
 */
int
TPCCStockLevel( pStockLevelData ppStockLevel )
{
  int retcode = ERR_DB_SUCCESS;

  pStockLevelData retptr;
  int slsiz = sizeof(StockLevelData);

  /* check to see that the database is connected. */
  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
      TPCCErr("IsTuxInit - stock level ");
      return ERR_DB_ERROR;
    }

  /* allocate memory and copy over data */
  if(NULL == ( retptr= (pStockLevelData) tpalloc("CARRAY", NULL,
slsiz)))
    {
      TPCCErr("tp alloc in stock level");
      return ERR_DB_ERROR;
    }
  memcpy( retptr, ppStockLevel, slsiz);

  /* Call tuxedo for Stock Level */
  retcode = tpcall("sl_transaction", (char *)retptr, slsiz,
                   (char**)&retptr, &slsiz, TPNOTIME |
TPSIGRSTRT );
  if( -1 == retcode )
    {
      TPCCErr("tpcall - stock level: %d", tperrno);
      tpfree((char*) retptr);
      return ERR_DB_ERROR;
    }
  memcpy(ppStockLevel, retptr, slsiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}

/* stg - end Tuxedo change of TPCCStockLevel */


/* stg - getGovernorValue gets registry setting with number of
threads you want to run. */
/* stg - getGovernorValue is not needed for use with Tuxedo */
/* stg - Tuxedo's will read the ubb binary format file to get this
info */
/*
**++
**  FUNCTION NAME: getGovernorValue
**--
*/
BOOL
getGovernorValue( char *group, long *ctr )
{
  HKEY    hKey;
  DWORD   size;
  DWORD   type;
  char    keyname[256];
  BOOL    status;

  sprintf( keyname,
             "SOFTWARE\\DigitalEquipmentCorporation\\TPware\\Group
Settings\\%s",
             group );

  if( ERROR_SUCCESS != RegOpenKeyEx( HKEY_LOCAL_MACHINE, keyname,
                                                0, KEY_READ, &hKey ))
{
    TPCCErr( "Error, group settings not found for %s\r\n", keyname
);
    return FALSE;
  }

  size = sizeof( *ctr );
  if ( ERROR_SUCCESS ==
       RegQueryValueEx( hKey, "MaxThreads", 0, &type, (LPBYTE) ctr,
&size )) {
    status = TRUE;
  }
  else {
    TPCCErr( "Error, MaxThreads value not found for key %s\r\n",
keyname );
    status = FALSE;
  }

  RegCloseKey( hKey );

  return status;
}

/*
**++
**  FUNCTION NAME: TPCCGetTransportData
**--
*/
int
TPCCGetTransportData( pTransportData pTransport )
{
  HKEY    hKey;
  DWORD   size;
  DWORD   type;
  char    szTmp[FILENAMESIZE];
  int   status;
  long  total;
  long dyGovValue;
  long noGovValue;
  long osGovValue;
  long ptGovValue;
  long slGovValue;
  long gcGovValue;

  memset( pTransport, 0, sizeof( *pTransport ));
  pTransport->asynchronous = FALSE; /* stg - this is FALSE for
Tuxedo */

  status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                                0, KEY_READ, &hKey);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_TPCC_KEY;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "GenericTransactionServers",
                                0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS )
    gbGeneric = atoi(szTmp);


  RegCloseKey(hKey);

  /**** Read the ACMSxp Governor values from the registry. ****/
  if( !gbGeneric ) {
    pTransport->generic = FALSE;
    total = 0;
    if ( ! getGovernorValue( "tpcc_dy", &dyGovValue ))
      return ERR_GOVERNOR_VALUE_NOT_FOUND;
    total += dyGovValue;
    pTransport->num_dy = dyGovValue;
    if ( ! getGovernorValue( "tpcc_no", &noGovValue ))
      return ERR_GOVERNOR_VALUE_NOT_FOUND;
    total += noGovValue;
    pTransport->num_no = noGovValue;
    if ( ! getGovernorValue( "tpcc_os", &osGovValue ))
      return ERR_GOVERNOR_VALUE_NOT_FOUND;
    total += osGovValue;
    pTransport->num_os = osGovValue;
    if ( ! getGovernorValue( "tpcc_pt", &ptGovValue ))
      return ERR_GOVERNOR_VALUE_NOT_FOUND;
    total += ptGovValue;
    pTransport->num_pt = ptGovValue;
    if ( ! getGovernorValue( "tpcc_sl", &slGovValue ))
      return ERR_GOVERNOR_VALUE_NOT_FOUND;
    total += slGovValue;
    pTransport->num_sl = slGovValue;
  }
```

```
  else {
    pTransport->generic = TRUE;
    if ( ! getGovernorValue( "tpcc_gc", &gcGovValue ))
      return ERR_GOVERNOR_VALUE_NOT_FOUND;
    total = gcGovValue;
    pTransport->num_gc = gcGovValue;
  }

  status = DELIGetTransportData( &pTransport->dy_use_transport,
                                 &pTransport-
>num_dy_servers,
                                 &pTransport-
>num_queued_deliveries,
                                 &pTransport-
>num_queued_responses );

  if( ERR_SUCCESS != status )
    return status;

  return ERR_SUCCESS;
}

/*
**++
**  FUNCTION NAME: TPCCStartup
**--
*/
int
TPCCStartup( int iNumUsers, pTransportData pTransport )
{
  HKEY    hKey;
  DWORD   size;
  DWORD   type;
  char    szTmp[FILENAMESIZE];
  int   status;
/*  unsigned long ulhThread;*/
/* stg - not needed for tuxedo  force_connect_args fca; */
/*  int counts[5];*/
/*  int count;*/
/* long total;*/
/*  int maxDBConnections;*/

/* stg - added for Tuxedo */
InitializeCriticalSection (&TLS_crit_sec);

#ifdef USE_MUX
  stdl_call_dll_init( DLL_PROCESS_ATTACH );
#endif

  status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                        0, KEY_READ, &hKey);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_TPCC_KEY;

  /**** Obtain the name of the server machine into which ACMSxp
    processing servers will log.  ****/
  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Server", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_SERVER_VALUE;
  strcpy(gszServer, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Database", 0, &type, szTmp,
&size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_DATABASE_VALUE;
  strcpy(gszDatabase, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "User", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_USER_VALUE;
  strcpy(gszUser, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Password", 0, &type, szTmp,
&size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_PASSWORD_VALUE;
  strcpy(gszPassword, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "LoginDelay", 0, &type, szTmp,
&size);
  if ( status == ERROR_SUCCESS )
    giLoginDelay = atoi(szTmp);

  RegCloseKey(hKey);

  /**** start up server specific stuff ****/
/* stg - get maxDBConnections from TPCC registry key, not
calculated.  the following is not needed for tuxedo web ora app
code
  total = 0;
  if( !gbGeneric ) {
    counts[TYPE_DY] = pTransport->num_dy;
    total += pTransport->num_dy;
    counts[TYPE_NO] = pTransport->num_no;
    total += pTransport->num_no;
    counts[TYPE_OS] = pTransport->num_os;
    total += pTransport->num_os;
    counts[TYPE_PT] = pTransport->num_pt;
    total += pTransport->num_pt;
    counts[TYPE_SL] = pTransport->num_sl;
    total += pTransport->num_sl;
  }
  else
    total = pTransport->num_gc;

  maxDBConnections = total + TOTAL_ADMIN_CONNECTIONS;
  if( !pTransport->dy_use_transport )
    maxDBConnections += pTransport->num_dy_servers; */

/* stg - the following lines of code are for ACMSxp.  They are not
needed for
         Tuxedo.
  status = ACMSxpStartup( maxDBConnections, total );
  if ( status != ERR_DB_SUCCESS )
    return status; */

/* stg - the following line will start up the servers listed in the
Tuxedo
         config (ubb* file) file.  It gets the Tuxedo app up and
running and
         starts the Tuxedo server */
  status = system("tmboot -y");
  if (status !=0)
    {
      TPCCErr("Error booting the Tuxedo servers.");
      return status;
    }



  /* start the delivery subsystem */
  status = DELIServerStartup( pTransport->dy_use_transport,
                                  pTransport->num_dy_servers,
                                  pTransport-
>num_queued_deliveries,
                                  pTransport-
>num_queued_responses );
  if( ERR_SUCCESS != status )
    return status;

  status = DELIClientStartup( );
  if( ERR_SUCCESS != status )
    return status;

/* stg - don't need any of the gForce section for Tuxedo */
#if 0
  gInitRetStatus = ERR_DB_SUCCESS;      /** preset to success */
  gForceAllThreadsCtr = total;
  gForceAllThreadsEvent = CreateEvent( 0, TRUE, FALSE, 0 );
  if ( NULL == gForceAllThreadsEvent ) return
ERR_CANT_CREATE_ALL_THREADS_EVENT;

  if( !gbGeneric ) {
    for( fca.type = TYPE_NO; fca.type <= TYPE_SL; fca.type++ ) {
      for( count = 0; count < counts[fca.type]; count++ ) {
          ulhThread = _beginthread( force_connect, 0, (void *)
fca.type );
          if( -1 == ulhThread ) {
            return ERR_CANT_START_FRCDINIT_THREAD;
          }
          Sleep( giLoginDelay );
      }
    }
  }
  else {
    for( count = 0; count < pTransport->num_gc; count++ ) {
      ulhThread = _beginthread( force_connect, 0, (void *) TYPE_GC
);
      if( -1 == ulhThread ) {
          return ERR_CANT_START_FRCDINIT_THREAD;
      }
      Sleep( giLoginDelay );
    }
  }

  WaitForSingleObject( gForceAllThreadsEvent, INFINITE );

  return gInitRetStatus;
#endif   /* stg - ends #if 0 section for gForce code */

/* stg - added for Tuxedo */
/* allocate the index for the thread local storage. Used for tpinit
*/
tlsIndex = TlsAlloc();
if(0>tlsIndex)
  {
    TPCCErr("TPCCStartup error - thread local storage index alloc
failed");
    return ERR_CANT_ALLOCATE_THREAD_LOCAL_STORAGE;
  }
return ERR_SUCCESS;
}


/*
**++
**  FUNCTION NAME: TPCCConnect
**--
*/
int
TPCCConnect( pLoginData pLogin )
{

  if( 0 != strcmp( pLogin->szServer, gszServer ))
    return ERR_SERVER_MISMATCH;
```

```
  if( 0 != strcmp( pLogin->szDatabase, gszDatabase ))
    return ERR_DATABASE_MISMATCH;

  if( 0 != strcmp( pLogin->szUser, gszUser ))
    return ERR_USER_MISMATCH;

  if( 0 != strcmp( pLogin->szPassword, gszPassword ))
    return ERR_PASSWORD_MISMATCH;

  return ERR_DB_SUCCESS;
}

/*
**++
**  FUNCTION NAME: TPCCDisconnect
**--
*/
int
TPCCDisconnect( pCallersContext pCC )
{
  return ERR_DB_SUCCESS;
}


/* stg - added for TuxShutdown function for Tuxedo */
/*
 *  FUNCTION int TuxShutdown
 */
int
TuxShutdown()
{
  return ERR_DB_SUCCESS;
}

/*
**++
**  FUNCTION NAME: TPCCShutdown
**--
*/
int
TPCCShutdown( void )
{
  int                           retcode;

/* stg - changed for Tuxedo */
  if(0 == (TlsFree(tlsIndex))) TPCCErr("Error freeing TLS for
tpinit");

  /* shut down the servers listed in the TUXCONFIG file (ubb* file)
*/
  retcode = system("tmshutdown -y");
  if (retcode != 0)
    {
      TPCCErr("Error shutting the tuxedo servers down.");
      return retcode;
    }

  retcode = DELIClientShutdown( );
  if( ERR_SUCCESS != retcode )
    return retcode;

  retcode = DELIServerShutdown( );
  if( ERR_SUCCESS != retcode )
    return retcode;

  return(TuxShutdown());
}

/* stg - don't need the following for Tuxedo - I think! */
#if 0
void __cdecl
force_connect( void *arglist )
{
  LoginData                     login;
  int                           txnType;

  login.w_id = 0;
  login.ld_id = 0;
  login.pCC = 0;
  login.szApplication[0] = '\0';
  strcpy( login.szServer, gszServer );
  strcpy( login.szDatabase, gszDatabase );
  strcpy( login.szUser, gszUser );
  strcpy( login.szPassword, gszPassword );

  txnType = (int) arglist;
  switch ( txnType ) {
  case TYPE_DY:
    dy_transaction_init( STDL_SYNCHRONOUS, &login,
                         (struct io_login_wksp *)&login );
    break;

  case TYPE_NO:
    no_transaction_init( STDL_SYNCHRONOUS, &login,
                         (struct io_login_wksp *)&login );
    break;

  case TYPE_OS:
    os_transaction_init( STDL_SYNCHRONOUS, &login,
                         (struct io_login_wksp *)&login );
    break;

  case TYPE_PT:
    pt_transaction_init( STDL_SYNCHRONOUS, &login,
                         (struct io_login_wksp *)&login );
    break;

  case TYPE_SL:
    sl_transaction_init( STDL_SYNCHRONOUS, &login,
                         (struct io_login_wksp *)&login );
    break;

  case TYPE_GC:
    gc_transaction_init( STDL_SYNCHRONOUS, &login,
                         (struct io_login_wksp *)&login );
    break;
  }
  if ( login.status != ERR_DB_SUCCESS) {
    /** Only store the first failure **/
    if ( ERR_DB_SUCCESS == gInitRetStatus )
      gInitRetStatus = ERR_FORCE_CONNECT_THREAD_FAILED;

    TPCCErr( "Connect Transaction returned %8X\r\n", login.status
);
  }
  if ( InterlockedDecrement( &gForceAllThreadsCtr ) == 0 )
    SetEvent( gForceAllThreadsEvent );
  return;
}
#endif /*stg - end #if 0 section */


--------------------------
tux_srv.c
--------------------------
/*+*****************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997, 2000 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

*****************************************************************
*******-*/
#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <windows.h>
#include <process.h>
#include <Winsock.h>

#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpccerr.h>

#include <tpcc.h>

#include <atmi.h>

#ifdef FFE_DEBUG
# include <crtdbg.h>
#endif

/* dbproc pointer for db connection */
DBContext DBC;




#if 0

/* prototypes */
```

```
int tpsvrinit( int argc, char *argv[] );

extern int _tmrunserver _((int));
extern void dy_transaction _((TPSVCINFO *));
extern void no_transaction _((TPSVCINFO *));
extern void os_transaction _((TPSVCINFO *));
extern void pt_transaction _((TPSVCINFO *));
extern void sl_transaction _((TPSVCINFO *));

#ifndef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif
static struct tmdsptchtbl_t _tmdsptchtbl[] = {
  {"dy_transaction", "dy_transaction",
   (void (*) _((TPSVCINFO *))) dy_transaction, 0,0 },
  {"no_transaction", "no_transaction",
   (void (*) _((TPSVCINFO *))) no_transaction, 1,0 },
  {"os_transaction", "os_transaction",
   (void (*) _((TPSVCINFO *))) os_transaction, 2,0 },
  {"pt_transaction", "pt_transaction",
   (void (*) _((TPSVCINFO *))) pt_transaction, 3,0 },
  {"sl_transaction", "sl_transaction",
   (void (*) _((TPSVCINFO *))) sl_transaction, 4,0 },
  {NULL, NULL, NULL, 0, 0}
};
_TMDLLIMPORT extern struct xa_switch_t tmnull_switch;

struct tmsvrargs_t tmsvrargs = {
  NULL,
  &_tmdsptchtbl[0],
  0,
  tpsvrinit,
  tpsvrdone,
  _tmrunserver,
  NULL,
  NULL,
  NULL,
  NULL
};

struct tmsvrargs_t *
_tmgetsvrargs(void)
{
  tmsvrargs.xa_switch = &tmnull_switch;
  return(&tmsvrargs);
}

int
main(int argc, char **argv)
{
  return( _tmstartserver(argc,argv, _tmgetsvrargs()));
}


#endif /* stg ends #if 0 */
```

```
#if defined(__cplusplus)
extern "C" {
#endif
extern int _tmrunserver _((int));
extern void dy_transaction _((TPSVCINFO *));
extern void no_transaction _((TPSVCINFO *));
extern void os_transaction _((TPSVCINFO *));
extern void pt_transaction _((TPSVCINFO *));
extern void sl_transaction _((TPSVCINFO *));
#if defined(__cplusplus)
}
#endif

static struct tmdsptchtbl_t _tmdsptchtbl[] = {
        { "dy_transaction", "dy_transaction", (void (*)
_((TPSVCINFO *))) dy_transaction, 0, 0 },
        { "no_transaction", "no_transaction", (void (*)
_((TPSVCINFO *))) no_transaction, 1, 0 },
        { "os_transaction", "os_transaction", (void (*)
_((TPSVCINFO *))) os_transaction, 2, 0 },
        { "pt_transaction", "pt_transaction", (void (*)
_((TPSVCINFO *))) pt_transaction, 3, 0 },
        { "sl_transaction", "sl_transaction", (void (*)
_((TPSVCINFO *))) sl_transaction, 4, 0 },
        { NULL, NULL, NULL, 0, 0 }
};

#ifndef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

_TMDLLIMPORT extern struct xa_switch_t tmnull_switch;

struct tmsvrargs_t tmsvrargs = {
         NULL,
         &_tmdsptchtbl[0],
         0,
         tpsvrinit,
         tpsvrdone,
         _tmrunserver,       /* PRIVATE  */
         NULL,                          /* RESERVED */
         NULL,                          /* RESERVED */
         NULL,                          /* RESERVED */
         NULL                           /* RESERVED */
};
```

```
struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
_tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
         tmsvrargs.xa_switch = &tmnull_switch;
         return(&tmsvrargs);
}

int
#ifdef _TMPROTOTYPES
main(int argc, char **argv)
#else
main(argc,argv)
int argc;
char **argv;
#endif
{
#ifdef TMMAINEXIT
#include "mainexit.h"
#endif

         return( _tmstartserver( argc, argv, _tmgetsvrargs()));
}
```

```
/*
**++
**  FUNCTION NAME: tpsvrinit
**--
*/
int
tpsvrinit( int argc, char *argv[] )
{
  BOOL                          bLog;
  char                          szPath[MAX_PATH];
  int                           Len = sizeof( szPath ) - 1;
/* stg next two lines not needed for v6 web ora tux app code
  StartupData                   Startup;
  pStartupData                  pStartup = &Startup; */
  int status;
  HKEY hKey;
  DWORD size;
  DWORD type;
  char  szTmp[FILENAMESIZE];
  int maxDBConnections;
  LoginData  login;

  /* to avoid compiler errors */
  argc = argc;
  argv = argv;

  /* used for debugging the server code */
  /*Sleep(30000);*/

  userlog("Starting tpcc server");

  /* Get configuration data */
  status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                        0, KEY_READ, &hKey);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_TPCC_KEY;

  /* Get login data from registry settings */
  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Server", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_SERVER_VALUE;
  strcpy(login.szServer, szTmp);


  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Database", 0, &type, szTmp,
&size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_DATABASE_VALUE;
  strcpy(login.szDatabase, szTmp);


  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "User", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_USER_VALUE;
  strcpy(login.szUser, szTmp);


  size = sizeof(szTmp);
```

```
  status = RegQueryValueEx(hKey, "Password", 0, &type, szTmp,
&size);
  if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_PASSWORD_VALUE;
  strcpy(login.szPassword, szTmp);


  /* Get Path registry value */
  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "PATH", 0, &type, szTmp, &size);
  if (status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_PATH_VALUE;
  strcpy (szPath, szTmp);

  /* stg - Get maxDBConnections registry value.  We added
maxDBConnections to
     the SOFTWARE\\Microsoft\\TPCC registry key, so that we don't
have to
     calculate it here based on upstream calculations */

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "maxDBConnections", 0, &type,
szTmp, &size);
  if (status != ERROR_SUCCESS)
    /* required */
    return ERR_CANT_FIND_MAXDBCONNECTIONS_VALUE;
  maxDBConnections = atoi(szTmp);

  RegCloseKey(hKey);

  if (ERROR_SUCCESS == status)
  {
    /* set application name */
/*     strcpy( pStartup->Login.databaseLogin.szApplication,
"TUX_SRV" ); */

    TPCCOpenLog( szPath );
/* tpcc_ps calls tpccstartupdb with maxDBConnections ...we should
keep it as maxDBconnections...stg */
    TPCCStartupDB( maxDBConnections );

/* populate LoginData login structure like in tpcc_fct.c */
/* Server, Database, User and Password already populated into login
above */
   login.w_id = 0;
   login.ld_id = 0;
   login.pCC = 0;
   login.szApplication[0] = '\0';


    status = TPCCConnectDB( &DBC, &login );

    if(ERR_DB_SUCCESS != status)
    {
      TPCCErr( "tpsvrinit : Error logging into db." );
      return ERR_DB_ERROR;
    }
/* stg     TPCCLog( "%s, dbprocptr = %8X\r\n",
              pStartup->Login.databaseLogin.szApplication, DBC );
*/
    TPCCLog( "Finished TPCCConnectDB, dbprocptr = %8X\r\n", DBC );
  }
  else
  {
    TPCCErr("tpsvrinit : could not get configuration settings");
  }


  return (0);
}


/*
**++
**  FUNCTION NAME: tpsvrdone
**--
*/
void tpsvrdone(void)
{
  TPCCShutdownDB();
  return;
}

/*
**++
**  FUNCTION NAME: dy_transaction
**--
*/
void
dy_transaction( TPSVCINFO *dy_wksp )
{
  pDeliveryData ptr;

  ptr = (pDeliveryData)dy_wksp->data;

  ptr->status = TPCCDeliveryDB( DBC, ptr );
  if(ERR_DB_ERROR != ptr->status)
    tpreturn(TPSUCCESS, ptr->status, dy_wksp->data, dy_wksp->len,
0);
  else
    tpreturn(TPFAIL, ptr->status, dy_wksp->data, 0L, 0);
}

/*
**++
**  FUNCTION NAME: no_transaction
**--
*/
void
no_transaction( TPSVCINFO *no_wksp )
{
  pNewOrderData ptr;
  ptr = (pNewOrderData)no_wksp->data;

  ptr->status = TPCCNewOrderDB( DBC, ptr );

  if(ERR_DB_ERROR != ptr->status)
    tpreturn(TPSUCCESS, ptr->status, no_wksp->data, no_wksp->len,
0);
  else
    tpreturn(TPFAIL, ptr->status, no_wksp->data, 0L, 0);
}

/*
**++
**  FUNCTION NAME: os_transaction
**--
*/
void
os_transaction( TPSVCINFO *os_wksp )
{
  pOrderStatusData ptr;

  ptr = (pOrderStatusData)os_wksp->data;

  ptr->status = TPCCOrderStatusDB( DBC, ptr );
  if(ERR_DB_ERROR != ptr->status)
    tpreturn(TPSUCCESS, ptr->status, os_wksp->data, os_wksp->len,
0);
  else
    tpreturn(TPFAIL, ptr->status, os_wksp->data, 0L, 0);
}

/*
**++
**  FUNCTION NAME: pt_transaction
**--
*/
void
pt_transaction( TPSVCINFO *pt_wksp )
{
  pPaymentData ptr;

  ptr = (pPaymentData)pt_wksp->data;

  ptr->status = TPCCPaymentDB( DBC, ptr );
  if(ERR_DB_ERROR != ptr->status)
    tpreturn(TPSUCCESS, ptr->status, pt_wksp->data,
sizeof(PaymentData), 0);
  else
    tpreturn(TPFAIL, ptr->status, pt_wksp->data, 0L, 0);

}

/*
**++
**  FUNCTION NAME: sl_transaction
**--
*/
void
sl_transaction( TPSVCINFO *sl_wksp )
{
  pStockLevelData ptr;

  ptr = (pStockLevelData)sl_wksp->data;

  ptr->status = TPCCStockLevelDB( DBC, ptr );
  if(ERR_DB_ERROR != ptr->status)
    tpreturn(TPSUCCESS, ptr->status, sl_wksp->data, sl_wksp->len,
0);
  else
    tpreturn(TPFAIL, ptr->status, sl_wksp->data, 0L, 0);
}


--------------------------
web_ui.c
--------------------------
/*+**********************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
```

```
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

/*+
 * Abstract: This file contains the Digital created front end
functions
 *                  for the tpcc benchmark.
 *
 * Author: A Bradley & W Carr
 * Creation Date: May 1997
 *
 * Modified history:
 *
 *
*/
#include <windows.h>
#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
#include <io.h>
#include <crtdbg.h>
#include <process.h>

#define WEB_UI_C
#include <tpccerr.h>
#include <tpccstruct.h>
#include <tpccapi.h>
#include <httpext.h>

#include <tpcc.h>
#include <web_ui.h>
#include <ckpt.h>

#ifdef FFE_DEBUG
# include <crtdbg.h>
static int tmpDbgFlag;
static _HFILE hMemFile;
#endif


#define MAX(a,b) ((a)>(b)?(a):(b))

#define PUT_STRING(szString, iLen, pStart, pStruct) \
pStruct.szStr=szString; pStruct.iIndex=pStart;
pStruct.iFieldSize=iLen;

#define CONVERT_SPECIAL(pout,pin,iwid)\
{\
  char *out = pout;\
  char *in = pin;\
  int wid = iwid;\
  while( wid && '\0' != *in )\
  {\
    if( '>' == *in )\
    {*out++='&'; *out++='g'; *out++='t'; *out++=';';}\
    else if( '<' == *in )\
    {*out++='&'; *out++='l'; *out++='t'; *out++=';';}\
    else if( '&' == *in )\
    {*out++='&'; *out++='a'; *out++='m'; *out++='p'; *out++=';';}\
    else if( '\"' == *in )\
    {*out++='&'; *out++='q'; *out++='u'; *out++='o'; *out++='t';
*out++=';';}\
    else\
    {*out++=*in;}\
    in++;\
    wid--;\
  }\
  while( wid-- ) *out++ = ' ';\
}

/* define indexes for the building of the forms */
/* defines for new order */
#define NO_WDID 0
#define NO_WID NO_WDID + 1
#define NO_DID NO_WID + 1
#define NO_DATE NO_DID + 1
#define NO_CID NO_DATE + 1
#define NO_LAST NO_CID + 1
#define NO_CREDIT NO_LAST + 1
#define NO_DISC NO_CREDIT + 1
#define NO_OID NO_DISC + 1
#define NO_LINES NO_OID + 1
#define NO_W_TAX NO_LINES + 1
#define NO_D_TAX NO_W_TAX + 1
#define NO_S_WID NO_D_TAX + 1
#define NO_IID NO_S_WID + 1
#define NO_INAME NO_IID + 1
#define NO_QTY NO_INAME + 1
#define NO_STOCK NO_QTY + 1
#define NO_BG NO_STOCK + 1
#define NO_PRICE NO_BG + 1
#define NO_AMT NO_PRICE + 1
#define NO_STAT NO_AMT + (14*8) + 1
#define NO_TOTAL NO_STAT + 1

/* defines for payment input form */
#define PT_WDID_INPUT 0
#define PT_WID_INPUT PT_WDID_INPUT + 1

/* defines for payment output form */
#define PT_WDID 0
#define PT_LONG_DATE PT_WDID + 1
#define PT_WID PT_LONG_DATE + 1
#define PT_DID PT_WID + 1
#define PT_W_ST_1 PT_DID + 1
#define PT_D_ST_1 PT_W_ST_1 + 1
#define PT_W_ST_2 PT_D_ST_1 + 1
#define PT_D_ST_2 PT_W_ST_2 + 1
#define PT_W_CITY PT_D_ST_2 + 1
#define PT_W_ST PT_W_CITY + 1
#define PT_W_ZIP PT_W_ST + 1
#define PT_D_CITY PT_W_ZIP + 1
#define PT_D_ST PT_D_CITY + 1
#define PT_D_ZIP PT_D_ST + 1
#define PT_CID PT_D_ZIP + 1
#define PT_C_WID PT_CID + 1
#define PT_C_DID PT_C_WID + 1
#define PT_FIRST PT_C_DID + 1
#define PT_MIDDLE PT_FIRST + 1
#define PT_LAST PT_MIDDLE + 1
#define PT_SM_DATE PT_LAST + 1
#define PT_C_STR_1 PT_SM_DATE + 1
#define PT_CREDIT PT_C_STR_1 + 1
#define PT_D_STR_2 PT_CREDIT + 1
#define PT_DISC PT_D_STR_2 + 1
#define PT_C_CITY PT_DISC + 1
#define PT_C_ST PT_C_CITY + 1
#define PT_C_ZIP PT_C_ST + 1
#define PT_C_PHONE PT_C_ZIP + 1
#define PT_AMT PT_C_PHONE + 1
#define PT_BAL PT_AMT + 1
#define PT_LIM PT_BAL + 1
#define PT_CUST_DATA PT_LIM + 1

/* defines for order status */
#define OS_WDID 0
#define OS_WID OS_WDID + 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_BAL OS_LAST + 1
#define OS_OID OS_BAL + 1
#define OS_DATE OS_OID + 1
#define OS_CAR_ID OS_DATE + 1
#define OS_S_WID OS_CAR_ID + 1
#define OS_IID OS_S_WID + 1
#define OS_QTY OS_IID + 1
#define OS_AMT OS_QTY + 1
#define OS_SM_DATE OS_AMT + 1
/* defines for delivery form */
#define D_WDID 0
#define D_WID D_WDID + 1
#define D_CAR D_WID + 1
#define D_QUEUE1 D_CAR + 1
#define D_DELTA1 D_QUEUE1 + 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

/* defines for stock level form */
#define SL_WDID 0
#define SL_WID SL_WDID + 1
#define SL_DID SL_WID + 1
#define SL_TH SL_DID + 1
#define SL_LOW SL_TH + 1

#define WDID(w_id,d_id) (w_id*10+(d_id-1))

#define PANIC_FORM_SIZE 4096
```

```
#define NUMBER_POOL_FORM_TYPES 5
#define DELIVERY_FORM 0
#define NEW_ORDER_FORM 1
#define ORDER_STATUS_FORM 2
#define PAYMENT_FORM 3
#define STOCK_LEVEL_FORM 4

#define NUMBER_POOL_RESPONSE_TYPES 5
#define DELIVERY_RESPONSE 0
#define NEW_ORDER_RESPONSE 1
#define ORDER_STATUS_RESPONSE 2
#define PAYMENT_RESPONSE 3
#define STOCK_LEVEL_RESPONSE 4

#ifdef FFE_DEBUG
# define FFE_ASSERT(arg) _ASSERT(arg)
#else
# define FFE_ASSERT(arg)
#endif

#define RESERVE_FORM(type,szForm)\
  {\
    EnterCriticalSection( &gpForms->critSec[type] );\
    FFE_ASSERT( gpForms->iNextFreeForm[type] <= gpForms-
>iMaxIndex[type] );\
    szForm = gpForms->index[gpForms->iFirstFormIndex[type] +\
                                gpForms->iNextFreeForm[type]++];\
    LeaveCriticalSection( &gpForms->critSec[type] );\
  }
#define UNRESERVE_FORM(type,szForm)\
  {\
    EnterCriticalSection( &gpForms->critSec[type] );\
    FFE_ASSERT( gpForms->iNextFreeForm[type] > 0 );\
    gpForms->index[gpForms->iFirstFormIndex[type] +\
                    --gpForms->iNextFreeForm[type]] = szForm;\
    LeaveCriticalSection( &gpForms->critSec[type] );\
  }

#define RESERVE_RESPONSE(type,szResponse)\
  {\
    EnterCriticalSection( &gpResponses->critSec[type] );\
    FFE_ASSERT(gpResponses->iNextFreeResponse[type]<=gpResponses-
>iMaxIndex[type]);\
    szResponse = gpResponses->index[gpResponses-
>iFirstResponseIndex[type] +\
                                    gpResponses-
>iNextFreeResponse[type]++];\
    LeaveCriticalSection( &gpResponses->critSec[type] );\
  }
#define UNRESERVE_RESPONSE(type,szResponse)\
  {\
    EnterCriticalSection( &gpResponses->critSec[type] );\
    FFE_ASSERT(gpResponses->iNextFreeResponse[type] > 0 );\
    gpResponses->index[gpResponses->iFirstResponseIndex[type] +\
                    --gpResponses->iNextFreeResponse[type]] =
szResponse;\
    LeaveCriticalSection( &gpResponses->critSec[type] );\
  }

#define RESERVE_PANIC_FORM(szForm)\
  {\
    EnterCriticalSection( &gpPanicForms->critSec );\
    FFE_ASSERT( gpPanicForms->iNextFree <= gpPanicForms->iMaxIndex
);\
    szForm = gpPanicForms->index[gpPanicForms->iNextFree++];\
    LeaveCriticalSection( &gpPanicForms->critSec );\
  }

#define UNRESERVE_PANIC_FORM(szForm)\
  {\
    EnterCriticalSection( &gpPanicForms->critSec );\
    FFE_ASSERT( gpPanicForms->iNextFree > 0 );\
    gpPanicForms->index[--gpPanicForms->iNextFree] = szForm;\
    LeaveCriticalSection( &gpPanicForms->critSec );\
  }

#if 0
CMD                0
FORM ID            3
LOGIN WAREHOUSE    4
LOGIN DISTRICT     5
DELI QUEUE TIME    6
CARRIER ID         7
DISTRICT  8
CUSTOMER  9
NEWORDER FIELDS A-X,a-u
CUST LAST NAME     Y
CUST WAREHOUSE     Z
CUST DISTRICT      v
AMOUNT PAID        w
THRESHOLD x
#endif

#define MENU_BAR \
"<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=OrderStatus>"\
"<INPUT TYPE=submit NAME=0 VALUE=StockLevel>"\
"<INPUT TYPE=submit NAME=0 VALUE=Exit>"

static char         szFormTemplate[] =
"<BODY><FORM ACTION=%s METHOD=GET>";

static char         szWelcomeFormTemplate[] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=3 VALUE=W00>"
"Please Identify your Warehouse and District for this session.<BR>"
"Warehouse ID <INPUT NAME=4 SIZE=5><BR>"
"District ID <INPUT NAME=5 SIZE=2><BR>"
"<HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Submit>"
"</FORM></BODY>";

static char
szWelcomeForm[sizeof(szWelcomeFormTemplate)+FILENAMESIZE];
static int  iWelcomeFormLen;

static char         szMainMenuFormTemplate[] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=3 VALUE=M%06d>"
"%55.55s<BR>"
"Select Desired Transaction.<BR>"
MENU_BAR
"</FORM></BODY>";

static char szDeliveryFormTemp2i[] =
"<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=1><BR><BR>"
"Execution Status:<BR></PRE>"
"<HR><INPUT TYPE=submit NAME=0 VALUE=Process>"
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szDeliveryFormTemp2p[] =
"<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>"
MENU_BAR
"</FORM></BODY>";

static char szNewOrderFormTemp2i[] =
"<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=4>  <INPUT NAME=B SIZE=6>"
"                            <INPUT NAME=C SIZE=1><BR>"
" <INPUT NAME=D SIZE=4>  <INPUT NAME=E SIZE=6>"
"                            <INPUT NAME=F SIZE=1><BR>"
" <INPUT NAME=G SIZE=4>  <INPUT NAME=H SIZE=6>"
"                            <INPUT NAME=I SIZE=1><BR>"
" <INPUT NAME=J SIZE=4>  <INPUT NAME=K SIZE=6>"
"                            <INPUT NAME=L SIZE=1><BR>"
" <INPUT NAME=M SIZE=4>  <INPUT NAME=N SIZE=6>"
"                            <INPUT NAME=O SIZE=1><BR>"
" <INPUT NAME=P SIZE=4>  <INPUT NAME=Q SIZE=6>"
"                            <INPUT NAME=R SIZE=1><BR>"
" <INPUT NAME=S SIZE=4>  <INPUT NAME=T SIZE=6>"
"                            <INPUT NAME=U SIZE=1><BR>"
" <INPUT NAME=V SIZE=4>  <INPUT NAME=W SIZE=6>"
"                            <INPUT NAME=X SIZE=1><BR>"
" <INPUT NAME=a SIZE=4>  <INPUT NAME=b SIZE=6>"
"                            <INPUT NAME=c SIZE=1><BR>"
" <INPUT NAME=d SIZE=4>  <INPUT NAME=e SIZE=6>"
"                            <INPUT NAME=f SIZE=1><BR>"
" <INPUT NAME=g SIZE=4>  <INPUT NAME=h SIZE=6>"
"                            <INPUT NAME=i SIZE=1><BR>"
" <INPUT NAME=j SIZE=4>  <INPUT NAME=k SIZE=6>"
"                            <INPUT NAME=l SIZE=1><BR>"
" <INPUT NAME=m SIZE=4>  <INPUT NAME=n SIZE=6>"
"                            <INPUT NAME=o SIZE=1><BR>"
" <INPUT NAME=p SIZE=4>  <INPUT NAME=q SIZE=6>"
"                            <INPUT NAME=r SIZE=1><BR>"
" <INPUT NAME=s SIZE=4>  <INPUT NAME=t SIZE=6>"
"                            <INPUT NAME=u SIZE=1><BR>"
"Execution Status:
Total:<BR><HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Process>"
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szNewOrderFormTemp2p[] =
"<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>"
"  ####   ######   ########################  ##    ###    #   "
```

```
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"  ####   ######   ########################  ##    ###    #   "
"$######  $#######  <BR>"
"Execution Status: ########################                     "
"Total:  $########  <BR>"
"</PRE>"
MENU_BAR
"</FORM></BODY>";

static char szOrderStatusFormTemp2i[] =
"<INPUT TYPE=hidden NAME=3 VALUE=O######>"
"<PRE>                              Order-Status<BR>"
"Warehouse: #####  District: <INPUT NAME=8 SIZE=1><BR>"
"Customer: <INPUT NAME=9 SIZE=4>   Name:                     "
"<INPUT NAME=Y SIZE=23><BR>"
"Cust-Balance:<BR><BR>"
"Order-Number:            Entry-Date:
Carrier-Number:<BR>"
"Supply-W     Item-Id    Qty     Amount      Delivery-
Date<BR></PRE><HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Process>"
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szOrderStatusFormTemp2p[] =
"<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-Date<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"  ####       ######     ##     $#########      ##########<BR>"
"</PRE>"
MENU_BAR
"</FORM></BODY>";

static char szPaymentFormTemp2i[] =
"<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=5>  "
"Cust-District: <INPUT NAME=v SIZE=1><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>"
"<INPUT TYPE=submit NAME=0 VALUE=Process>"
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szPaymentFormTemp2p[] =
"<INPUT TYPE=hidden NAME=3 VALUE=p######>"
"<PRE>                                 Payment<BR>"
"Date: ################### <BR><BR>"
"Warehouse: #####                       District: ##<BR>"
"####################                   ####################<BR>"
"####################                         ####################<BR>"
"#################### ## ##########       #################### ##
##########"
"<BR><BR>"
"Customer: ####  Cust-Warehouse: ##### Cust-District: ##<BR>"
"Name:   ################ ## ################     Since:
##########<BR>"
"        ####################                     Credit: ##<BR>"
"        ####################                     %Disc:
#####<BR>"
"        #################### ## ##########       Phone:
###################"
"<BR><BR>"
"Amount Paid:          $#######      New Cust Balance:
$##############<BR>"
"Credit Limit:   $#############<BR><BR>"
"Cust-Data: ##################################################<BR>"
"           ##################################################<BR>"
"           ##################################################<BR>"
"           ##################################################<BR>"
"</PRE>"
MENU_BAR
"</FORM></BODY>";

static char szStockLevelFormTemp2i[] =
"<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><HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Process>"
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szStockLevelFormTemp2p[] =
"<INPUT TYPE=hidden NAME=3 VALUE=s######>"
"<PRE>                                   Stock-Level<BR>"
"Warehouse: #####  District: ##<BR><BR>"
"Stock Level Threshold: ##<BR><BR>"
"low stock: ###"
"</PRE>"
MENU_BAR
"</FORM></BODY>";

static char szErrorFormTemplate[] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=3 VALUE=e%06d>"
"Error: %d (%s): %s"
MENU_BAR
"</FORM></BODY>";

static char szResponseHeaderTemplate[] =
"Connection: keep-alive\r\n"
"Content-type: text/html\r\n"
"Content-length: ####\r\n"
"\r\n";
static char  szResponseHeader[sizeof(szResponseHeaderTemplate)];
FORM_INDEXES responseHeaderIndexes[1] = { 0 };
int            responseHeaderLen = 0;


#define MATCHES_BEGIN(p) ('B'==p[0])
#define MATCHES_CHECKPOINT(p) \
(0==strncmp(p,"Checkpoint",strlen("Checkpoint")))
#define MATCHES_CHECKPOINT_STARTUP(p) \
(0==strncmp(p,"CheckpointStartup",strlen("CheckpointStartup")))
#define MATCHES_CLEAR(p) ('C'==p[0]&&'l'==p[1])
#define MATCHES_DELIVERY(p) ('D'==p[0])
#define MATCHES_EXIT(p) ('E'==p[0])
#define MATCHES_MENU(p) ('M'==p[0])
#define MATCHES_NEWORDER(p) ('N'==p[0])
#define MATCHES_ORDERSTATUS(p) ('O'==p[0])
#define MATCHES_PAYMENT(p) ('P'==p[0]&&'a'==p[1])
#define MATCHES_PROCESS(p) ('P'==p[0]&&'r'==p[1])
#define MATCHES_STOCKLEVEL(p) ('S'==p[0]&&'t'==p[1])
#define MATCHES_SUBMIT(p) ('S'==p[0]&&'u'==p[1])
#ifdef FFE_DEBUG
# define MATCHES_MEMORYCHECK(p) ('!'==p[0]&&'M'==p[1])
#endif

/* function prototypes */
void BeginCmd( EXTENSION_CONTROL_BLOCK *pECB );
void CheckpointCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id );
void CheckpointStartupCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id,
int ld_id );
void ClearCmd( EXTENSION_CONTROL_BLOCK *pECB );
void ExitCmd( EXTENSION_CONTROL_BLOCK *pECB );
void MenuCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id );
void SubmitCmd( EXTENSION_CONTROL_BLOCK *pECB, int *w_id, int
*ld_id );
void MemoryCheckCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id );

BOOL GetKeyValuePtr( char *szIPtr, char *szKey, char **pszOPtr );
BOOL GetCharKeyValuePtr( char *szIPtr, char cKey, char **pszOPtr );
BOOL GetKeyValueString( char *szIPtr, char *szKey,
                        char *szValue, int iSize );
BOOL GetWDID(char *ptr, int *lw_id, int *ld_id, char **optr);

void Log( char *szType, char *szStr );
void MakePanicPool( DWORD dwResponseSize );
void MakeTemplatePool( DWORD dwFormSize, DWORD dwResponseSize );
void MakeTransactionPool( DWORD dwTransactionPoolSize );
void DeletePanicPool( void );
void DeleteTemplatePool( void );
void DeleteTransactionPool( void );
```

```
int ProcessDeliveryQuery( EXTENSION_CONTROL_BLOCK *pECB,
                                char *lpszQueryString,
                                int w_id, int ld_id );
int ProcessNewOrderQuery( EXTENSION_CONTROL_BLOCK *pECB,
                                char *lpszQueryString,
                                int w_id, int ld_id );
int ProcessOrderStatusQuery( EXTENSION_CONTROL_BLOCK *pECB,
                                  char *lpszQueryString,
                                  int w_id, int ld_id );
int ProcessPaymentQuery( EXTENSION_CONTROL_BLOCK *pECB,
                               char *lpszQueryString,
                               int w_id, int ld_id );
int ProcessStockLevelQuery( EXTENSION_CONTROL_BLOCK *pECB,
                                 char *lpszQueryString,
                                 int w_id, int ld_id );

DWORD ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB);

void PutNumeric( int iInt, int iFieldSize, char *pChar );
void SendErrorResponse( EXTENSION_CONTROL_BLOCK *pECB, int iError,
                            int iErrorType, char *szMsg, int
w_id, int ld_id,
                            pConnData pConn );
void SendMainMenuForm( EXTENSION_CONTROL_BLOCK *pECB,
                          int w_id, int ld_id, char *szStatus );
void SendResponse(EXTENSION_CONTROL_BLOCK *pECB, char *szStr, int
iStrLen);
void SendWelcomeForm(EXTENSION_CONTROL_BLOCK *pECB);
#ifdef FFE_DEBUG
unsigned __stdcall CheckMemory(void *param);
#endif

/* typedefs */
typedef struct
{
  char *szStr;
  int iIndex;
  int iFieldSize;
  int iNewIndex;
  int iNewFieldSize;
} PutStrStruct, *pPutStrStruct;

typedef struct
{
  CRITICAL_SECTION critSec;
#ifdef FFE_DEBUG
  int     iMaxIndex;
#endif
  int     iNextFree;
  char    *index[1];
  char    forms[PANIC_FORM_SIZE];
} PanicStruct, *pPanicStruct;

typedef struct
{
  CRITICAL_SECTION critSec[NUMBER_POOL_FORM_TYPES];
#ifdef FFE_DEBUG
  int     iMaxIndex[NUMBER_POOL_FORM_TYPES];
#endif
  int     iNextFreeForm[NUMBER_POOL_FORM_TYPES];
  int     iFirstFormIndex[NUMBER_POOL_FORM_TYPES];
  char    *index[1];
  char    forms[1];
} FormStruct, *pFormStruct;

typedef struct
{
  CRITICAL_SECTION critSec[NUMBER_POOL_RESPONSE_TYPES];
#ifdef FFE_DEBUG
  int     iMaxIndex[NUMBER_POOL_RESPONSE_TYPES];
#endif
  int     iNextFreeResponse[NUMBER_POOL_RESPONSE_TYPES];
  int     iFirstResponseIndex[NUMBER_POOL_RESPONSE_TYPES];
  char    *index[1];
  char    responses[1];
} ResponseStruct, *pResponseStruct;

/* global variables */
static int                    iInitStatus         = ERR_SUCCESS;

static CRITICAL_SECTION startupCriticalSection;
static BOOL             startupFlag     = FALSE;

static pPanicStruct gpPanicForms        = NULL;
static int giPanic                      = 0;
static pFormStruct gpForms              = 0;
static int giFormLen[NUMBER_POOL_FORM_TYPES] = { 0 };
static pResponseStruct gpResponses      = 0;
static int giResponseLen[NUMBER_POOL_RESPONSE_TYPES] = { 0 };

/* FUNCTION: BOOL APIENTRY DllMain(HANDLE hModule, DWORD
ul_reason_for_call,
 *                                      LPVOID lpReserved)
 *
 * PURPOSE:         This is the main entry point to an ISAPI dll.
All dll
 *                  global initializations should be done in this
routine.
 *
 * ARGUMENTS:       HANDLE    hModule                       dll
module handle
 *                  DWORD     ul_reason_for_call  reason for call
 *                  LPVOID    lpReserved
        reserved for future use
 *
 * RETURNS:         BOOL      Always TRUE                   Errors
in intiialization
 *                                                          are
presented at the first
 *                                                          screen
to the user.
 * COMMENTS:        None
 *
 */

BOOL APIENTRY
DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpRes)
{
  char              szTmpFileName[FILENAMESIZE];
  DWORD             dwFileNameLen;
  char              *pChr;

  switch( ul_reason_for_call )
  {
  case DLL_PROCESS_ATTACH:
#ifdef FFE_DEBUG
    tmpDbgFlag = _CrtSetDbgFlag(_CRTDBG_REPORT_FLAG);
    tmpDbgFlag |= _CRTDBG_CHECK_CRT_DF;
    tmpDbgFlag |= _CRTDBG_LEAK_CHECK_DF;
    tmpDbgFlag |= _CRTDBG_ALLOC_MEM_DF;
    tmpDbgFlag |= _CRTDBG_CHECK_ALWAYS_DF;
    _CrtSetDbgFlag(tmpDbgFlag);

    hMemFile = CreateFile( "MemErrors", GENERIC_WRITE,
FILE_SHARE_READ, NULL,
                                OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL );

   _CrtSetReportMode( _CRT_WARN, _CRTDBG_MODE_FILE );
   _CrtSetReportFile( _CRT_WARN, hMemFile );
   _CrtSetReportMode( _CRT_ERROR, _CRTDBG_MODE_FILE );
   _CrtSetReportFile( _CRT_ERROR, hMemFile );
   _CrtSetReportMode( _CRT_ASSERT, _CRTDBG_MODE_FILE );
   _CrtSetReportFile( _CRT_ASSERT, hMemFile );
#endif
#ifdef DEBUG_ENTRY
    DebugBreak( );
#endif
    InitializeCriticalSection( &startupCriticalSection );

    dwFileNameLen = GetModuleFileName( hModule, szTmpFileName,
FILENAMESIZE-1);
    if( 0 == dwFileNameLen )
      return FALSE;

    pChr = strrchr( szTmpFileName, '\\' );
    if( NULL == pChr )
      return FALSE;

    pChr++;
    dwFileNameLen = strlen( pChr );
    if( 0 >= dwFileNameLen )
      return FALSE;

    CopyMemory( szModName, pChr, dwFileNameLen+1 );

    iWelcomeFormLen = sprintf(szWelcomeForm, szWelcomeFormTemplate,
szModName);

    if( ERR_SUCCESS != iInitStatus )
      return TRUE;

    break;
  case DLL_THREAD_ATTACH:
    break;
  case DLL_THREAD_DETACH:
    dwFileNameLen = 0;
    break;
  case DLL_PROCESS_DETACH:

    TPCCShutdown( );

    DeleteTransactionPool( );
    DeleteTemplatePool( );
    DeletePanicPool( );

    DeleteCriticalSection( &startupCriticalSection );

    TPCCCloseLog( );

    break;
  }
  return TRUE;
}

/* FUNCTION: BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO
*pVersion)
 *
 * PURPOSE:         This function is called by the inet service
when the DLL
 *                  is first loaded.
 *
 * ARGUMENTS:       HSE_VERSION_INFO  *pVersion   passed in
structure in
 *                                                          which
to place expected
 *                                                          version
number.
 *
 * RETURNS:         TRUE
        expected return value.
 *
 * COMMENTS:        None
 *
 */
```

```
BOOL WINAPI
GetExtensionVersion(HSE_VERSION_INFO *pVersion)
{

  pVersion->dwExtensionVersion =
    MAKELONG(HSE_VERSION_MINOR, HSE_VERSION_MAJOR);
  strncpy(pVersion->lpszExtensionDesc,
          "Digital TPC-C Server.",
          HSE_MAX_EXT_DLL_NAME_LEN-1);
  *(pVersion->lpszExtensionDesc) = '\0';

  return TRUE;
}

/* FUNCTION: DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK
*pECB)
 *
 * PURPOSE:         This function is the main entry point for the
TPCC DLL.
 *                  The internet service calls this function
passing in the
 *                  http string.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB
        structure ptr containing the
 *
        internet service information.
 *
 * RETURNS:         DWORD     HSE_STATUS_SUCCESS  connection can be
dropped if
 *                                                          error
 *                            HSE_STATUS_SUCCESS_AND_KEEP_CONN
keep connect valid
 *                                                          comment
sent
 *
 * COMMENTS:        None
 *
 */

DWORD WINAPI
HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
  DWORD                       status;
  int                         dbstatus;
  TransportData               transport;
  int                         formPoolSize;
  int                         responsePoolSize;
  int                         panicPoolSize;
  int                         transactionPoolSize;
  int                         transportThreadLimit;
  int                         webServerThreadLimit;
#ifdef FFE_DEBUG
  unsigned long               ulHandle;
  unsigned                    uThreadAddr;
#endif

  if ( ! startupFlag ) {
    EnterCriticalSection( &startupCriticalSection );
    if ( ! startupFlag ) {
      if( ERR_SUCCESS == iInitStatus ) {
          if ( ERR_SUCCESS != ( iInitStatus = ReadRegistrySettings(
)))
            MakePanicPool( 50 );        /* make room for error
messages */
          else {
            TPCCOpenLog( );
            iInitStatus = TPCCGetTransportData( &transport );
            if( ERR_SUCCESS != iInitStatus ) {
              MakePanicPool( 50 );      /* make room for error
messages */
            }
            else {
              webServerThreadLimit = gdwPoolThreadLimit;
              if( transport.generic ) {
                transportThreadLimit = transport.num_gc;
              }
              else {
                transportThreadLimit = transport.num_dy +
                    transport.num_no + transport.num_os +
                    transport.num_pt + transport.num_sl;
              }

              /* The size of the transaction pool represents the
maximum number */
              /* of transactions that can be carried by any part of
the system. */
              /* First sum the transactions that can be carried by
the delivery */
              /* subsystem. */
              transactionPoolSize = transport.num_dy_servers +
                transport.num_queued_deliveries +
transport.num_queued_responses;

              /* Menu transactions are only carried by the web
server */
              /* and hence are limited by the count of web server
threads. */
              formPoolSize = webServerThreadLimit;

              if( transport.asynchronous ) {
                /* Delivery transactions and their interactive
response */
                /* are carried by the web server threads since they
are */
                /* synchronous.  Database transactions and their
responses */
                /* are limited by transport threads.  Therefore the
maximum */
                /* number of concurrent responses is the greater of
the */
                /* web server and transport thread counts. */
                responsePoolSize =
MAX(webServerThreadLimit,transportThreadLimit);

                /* Panic forms are used either to report errors or
to */
                /* handle the case where a response required the
quoting */
                /* of a character that is an HTML special
character. */
                /* The maximum number of these forms is also the
greater of the */
                /* web server and transport thread counts. */
                panicPoolSize = MAX( webServerThreadLimit,
transportThreadLimit );

                /* Since the transport is asynchronous, all web
server and */
                /* transport threads can carry a transaction. */
                transactionPoolSize +=
                    webServerThreadLimit + transportThreadLimit;
              }
              else {
                /* If the transport runs synchronously, only web
server */
                /* threads can generate any form of response. */
                responsePoolSize = webServerThreadLimit;
                panicPoolSize = webServerThreadLimit;

                /* Since the transport is synchronous, a single
transaction is */
                /* shared by the web thread and the transport
thread. */
                transactionPoolSize +=
                    MAX( webServerThreadLimit, transportThreadLimit
);
              }

              MakeTemplatePool( formPoolSize, responsePoolSize );
              MakePanicPool( panicPoolSize );
              MakeTransactionPool( transactionPoolSize );
              dbstatus = TPCCStartup( iMaxConnections, &transport
);
              if( ERR_DB_SUCCESS != dbstatus ) {
                iInitStatus = dbstatus;
              }

#ifdef FFE_DEBUG
                    ulHandle = _beginthreadex(
                                NULL,                           /*
Security            */
                                0,
          /* Stack size        */
                                CheckMemory,        /* Start
Address*/
                                NULL,                           /*
Arglist             */
                                0,
          /* Init flag         */
                                &uThreadAddr);  /* Thread address */

                    _ASSERT (ulHandle != 0);
#endif
            }
          }
      }
      startupFlag = TRUE;
    }
    LeaveCriticalSection( &startupCriticalSection );
  }

  if( ERR_SUCCESS != iInitStatus )
  {
    SendErrorResponse(pECB, iInitStatus, ERR_TYPE_WEBDLL, NULL, -1,
-1, NULL);
    return HSE_STATUS_SUCCESS;
  }

  /* if registry setting is for html logging then show http string
passed in.*/
  if ( bLog )
  {
    TPCCLog( "%s %s\r\n", "* QUERY    *", pECB->lpszQueryString );
  }

  /* process http query */
  status = ProcessQueryString(pECB);

  /* finish up with status returned by Processing functions */
  return status;
}

/* FUNCTION: void SendErrorResponse( EXTENSION_CONTROL_BLOCK *pECB,
int iError,
 *                                          int iErrorType, char
*szMsg,
 *                                          int w_id, int ld_id )
 *
 * PURPOSE:         This function displays an error form in the
client browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
```

```
 *                                                              unique
to this connection.
 *                int       iError                        id of
error message
 *                int       iErrorType                    error
type, ERR_TYPE_SQL,
 *
         ERR_TYPE_DBLIB, ERR_TYPE_WEBDLL
 *                int       w_id                          Login
warehouse ID.
 *                int       ld_id                         Login
district ID.
 *                char      *szMsg
         optional error message string
 *                                                              used
with ERR_TYPE_SQL and
 *
         ERR_TYPE_DBLIB
 *
 * RETURNS:       None
 *
 * COMMENTS:      If the error type is ERR_TYPE_WEBDLL the szMsg
parameter
 *                may be NULL because it is ignored. If the error
type is
 *                ERR_TYPE_SQL or ERR_TYPE_DBLIB then the szMsg
parameter
 *                contains the text of the error message, so the
szMsg
 *                parameter cannot be NULL.
 *
 */

void
SendErrorResponse( EXTENSION_CONTROL_BLOCK *pECB, int iError, int
iErrorType,
                   char *szMsg, int w_id, int ld_id, pConnData
pConn )
{
  int     ii;

  static char szNoMsg[] = "";
  char              *szErrorTypeMsg;
  char              *szErrorMsg;
  char              *szForm;
  int               iStrLen;

  if ( !szMsg  )
    szMsg = szNoMsg;

  RESERVE_PANIC_FORM( szForm );

  if( ERR_TYPE_WEBDLL == iErrorType )
  {
    ii = 0;
    while( '\0' != errorMsgs[ii].szMsg[0] && iError !=
errorMsgs[ii].iError )
      ii++;
    if ( '\0' == errorMsgs[ii].szMsg[0] )
      ii = 1;  /* ERR_NO_MESSAGE */
    szErrorTypeMsg = "TPCCWEB";
    szErrorMsg = errorMsgs[ii].szMsg;
  }
  else if( ERR_TYPE_SQL == iErrorType )
  {
    szErrorTypeMsg = "SQLSVR";
    szErrorMsg = szMsg;
  }
  else if( ERR_TYPE_DBLIB == iErrorType )
  {
    szErrorTypeMsg = "DBLIB";
    szErrorMsg = szMsg;
  }

  if( NULL != pConn )
    TPCCTransactionErr( pConn, "%s(%d): %s\r\n",
                        szErrorTypeMsg, iError, szErrorMsg );
  else
    TPCCErr( "%s(%d): %s\r\n", szErrorTypeMsg, iError, szErrorMsg
);

  iStrLen = sprintf( szForm, szErrorFormTemplate, szModName,
                     WDID(w_id,ld_id), iError, szErrorTypeMsg,
szErrorMsg );

  SendResponse(pECB, szForm, iStrLen);

  UNRESERVE_PANIC_FORM( szForm );
}

/* FUNCTION: void HandlePanic(pPutStrStruct pStruct,
 *                                    char *szInput, int
iInputSize,
 *                                    char **szOutput, int
*iOutputSize )
 *
 * PURPOSE:   This routine handles the case where the output string
contains
 * at least one of the special characters double quote ("),
ampersand (&),
 * less than (<), or greater than (>).  What it does is scan the
strings
 * to be output checking for all special characters.  It then moves
the
 * input string template sections further along in the output
string
 * making enough room for the strings including their special
quoted
 * charaters, then fills the new template with the output strings.
 *
 * ARGUMENTS:
 *
 * RETURNS:         void
 *
 * COMMENTS:
 */

void
HandlePanic( pPutStrStruct pStruct,
             char *szInput, int iInputSize,
             char **szOutput, int *iOutputSize )
{
  pPutStrStruct pStructTmp1;
  pPutStrStruct pStructTmp2;
  char *pIChar;
  int iExtra;
  int iTotalExtra;
  char *szTmp;

  RESERVE_PANIC_FORM( szTmp );

  /* first, save what we've done so far */
  *szOutput = szTmp;
  memcpy( szTmp, szInput, pStruct->iIndex );

  /* save the original values for string moving */
  pStructTmp1 = pStruct;
  while( NULL != pStructTmp1->szStr ) {
    pStructTmp1->iNewIndex = pStructTmp1->iIndex;
    pStructTmp1->iNewFieldSize = pStructTmp1->iFieldSize;
    pStructTmp1++;
  }

  /* parse all remaining strings for special characters and fix
indicies */
  pStructTmp1 = pStruct;
  iTotalExtra = 0;
  while( NULL != pStructTmp1->szStr ) {
    pIChar = pStructTmp1->szStr;
    iExtra = 0;
    while( 0 != *pIChar )
    {
      if( '"' == *pIChar )
          iExtra += 5;
      else if( '&' == *pIChar )
          iExtra += 4;
      else if( '<' == *pIChar )
          iExtra += 3;
      else if( '>' == *pIChar )
          iExtra += 3;
      pIChar++;
    }

    /* reset field width for this string */
    pStructTmp1->iNewFieldSize += iExtra;

    /* move all following indicies */
    for( pStructTmp2 = pStructTmp1+1;
           NULL != pStructTmp2->szStr;
           pStructTmp2++ )
      pStructTmp2->iNewIndex += iExtra;

    pStructTmp1++;
    iTotalExtra += iExtra;
  }

  /* update new string length */
  *iOutputSize = iInputSize + iTotalExtra;

  /* move end of string to new output string */
  --pStructTmp1;
  memcpy( &szTmp[pStructTmp1->iNewIndex + pStructTmp1-
>iNewFieldSize],
          &szInput[pStructTmp1->iIndex + pStructTmp1-
>iFieldSize],
          iInputSize - pStructTmp1->iIndex + pStructTmp1-
>iFieldSize);

  /* move input string pieces to new locations in output string */
  pStructTmp2 = pStructTmp1--;
  while( pStruct != pStructTmp2 )
  {
    memcpy( &szTmp[pStructTmp1->iNewIndex + pStructTmp1-
>iNewFieldSize],
            &szInput[pStructTmp1->iIndex + pStructTmp1-
>iFieldSize],
             pStructTmp2->iIndex -
             ( pStructTmp1->iIndex + pStructTmp1->iFieldSize ));
    pStructTmp2 = pStructTmp1--;
  }

  /* Now put in the strings */
  pStructTmp1 = pStruct;
  while( NULL != pStructTmp1->szStr ) {
    CONVERT_SPECIAL( &szTmp[pStructTmp1->iNewIndex], pStructTmp1-
>szStr,
                     pStructTmp1->iNewFieldSize );
    pStructTmp1++;
  }
}

/* FUNCTION: void SendResponse(EXTENSION_CONTROL_BLOCK *pECB, char
*szForm,
 *                                  int iStrLen)
 *
 * PURPOSE:
```

```
 *    This function takes the forms generated by each transaction
routine
 *    and calls the server callback function to pass it on to the
browser.
 *
 * ARGUMENTS:
 *    EXTENSION_CONTROL_BLOCK *pECB               Server context
structure.
 *    char                              *szForm             form to
pass to browser.
 *    int                     iStrLen             length of form
excluding null.
 *
 * RETURNS:
 *    None
 *
 * COMMENTS:
 */

void
SendResponse(EXTENSION_CONTROL_BLOCK *pECB, char *szForm, int
iStrLen)
{
  char              szHeader1[sizeof(szResponseHeader)];
  static char       szHeader[] = "200 Ok";
  static int        iSize = sizeof(szHeader)-1;
  int               lpbSize;

  lpbSize = iStrLen + 1;

  if ( bLog )
    TPCCLog( "%s %s\r\n", "* RESPONSE *", szForm );

  CopyMemory( szHeader1, szResponseHeader, sizeof(szResponseHeader)
);
  PutNumeric(lpbSize, responseHeaderIndexes[0].iLen,
              &szHeader1[responseHeaderIndexes[0].iStartIndex]);

  (*pECB->ServerSupportFunction)(pECB->ConnID,
HSE_REQ_SEND_RESPONSE_HEADER,
                                 szHeader, &iSize,
(LPDWORD)szHeader1);
  (*pECB->WriteClient)(pECB->ConnID, szForm, &lpbSize, 0);
}

/* FUNCTION: ParseTemplateString(char *szForm, int *pcurLen,
 *                                char *formTemplate,
FORM_INDEXES *indexes)
 *
 * PURPOSE:         This function parses the query string to find
the ## signs
 *                  that mark the positions for the values to be
put, and
 *                  stores these locations and lengths in the
indexes structure.
 *
 * ARGUMENTS: char  *szForm    the resultant form
 *              int *pcurLen the current length of szForm
 *              char          *formTemplate       the form's
template
 *              FORM_INDEXES *indexes   ptr to the array of indexes
for the
 *                                                  tag values of the
form
 *
 * RETURNS:         void
 *
 * COMMENTS:
 */

void
ParseTemplateString(char *szForm, int *pcurLen,
                    char *formTemplate, FORM_INDEXES *indexes)
{
  int   curIndex = 0;
  int     ii = 0;
  int     jj;
  int     curLen;

  curLen = *pcurLen;
  while ('\0' != formTemplate[ii])
  {
    if('#' != formTemplate[ii])
    {
      szForm[curLen] = formTemplate[ii];
      ii++;
      curLen++;
    }
    else
    {
      jj = 0;
      indexes[curIndex].iStartIndex = curLen;
      while('#' == formTemplate[ii])
      {
          jj++;
          szForm[curLen] = formTemplate[ii];
          curLen++;
          ii++;
      }
      indexes[curIndex].iLen = jj;
      curIndex++;
    }
  }
  szForm[curLen] = '\0';
  *pcurLen = curLen;
}

/* FUNCTION: void PutNumeric(int iInt, int iFieldSize, char *pChar
)
 *
 * PURPOSE:         This function converts an integer to a char
string.
 *
 * ARGUMENTS:       int       iInt                the integer to
convert to string
 *                  int       iFieldSize          max size of char
string to return.
 *                  char      *pChar              the string to put
the int into.
 *
 * RETURNS:         None
 *
 * COMMENTS:        If the Integer value exceeds the max field
size, then
 *                  the string will be filled with iFieldSize "*"
to signal
 *                  an error.
 */

void
PutNumeric( int iInt, int iFieldSize, char *pChar )
{
  int iSaveSize = iFieldSize;
  char *pSaveStart = pChar;
  char pAsterisk[] = "**********";
  BOOL bSignFlag = TRUE;

  pChar += (iFieldSize - 1);
  if(0 > iInt)
  {
    bSignFlag = FALSE;
    iInt = abs(iInt);
  }

  do
  {
    *pChar = ( iInt % 10 ) + '0';
    iInt /= 10;
    iFieldSize--;
    if( iFieldSize )
      pChar--;
  } while( iFieldSize );

  if( !bSignFlag )
  {
    if('0' == *pChar)
      *pChar = '-';
    else
    {
      memcpy( pSaveStart, pAsterisk, iSaveSize );
      return;
    }
  }

  if( 0 != iInt )
  {
    /* put in string of ** to signal error */
    memcpy( pSaveStart, pAsterisk, iSaveSize );
  }
}

/* FUNCTION: void SendDeliveryForm( EXTENSION_CONTROL_BLOCK *pECB,
 *                                        int w_id, int ld_id )
 *
 * PURPOSE:         This function puts the data into the input form
and then
 *                  returns the form to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB
 *          structure pointer to passed in
 *
 *          internet service information.
 *                  int       w_id                          Login
warehouse ID.
 *                  int       ld_id                         Login
district ID.
 &
 * RETURNS:                   None
 *
 * COMMENTS:        None
 *
 */

void
SendDeliveryForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id )
{
  char    *deliveryForm;

  RESERVE_FORM( DELIVERY_FORM, deliveryForm );

  PutNumeric(WDID(w_id,ld_id),
               deliveryFormIndexesI[D_WDID].iLen,

&deliveryForm[deliveryFormIndexesI[D_WDID].iStartIndex]);
  PutNumeric(w_id,
               deliveryFormIndexesI[D_WID].iLen,

&deliveryForm[deliveryFormIndexesI[D_WID].iStartIndex]);

  SendResponse(pECB, deliveryForm, giFormLen[DELIVERY_FORM]);

  UNRESERVE_FORM( DELIVERY_FORM, deliveryForm );
}
```

```
/* FUNCTION: void SendNewOrderForm( EXTENSION_CONTROL_BLOCK *pECB,
 *                                      int w_id, int ld_id )
 *
 * PURPOSE:         This function puts the data into the input form
and then
 *                  returns the form to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     pointer
to the structure that
 *                                                            is
passed in the internet
 *                  int                 w_id      warehouse id
 *                  int                 ld_id     login district id
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

void
SendNewOrderForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id )
{
  char    *newOrderForm;

  RESERVE_FORM( NEW_ORDER_FORM, newOrderForm );

  PutNumeric(WDID(w_id,ld_id),
             newOrderFormIndexes[NO_WDID].iLen,

&newOrderForm[newOrderFormIndexes[NO_WDID].iStartIndex]);
  PutNumeric(w_id,
             newOrderFormIndexes[NO_WID].iLen,

&newOrderForm[newOrderFormIndexes[NO_WID].iStartIndex]);

  SendResponse(pECB, newOrderForm, giFormLen[NEW_ORDER_FORM]);

  UNRESERVE_FORM( NEW_ORDER_FORM, newOrderForm );
}

/* FUNCTION: void SendPaymentForm(EXTENSION_CONTROL_BLOCK *pECB,
 *                                    int w_id, int ld_id,
DBContext *pdb)
 *
 * PURPOSE:         This function puts the data into the input form
and then
 *                  returns the form to the browser.
 *
 * ARGUMENTS:
 *                  EXTENSION_CONTROL_BLOCK       *pECB     pointer
to structure passed in
 *                                                            the
internet
 *                  int                           w_id
         warehouse id
 *                  int                           ld_id     login
district id
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

void
SendPaymentForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id
)
{
  char    *paymentForm;

  RESERVE_FORM( PAYMENT_FORM, paymentForm );

  PutNumeric(WDID(w_id,ld_id),
             paymentFormIndexes[PT_WDID_INPUT].iLen,

&paymentForm[paymentFormIndexes[PT_WDID_INPUT].iStartIndex]);
  /* the date field is before wid for the response so use 2 here */
  PutNumeric(w_id,
             paymentFormIndexes[PT_WID_INPUT].iLen,

&paymentForm[paymentFormIndexes[PT_WID_INPUT].iStartIndex]);

  SendResponse(pECB, paymentForm, giFormLen[PAYMENT_FORM]);

  UNRESERVE_FORM( PAYMENT_FORM, paymentForm );
}

/* FUNCTION: void SendOrderStatusForm(EXTENSION_CONTROL_BLOCK
*pECB,
 *                                          int w_id, int
ld_id, DBContext *pdb)
 *
 * PURPOSE:         This function fills in data and then sends the
order status
 *                  input form back to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK *pECB ptr to structure
passed in the
 *
         internet.
 *                  int                 w_id      warehouse id
 *                  int                 ld_id     login district id
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

void
SendOrderStatusForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id )
{
  char    *orderStatusForm;

  RESERVE_FORM( ORDER_STATUS_FORM, orderStatusForm );

  PutNumeric(WDID(w_id,ld_id),
             orderStatusFormIndexes[OS_WDID].iLen,

&orderStatusForm[orderStatusFormIndexes[OS_WDID].iStartIndex]);
  PutNumeric(w_id,
             orderStatusFormIndexes[OS_WID].iLen,

&orderStatusForm[orderStatusFormIndexes[OS_WID].iStartIndex]);
  SendResponse(pECB, orderStatusForm,
giFormLen[ORDER_STATUS_FORM]);

  UNRESERVE_FORM( ORDER_STATUS_FORM, orderStatusForm );
}

/* FUNCTION: void SendStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB,
 *                                                      int w_id, int
d_id, DBContext *pdb)
 *
 * PURPOSE:         This function puts the data into the input form
and then
 *                  returns the form to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB
         structure pointer to passed
 *                                                            in
internet service information
 *                  int                 w_id
         warehouse id
 *                  int                 d_id
         district id
 *                  DBContext *pdb                pointer to
database context.
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

void
SendStockLevelForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
d_id )
{
  char    *stockLevelForm;

  RESERVE_FORM( STOCK_LEVEL_FORM, stockLevelForm );

  PutNumeric(WDID(w_id,d_id),
             stockLevelFormIndexes[SL_WDID].iLen,

&stockLevelForm[stockLevelFormIndexes[SL_WDID].iStartIndex]);
  PutNumeric(w_id,
             stockLevelFormIndexes[SL_WID].iLen,

&stockLevelForm[stockLevelFormIndexes[SL_WID].iStartIndex]);
  PutNumeric(d_id,
             stockLevelFormIndexes[SL_DID].iLen,

&stockLevelForm[stockLevelFormIndexes[SL_DID].iStartIndex]);

  SendResponse(pECB, stockLevelForm, giFormLen[STOCK_LEVEL_FORM]);

  UNRESERVE_FORM( STOCK_LEVEL_FORM, stockLevelForm );
}


/* FUNCTION: void SendMainMenuForm(EXTENSION_CONTROL_BLOCK *pECB,
 *                                        int w_id, int ld_id,
char *szStatus)
 *
 * PURPOSE:         This function sends the main menu form to the
browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                            unique
to this connection.
 *
 *                  int       w_id
         warehouse id
 *                  int       ld_id                           login
district id
 *                  char      *szStatus           String to report
previous
 *
         operation status.
 *
 * RETURNS:         None
 *
 * COMMENTS:
 */

void
SendMainMenuForm( EXTENSION_CONTROL_BLOCK *pECB,
                     int w_id, int ld_id, char *szStatus )
{
  char    *szForm;
```

```
  int     iStrLen;
  static char       *szNoStatus = "";
  char    *pszStatus;

  pszStatus = ( NULL == szStatus ) ? szNoStatus : szStatus;

  RESERVE_PANIC_FORM( szForm );

  iStrLen = sprintf( szForm, szMainMenuFormTemplate,
                     szModName, WDID(w_id,ld_id), pszStatus );

  SendResponse(pECB, szForm, iStrLen);

  UNRESERVE_PANIC_FORM( szForm );
}

/* FUNCTION: void SendWelcomeForm(EXTENSION_CONTROL_BLOCK *pECB)
 *
 * PURPOSE:         This function sends the welcome form to the
browser.
 *
 * ARGUMENTS:   None
 *
 * RETURNS:         None
 *
 * COMMENTS:        The welcome form is generated on
initialization.
 */

void
SendWelcomeForm(EXTENSION_CONTROL_BLOCK *pECB)
{
  SendResponse( pECB, szWelcomeForm, iWelcomeFormLen );
}

/* FUNCTION: DWORD ProcessQueryString(EXTENSION_CONTROL_BLOCK
*pECB)
 *
 * PURPOSE:         This function extracts the relevent information
out
 *                  of the http command passed in from the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                             unique
to this connection.
 *
 * RETURNS:         DWORD                                     server
connection status code
 *
 * COMMENTS:        If this is the initial connection i.e. client
is at
 *                  welcome screen then there will not be a
terminal id or
 *                  current form id if this is the case then the
pTermid and
 *                  pFormid return values are undefined.
 */

DWORD
ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB)
{
  static char *beginptr = "Begin";
  char *ptr;
  char *cmdptr;
  int cFormID;
  int w_id;
  int ld_id;
  DWORD status;
  int retcode;

  w_id = 0;
  ld_id = 0;

  if ( GetCharKeyValuePtr( pECB->lpszQueryString, '3', &ptr ))
  {
    cFormID = *ptr++;
    if( !GetWDID( ptr, &w_id, &ld_id, &ptr )) {
      SendErrorResponse( pECB, ERR_W_ID_INVALID, ERR_TYPE_WEBDLL,
NULL,
                         w_id, ld_id, NULL );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
  }
  else
    cFormID = '\0';

  /* now figure out what command we have and execute it */
  if ( !GetCharKeyValuePtr( ptr, '0', &cmdptr ))
  {
    if( 0 == strlen( pECB->lpszQueryString )) {
      cmdptr = beginptr;
    }
    else {
      SendErrorResponse( pECB, ERR_COMMAND_UNDEFINED,
ERR_TYPE_WEBDLL,
                         NULL, w_id, ld_id, NULL );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
  }

  if( '\0' == cFormID && !MATCHES_BEGIN( cmdptr )) {
    SendErrorResponse( pECB, ERR_INVALID_FORM_AND_CMD_NOT_BEGIN,
                       ERR_TYPE_WEBDLL, NULL, w_id, ld_id, NULL
);
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }

  status = HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  if( MATCHES_PROCESS( cmdptr ))
  {
    if( 'N' == cFormID )
      retcode = ProcessNewOrderQuery( pECB, ptr, w_id, ld_id );
    else if( 'P' == cFormID )
      retcode = ProcessPaymentQuery( pECB, ptr, w_id, ld_id );
    else if( 'D' == cFormID )
      retcode = ProcessDeliveryQuery( pECB, ptr, w_id, ld_id );
    else if( 'O' == cFormID )
      retcode = ProcessOrderStatusQuery( pECB, ptr, w_id, ld_id );
    else if( 'S' == cFormID )
      retcode = ProcessStockLevelQuery( pECB, ptr, w_id, ld_id );
    else {
      SendErrorResponse( pECB, ERR_INVALID_FORM, ERR_TYPE_WEBDLL,
NULL,
                         w_id, ld_id, NULL );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }

    if( ERR_DB_PENDING == retcode )
      status = HSE_STATUS_PENDING;
    else if( ERR_DB_SUCCESS != retcode ) {
      SendErrorResponse( pECB, retcode, ERR_TYPE_WEBDLL,
                         NULL, w_id, ld_id, NULL );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
  }
  else if( MATCHES_PAYMENT( cmdptr ))
    SendPaymentForm( pECB, w_id, ld_id );
  else if( MATCHES_NEWORDER( cmdptr ))
    SendNewOrderForm( pECB, w_id, ld_id );
  else if( MATCHES_DELIVERY( cmdptr ))
    SendDeliveryForm( pECB, w_id, ld_id );
  else if( MATCHES_ORDERSTATUS( cmdptr ))
    SendOrderStatusForm( pECB, w_id, ld_id );
  else if( MATCHES_STOCKLEVEL( cmdptr ))
    SendStockLevelForm( pECB, w_id, ld_id );
  else if( MATCHES_EXIT( cmdptr ))
    ExitCmd( pECB );
  else if( MATCHES_SUBMIT( cmdptr ))
    SubmitCmd( pECB, &w_id, &ld_id );
  else if( MATCHES_BEGIN( cmdptr ))
    BeginCmd( pECB );
  else if( MATCHES_MENU( cmdptr ))
    MenuCmd( pECB, w_id, ld_id );
  else if( MATCHES_CLEAR( cmdptr ))
    ClearCmd( pECB );
  else if( MATCHES_CHECKPOINT_STARTUP( cmdptr ))
    CheckpointStartupCmd( pECB, w_id, ld_id );
  else if( MATCHES_CHECKPOINT( cmdptr ))
    CheckpointCmd( pECB, w_id, ld_id );
#ifdef FFE_DEBUG
  else if( MATCHES_MEMORYCHECK( cmdptr ))
    MemoryCheckCmd( pECB, w_id, ld_id );
#endif
  else
    SendErrorResponse( pECB, ERR_COMMAND_UNDEFINED,
ERR_TYPE_WEBDLL,
                       NULL, w_id, ld_id, NULL );

  return status;
}

/* FUNCTION: PutFloat2(double dVal, int iFieldSize, char *pChar )
 *
 * PURPOSE:         This function converts a double into a char
string
 *                  in the format of xx.xx
 *
 * ARGUMENTS: double          dVal                the value to
convert to char
 *              int iFieldSize          max size of char string
 *              char          pChar                string where to
put value
 *
 * RETURNS:         void
 *
 * COMMENTS:        If the double exceeds the max field size
entered,
 *                  the char string will be filled with iFieldSize
*'s
 *                  to signal an error
 */

void
PutFloat2( double dVal, int iFieldSize, char *pChar )
{
  int iInt;
  int iDecimal;
  BOOL bSignFlag = TRUE;
  int iSaveSize = iFieldSize;
  char *pSaveStart = pChar;
  char pAsterisk[] = "********************";

  pChar += (iFieldSize - 1);

  if(0 > dVal)
  {
    bSignFlag = FALSE;
    iInt = abs((int)( dVal * 100. ));
  }
  else
  {
    iInt = (int)( dVal * 100. );
  }
  iDecimal = 2;
  do
```

```
  {
    *pChar-- = ( iInt % 10 ) + '0';
    iInt /= 10;
    iFieldSize--;
  } while( --iDecimal );

  *pChar-- = '.';
  iFieldSize--;

  do
  {
    *pChar-- = ( iInt % 10 ) + '0';
    iInt /= 10;
    iFieldSize--;
  } while( iFieldSize && iInt != 0 );

  if( !iFieldSize && iInt != 0 )
  {
    /* put in string of ** to signal error */
    memcpy(pSaveStart, pAsterisk, iSaveSize);
    return;
  }
  if(!bSignFlag)
  {
    iFieldSize--;
    if( 0 >= iFieldSize )
    {
      /* put in string of  ** to signal error */
      memcpy(pSaveStart, pAsterisk, iSaveSize);
      return;
    }
    *pChar-- = '-';
  }

  /* Fill in the remaining spaces in the field with blanks. */
  while( iFieldSize-- )
    *pChar-- = ' ';
}
/* FUNCTION: void PutHTMLStrings( pPutStrStruct pStruct,
 *                                                 char *szInput,
int iInputSize,
 *                                                 char **szOutput,
int *iOutputSize )
 *
 * PURPOSE:  This routine takes a template output string and a data
structure
 *           containing strings, positions, and field widths of
strings to be
 *           compied into the template.  The routine scans all
input strings to
 *           determine if any contain special charaters that need
to be quoted
 *           in the output string.  If none exist, the template
is filled with
 *           the desired strings.  If at least one special
character exists in
 *           the output strings, a more expensive routine is
called to build a
 *           new output string template containing the quoted
strings.
 *
 * ARGUMENTS: pPutStrStruct   pStruct   pointer to structure
containing the
 *                                                 strings,
positions and field lengths.
 *            char                    *szInput  pointer to input
form
 *            int           iInputSize          length of the
input form
 *            char                    **szOutput          pointer
to the new input form
 *                                                 it may
or may not be different
 *                                                 than
the input form.
 *            int           iOutputSize         length of the new
input form.
 *
 * RETURNS:   none
 *
 * COMMENTS:        none
 */

void
PutHTMLStrings( pPutStrStruct pStruct,
                char *szInput, int iInputSize,
                char **szOutput, int *iOutputSize )
{
  char *pIChar;
  char *pOChar;
  int     iFieldSize;

  while( NULL != pStruct->szStr )
  {
    pIChar = pStruct->szStr;
    pOChar = szInput + pStruct->iIndex;
    iFieldSize = pStruct->iFieldSize;
    while( 0 != *pIChar && iFieldSize )
    {
      /* '>' is the highest ACSII value of the special characters.
*/
      /* If '>' is greater than the character is question, check
further. */
      if( '>' > *pIChar )
      {
          if( '"' == *pIChar || '&' == *pIChar ||
              '<' == *pIChar || '>' == *pIChar )
          {
              /* We have found at least one special character in the
desired */
              /* output string, go the the more expensive routine to
build */
              /* the desired output string. */
              InterlockedIncrement( &giPanic );
              HandlePanic( pStruct, szInput, iInputSize, szOutput,
iOutputSize );
              return;
          }
          else
            *pOChar = *pIChar;
      }
      else
          *pOChar = *pIChar;

      pIChar++;
      pOChar++;
      iFieldSize--;
    }

    /* Fill in the remaining spaces in the field with blanks. */
    while( iFieldSize-- )
      *pOChar++ = ' ';

    pStruct++;
  }

  /* The output string is the template and the length is unchanged
*/
  *szOutput = szInput;
  *iOutputSize = iInputSize;

  return;
}

/* FUNCTION: void TPCCDeliveryResponse( EXTENSION_CONTROL_BLOCK
*pECB,
 *                                                 int retcode,
 *                                                 DeliveryData
*deliveryData )
 *
 * PURPOSE:         This function fills in the values and returns
the
 *                  response form to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK *pECB
 *                  int                             retcode  return
code from db
 *                  DeliveryData                    *deliveryData
pointer to the delivery
 *                                                          data
structure.
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */

void
TPCCDeliveryResponse( int retcode, pDeliveryData pDelivery,
                      pDeliveryData
CompletedDeliveries[DELIVERY_RESPONSE_COUNT] )
{
  int ssCnt = 0;
  char *szOutput;
  int iOutputLen;
  PutStrStruct StrStruct[2];
  char *deliveryForm;
  int ii, jj, index;
  pDeliveryData pCompletedDelivery;
  static DeliveryData blankDelivery = { 0 };
  EXTENSION_CONTROL_BLOCK *pECB;

  pECB = pDelivery->pCC;

  if ( ERR_DB_PENDING == retcode )
  {
    return;
  }
  else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
  {
    for( jj = 0; jj < DELIVERY_RESPONSE_COUNT; jj++ )
      if( NULL != CompletedDeliveries[jj] ) {
#ifdef FFE_DEBUG
          CompletedDeliveries[jj]->iStage |= UNRESERVING;
#endif
          UNRESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS,
CompletedDeliveries[jj] );
      }

    SendErrorResponse( pECB, ERR_DELIVERY_NOT_PROCESSED,
                            ERR_TYPE_WEBDLL, NULL,
                            pDelivery->w_id, pDelivery->ld_id,
                            (pConnData)pDelivery );

    return;
  }
  else    if ( ERR_DB_SUCCESS != retcode )
  {
    for( jj = 0; jj < DELIVERY_RESPONSE_COUNT; jj++ )
      if( NULL != CompletedDeliveries[jj] ) {
#ifdef FFE_DEBUG
          CompletedDeliveries[jj]->iStage |= UNRESERVING;
#endif
          UNRESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS,
CompletedDeliveries[jj] );
      }
```

```
    SendErrorResponse( pECB, ERR_DB_DELIVERY_NOT_QUEUED,
                           ERR_TYPE_WEBDLL, NULL,
                           pDelivery->w_id, pDelivery->ld_id,
                           (pConnData)pDelivery );

    return;
  }

  RESERVE_RESPONSE( DELIVERY_RESPONSE, deliveryForm );

  PutNumeric(WDID(pDelivery->w_id,pDelivery->ld_id),
               deliveryFormIndexesP[D_WDID].iLen,

&deliveryForm[deliveryFormIndexesP[D_WDID].iStartIndex]);
  PutNumeric(pDelivery->w_id,
               deliveryFormIndexesP[D_WID].iLen,

&deliveryForm[deliveryFormIndexesP[D_WID].iStartIndex]);
  PutNumeric(pDelivery->o_carrier_id,
               deliveryFormIndexesP[D_CAR].iLen,

&deliveryForm[deliveryFormIndexesP[D_CAR].iStartIndex]);

  index = D_QUEUE1;
  for( jj = 0; jj < DELIVERY_RESPONSE_COUNT; jj++ ) {
    if( NULL == CompletedDeliveries[jj] )
      pCompletedDelivery = &blankDelivery;
    else
      pCompletedDelivery = CompletedDeliveries[jj];
    PutNumeric(pCompletedDelivery->queue_time,
                 deliveryFormIndexesP[index].iLen,

&deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
    index++;
    PutNumeric(pCompletedDelivery->delta_time,
                 deliveryFormIndexesP[index].iLen,

&deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
    index++;
    PutNumeric(pCompletedDelivery->w_id,
                 deliveryFormIndexesP[index].iLen,

&deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
    index++;
    PutNumeric(pCompletedDelivery->o_carrier_id,
                 deliveryFormIndexesP[index].iLen,

&deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
    index++;
    for( ii = 0; ii < 10; ii++ ) {
      PutNumeric(pCompletedDelivery->o_id[ii],
                   deliveryFormIndexesP[index].iLen,

&deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
      index++;
    }
    if( NULL != CompletedDeliveries[jj] ) {
#ifdef FFE_DEBUG
      CompletedDeliveries[jj]->iStage |= UNRESERVING;
#endif
      UNRESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS,
CompletedDeliveries[jj] );
    }
  }

  PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);
  PutHTMLStrings(StrStruct, deliveryForm,
giResponseLen[DELIVERY_RESPONSE],
                   &szOutput, &iOutputLen);

  SendResponse(pECB, szOutput, iOutputLen);

  UNRESERVE_RESPONSE( DELIVERY_RESPONSE, deliveryForm );

  if( szOutput != deliveryForm )
    UNRESERVE_PANIC_FORM( szOutput );
}

/* FUNCTION: void TPCCNewOrderResponse(EXTENSION_CONTROL_BLOCK
*pECB,
 *                                                  int retcode,
 *                                                  NewOrderData
*newOrderData )
 *
 * PURPOSE:         This function fills in the values and returns
the
 *                  response form to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     pointer
to the structure
 *                                                           that
contains the internet
 *                                                           service
information.
 *                  int                 retcode   return status
from the db.
 *                  NewOrderData        *newOrderData       pointer
to structure containing
 *                                                           data
about the current txn.
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */

void
TPCCNewOrderResponse( int retcode, pNewOrderData pNewOrder )
{
  int     i;
  char  szDate[]   = "xx-xx-xxxx xx:xx:xx";
  char  szBlanks[] = "                      ";
  char  szDollar[] = "$";
  PutStrStruct StrStruct[133];
  int ssCnt = 0;
  int jj;
  int kk;
  int mm;
  char *newOrderForm;
  char *szOutput;
  int iOutputLen;
  BOOL bValid;
  char *execution_status;
  char szStatus[80];
  EXTENSION_CONTROL_BLOCK *pECB;

  pECB = pNewOrder->pCC;

  if ( ERR_DB_PENDING == retcode )
  {
    return;
  }
  else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
  {
    SendErrorResponse( pECB, ERR_NEW_ORDER_NOT_PROCESSED,
                           ERR_TYPE_WEBDLL, NULL,
                           pNewOrder->w_id, pNewOrder->ld_id,
                           (pConnData)pNewOrder );
    return;
  }
  else if( ERR_DB_SUCCESS != retcode && ERR_DB_NOT_COMMITED !=
retcode )
  {
    sprintf( szStatus,
               "Item number is not valid, or DB error = %d",
               pNewOrder->dbstatus );
    SendErrorResponse( pECB, ERR_DB_ERROR,
                           ERR_TYPE_WEBDLL, NULL,
                           pNewOrder->w_id, pNewOrder->ld_id,
                           (pConnData)pNewOrder );
    return;
  }
  else if ( ERR_DB_SUCCESS == retcode )
  {
    bValid = TRUE;
    execution_status = "Transaction commited.";
  }
  else if ( ERR_DB_NOT_COMMITED == retcode )
  {
    bValid = FALSE;
    execution_status = "Item number is not valid.";
  }

  RESERVE_RESPONSE( NEW_ORDER_RESPONSE, newOrderForm );

  if(bValid)
  {
    PutNumeric(WDID(pNewOrder->w_id,pNewOrder->ld_id),
                 newOrderResponseIndexes[NO_WDID].iLen,

&newOrderForm[newOrderResponseIndexes[NO_WDID].iStartIndex]);
    PutNumeric(pNewOrder->w_id,
                 newOrderResponseIndexes[NO_WID].iLen,

&newOrderForm[newOrderResponseIndexes[NO_WID].iStartIndex]);
    PutNumeric(pNewOrder->d_id,
                 newOrderResponseIndexes[NO_DID].iLen,

&newOrderForm[newOrderResponseIndexes[NO_DID].iStartIndex]);

    /* put the date in if valid */
    PutNumeric(pNewOrder->o_entry_d.day, 2, &szDate[0]);
    PutNumeric(pNewOrder->o_entry_d.month, 2, &szDate[3]);
    PutNumeric(pNewOrder->o_entry_d.year, 4, &szDate[6]);
    PutNumeric(pNewOrder->o_entry_d.hour, 2, &szDate[11]);
    PutNumeric(pNewOrder->o_entry_d.minute, 2, &szDate[14]);
    PutNumeric(pNewOrder->o_entry_d.second, 2, &szDate[17]);

memcpy(&newOrderForm[newOrderResponseIndexes[NO_DATE].iStartIndex],
             szDate, newOrderResponseIndexes[NO_DATE].iLen);
  }
  else
  {
    /* put in blanks for the date if not valid */

memcpy(&newOrderForm[newOrderResponseIndexes[NO_DATE].iStartIndex],
             szBlanks, newOrderResponseIndexes[NO_DATE].iLen);
  }
  /* put in value for the customer id. */
  PutNumeric(pNewOrder->c_id,
               newOrderResponseIndexes[NO_CID].iLen,

&newOrderForm[newOrderResponseIndexes[NO_CID].iStartIndex]);

  /* put in the values for the last name and  credit rating */
  PUT_STRING(pNewOrder->c_last,
               newOrderResponseIndexes[NO_LAST].iLen,
               newOrderResponseIndexes[NO_LAST].iStartIndex,
               StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pNewOrder->c_credit,
               newOrderResponseIndexes[NO_CREDIT].iLen,
               newOrderResponseIndexes[NO_CREDIT].iStartIndex,
               StrStruct[ssCnt]);
  ssCnt++;
```

```
  if(bValid)
  {
    /* put in the values */
    PutFloat2(pNewOrder->c_discount,
                newOrderResponseIndexes[NO_DISC].iLen,
&newOrderForm[newOrderResponseIndexes[NO_DISC].iStartIndex]);
    PutNumeric(pNewOrder->o_id,
                newOrderResponseIndexes[NO_OID].iLen,
&newOrderForm[newOrderResponseIndexes[NO_OID].iStartIndex]);
    PutNumeric(pNewOrder->o_ol_cnt,
                newOrderResponseIndexes[NO_LINES].iLen,
&newOrderForm[newOrderResponseIndexes[NO_LINES].iStartIndex]);
    PutFloat2(pNewOrder->w_tax,
                newOrderResponseIndexes[NO_W_TAX].iLen,
&newOrderForm[newOrderResponseIndexes[NO_W_TAX].iStartIndex]);
    PutFloat2(pNewOrder->d_tax,
                newOrderResponseIndexes[NO_D_TAX].iLen,
&newOrderForm[newOrderResponseIndexes[NO_D_TAX].iStartIndex]);

    for(i=0; i<pNewOrder->o_ol_cnt; i++)
    {
      PutNumeric(pNewOrder->o_ol[i].ol_supply_w_id,
                  newOrderResponseIndexes[NO_S_WID+(i*8)].iLen,
&newOrderForm[newOrderResponseIndexes[NO_S_WID+(i*8)].iStartIndex])
;
      PutNumeric(pNewOrder->o_ol[i].ol_i_id,
                  newOrderResponseIndexes[NO_IID+(i*8)].iLen,
&newOrderForm[newOrderResponseIndexes[NO_IID+(i*8)].iStartIndex]);
      PUT_STRING(pNewOrder->o_ol[i].i_name,
                  newOrderResponseIndexes[NO_INAME+(i*8)].iLen,
newOrderResponseIndexes[NO_INAME+(i*8)].iStartIndex,
                  StrStruct[ssCnt]);
      ssCnt++;
      PutNumeric(pNewOrder->o_ol[i].ol_quantity,
                  newOrderResponseIndexes[NO_QTY+(i*8)].iLen,
&newOrderForm[newOrderResponseIndexes[NO_QTY+(i*8)].iStartIndex]);
      PutNumeric(pNewOrder->o_ol[i].s_quantity,
                  newOrderResponseIndexes[NO_STOCK+(i*8)].iLen,
&newOrderForm[newOrderResponseIndexes[NO_STOCK+(i*8)].iStartIndex])
;
      PUT_STRING(pNewOrder->o_ol[i].b_g,
                  newOrderResponseIndexes[NO_BG+(i*8)].iLen,
newOrderResponseIndexes[NO_BG+(i*8)].iStartIndex,
                  StrStruct[ssCnt]);
      ssCnt++;
memcpy(&newOrderForm[newOrderResponseIndexes[NO_PRICE+(i*8)].iStart
Index-1],
             szDollar, 1);
      PutFloat2(pNewOrder->o_ol[i].i_price,
                 newOrderResponseIndexes[NO_PRICE+(i*8)].iLen,
          &newOrderForm[newOrderResponseIndexes[NO_PRICE+(i*8)].iSt
artIndex]);
memcpy(&newOrderForm[newOrderResponseIndexes[NO_AMT+(i*8)].iStartIn
dex-1],
             szDollar, 1);
      PutFloat2(pNewOrder->o_ol[i].ol_amount,
                 newOrderResponseIndexes[NO_AMT+(i*8)].iLen,
          &newOrderForm[newOrderResponseIndexes[NO_AMT+(i*8)].iStar
tIndex]);
    }
    /* need to blank out the rest of the unused item rows */
    jj = NO_AMT + ((i-1)*8) + 1;
    for(kk=i; kk<15; kk++)
    {
      /* there are 8 items per row - 6 plain and 2 with $*/
      for(mm=0; mm<6; mm++)
      {
          memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartIn
dex],
                 szBlanks, newOrderResponseIndexes[jj].iLen);
          jj++;
      }
      /* blank out the '$' for the blank $values */
      for(mm=0; mm<2; mm++)
      {
          memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartIn
dex-1],
                 szBlanks, newOrderResponseIndexes[jj].iLen+1);
          jj++;
      }
    }
  }
  else
  {
    /* will need to blank out any fields not entered when not valid
*/
    /* space for discount */
memcpy(&newOrderForm[newOrderResponseIndexes[NO_DISC].iStartIndex],
            szBlanks, newOrderResponseIndexes[NO_DISC].iLen);
    /*the actual order number */
    PutNumeric(pNewOrder->o_id,
                newOrderResponseIndexes[NO_OID].iLen,
&newOrderForm[newOrderResponseIndexes[NO_OID].iStartIndex]);
    /* space for number of lines, w_tax, and d_tax */
    for(kk=0; kk<3; kk++)
    {
memcpy(&newOrderForm[newOrderResponseIndexes[NO_LINES+kk].iStartInd
ex],
             szBlanks,
newOrderResponseIndexes[NO_LINES+kk].iLen);
    }
    /* spaces for each of the fields in the row items */
    jj = NO_S_WID;
    for(kk=0; kk<15; kk++)
    {
      /* there are 8 items per row - 6 plain and 2 with $*/
      for(mm=0; mm<6; mm++)
      {
          memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartIn
dex],
                 szBlanks, newOrderResponseIndexes[jj].iLen);
          jj++;
      }
      /* blank out the '$' for the blank $values */
      for(mm=0; mm<2; mm++)
      {
          memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartIn
dex-1],
                 szBlanks, newOrderResponseIndexes[jj].iLen+1);
          jj++;
      }

    }
  }

  /* output the execution status */
  PUT_STRING(execution_status,
newOrderResponseIndexes[NO_STAT].iLen,
              newOrderResponseIndexes[NO_STAT].iStartIndex,
              StrStruct[ssCnt]);
  ssCnt++;

  if(bValid)
  {
    /* total */
    PutFloat2(pNewOrder->total_amount,
                newOrderResponseIndexes[NO_TOTAL].iLen,
&newOrderForm[newOrderResponseIndexes[NO_TOTAL].iStartIndex]);
  }
  else
  {
    /* put blanks for total */
memcpy(&newOrderForm[newOrderResponseIndexes[NO_TOTAL].iStartIndex]
,
            szBlanks, newOrderResponseIndexes[NO_TOTAL].iLen);
  }
  PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);
  PutHTMLStrings(StrStruct, newOrderForm,
giResponseLen[NEW_ORDER_RESPONSE],
                 &szOutput, &iOutputLen);

#ifdef FFE_DEBUG
  pNewOrder->iStage |= UNRESERVING;
#endif

  UNRESERVE_TRANSACTION_STRUCT( NEW_ORDER_TRANS, pNewOrder );

  SendResponse(pECB, szOutput, iOutputLen);

  UNRESERVE_RESPONSE( NEW_ORDER_RESPONSE, newOrderForm );

  if( szOutput != newOrderForm )
    UNRESERVE_PANIC_FORM( szOutput );
}

/* FUNCTION: void TPCCPaymentResponse(EXTENSION_CONTROL_BLOCK
*pECB,
 *                                          int retcode,
 *                                          PaymentData
*paymentData)
 *
 * PURPOSE:         This function fills in the values and returns
the
 *                  response form to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     pointer
to structure that
 *
 *          contains internet service
 *
 *          information.
 *                  int                 retcode   return status
from the db call
 *                  PaymentData         *paymentData        pointer
to structure containing
 *                                                          the
data for this transaction.
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */
```

```
void
TPCCPaymentResponse( int retcode, pPaymentData pPayment )
{
  char    *ptr;
  char    szTmp[64];
  char    szcdata[4][64];
  char    szW_Zip[26];
  char    szD_Zip[26];
  char    szC_Zip[26];
  char    szC_Phone[26];
  int     i;
  int     l;
  char    *szZipPic = "XXXXX-XXXX";
  char  szLongDate[] = "XX-XX-XXXX XX:XX:XX";
  char  szDate[]   = "xx-xx-xxxx";
  char  szBlanks[] = "
";
  PutStrStruct StrStruct[34];
  int ssCnt = 0;
  char *paymentForm;
  char *szOutput;
  int iOutputLen;
  EXTENSION_CONTROL_BLOCK *pECB;

  pECB = pPayment->pCC;

  if ( ERR_DB_PENDING == retcode )
  {
    return;
  }
  else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
  {
    SendErrorResponse( pECB, ERR_PAYMENT_NOT_PROCESSED,
                       ERR_TYPE_WEBDLL, NULL,
                       pPayment->w_id, pPayment->ld_id,
                       (pConnData)pPayment );
    return;
  }
  else if ( ERR_DB_NOT_COMMITED == retcode )
  {
    SendErrorResponse( pECB, ERR_PAYMENT_INVALID_CUSTOMER,
                       ERR_TYPE_WEBDLL, NULL,
                       pPayment->w_id, pPayment->ld_id,
                       (pConnData)pPayment );
    return;
  }
  else if ( ERR_DB_SUCCESS != retcode )
  {
    SendErrorResponse( pECB, ERR_DB_ERROR,
                       ERR_TYPE_WEBDLL, NULL,
                       pPayment->w_id, pPayment->ld_id,
                       (pConnData)pPayment );
    return;
  }

  RESERVE_RESPONSE( PAYMENT_RESPONSE,  paymentForm );

  PutNumeric(WDID(pPayment->w_id,pPayment->ld_id),
             paymentResponseIndexes[PT_WDID].iLen,

&paymentForm[paymentResponseIndexes[PT_WDID].iStartIndex]);
  PutNumeric(pPayment->h_date.day, 2,
             &szLongDate[0]);
  PutNumeric(pPayment->h_date.month, 2,
             &szLongDate[3]);
  PutNumeric(pPayment->h_date.year, 4,
             &szLongDate[6]);
  PutNumeric(pPayment->h_date.hour, 2,
             &szLongDate[11]);
  PutNumeric(pPayment->h_date.minute, 2,
             &szLongDate[14]);
  PutNumeric(pPayment->h_date.second, 2,
             &szLongDate[17]);

memcpy(&paymentForm[paymentResponseIndexes[PT_LONG_DATE].iStartInde
x],
         szLongDate, paymentResponseIndexes[PT_LONG_DATE].iLen);

  PutNumeric(pPayment->w_id,
             paymentResponseIndexes[PT_WID].iLen,

&paymentForm[paymentResponseIndexes[PT_WID].iStartIndex]);
  PutNumeric(pPayment->d_id,
             paymentResponseIndexes[PT_DID].iLen,

&paymentForm[paymentResponseIndexes[PT_DID].iStartIndex]);

  PUT_STRING(pPayment->w_street_1,
             paymentResponseIndexes[PT_W_ST_1].iLen,
             paymentResponseIndexes[PT_W_ST_1].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pPayment->d_street_1,
             paymentResponseIndexes[PT_D_ST_1].iLen,
             paymentResponseIndexes[PT_D_ST_1].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pPayment->w_street_2,
             paymentResponseIndexes[PT_W_ST_2].iLen,
             paymentResponseIndexes[PT_W_ST_2].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pPayment->d_street_2,
             paymentResponseIndexes[PT_D_ST_2].iLen,
             paymentResponseIndexes[PT_D_ST_2].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pPayment->w_city,
             paymentResponseIndexes[PT_W_CITY].iLen,
             paymentResponseIndexes[PT_W_CITY].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pPayment->w_state,
             paymentResponseIndexes[PT_W_ST].iLen,
             paymentResponseIndexes[PT_W_ST].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  FormatString(szW_Zip, szZipPic, pPayment->w_zip);

memcpy(&paymentForm[paymentResponseIndexes[PT_W_ZIP].iStartIndex],
          szW_Zip, paymentResponseIndexes[PT_W_ZIP].iLen);
  PUT_STRING(pPayment->d_city,
             paymentResponseIndexes[PT_D_CITY].iLen,
             paymentResponseIndexes[PT_D_CITY].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pPayment->d_state,
             paymentResponseIndexes[PT_D_ST].iLen,
             paymentResponseIndexes[PT_D_ST].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  FormatString(szD_Zip, szZipPic, pPayment->d_zip);

memcpy(&paymentForm[paymentResponseIndexes[PT_D_ZIP].iStartIndex],
           szD_Zip, paymentResponseIndexes[PT_D_ZIP].iLen);
  PutNumeric(pPayment->c_id,
             paymentResponseIndexes[PT_CID].iLen,

&paymentForm[paymentResponseIndexes[PT_CID].iStartIndex]);
  PutNumeric(pPayment->c_w_id,
             paymentResponseIndexes[PT_C_WID].iLen,

&paymentForm[paymentResponseIndexes[PT_C_WID].iStartIndex]);
  PutNumeric(pPayment->c_d_id,
             paymentResponseIndexes[PT_C_DID].iLen,

&paymentForm[paymentResponseIndexes[PT_C_DID].iStartIndex]);

  PUT_STRING(pPayment->c_first,
             paymentResponseIndexes[PT_FIRST].iLen,
             paymentResponseIndexes[PT_FIRST].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pPayment->c_middle,
             paymentResponseIndexes[PT_MIDDLE].iLen,
             paymentResponseIndexes[PT_MIDDLE].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pPayment->c_last,
             paymentResponseIndexes[PT_LAST].iLen,
             paymentResponseIndexes[PT_LAST].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;

  PutNumeric(pPayment->c_since.day, 2, &szDate[0]);
  PutNumeric(pPayment->c_since.month, 2, &szDate[3]);
  PutNumeric(pPayment->c_since.year, 4, &szDate[6]);

memcpy(&paymentForm[paymentResponseIndexes[PT_SM_DATE].iStartIndex]
, szDate,
           paymentResponseIndexes[PT_SM_DATE].iLen);

  PUT_STRING(pPayment->c_street_1,
             paymentResponseIndexes[PT_C_STR_1].iLen,
             paymentResponseIndexes[PT_C_STR_1].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pPayment->c_credit,
             paymentResponseIndexes[PT_CREDIT].iLen,
             paymentResponseIndexes[PT_CREDIT].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;

  PUT_STRING(pPayment->d_street_2,
             paymentResponseIndexes[PT_D_STR_2].iLen,
             paymentResponseIndexes[PT_D_STR_2].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;

  PutFloat2(pPayment->c_discount,
            paymentResponseIndexes[PT_DISC].iLen,

&paymentForm[paymentResponseIndexes[PT_DISC].iStartIndex]);

  PUT_STRING(pPayment->c_city,
             paymentResponseIndexes[PT_C_CITY].iLen,
             paymentResponseIndexes[PT_C_CITY].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;

  PUT_STRING(pPayment->c_state,
             paymentResponseIndexes[PT_C_ST].iLen,
             paymentResponseIndexes[PT_C_ST].iStartIndex,
             StrStruct[ssCnt]);
  ssCnt++;

  FormatString(szC_Zip, szZipPic, pPayment->c_zip);

memcpy(&paymentForm[paymentResponseIndexes[PT_C_ZIP].iStartIndex],
szC_Zip,
           paymentResponseIndexes[PT_C_ZIP].iLen);
  FormatString(szC_Phone, "XXXXXX-XXX-XXX-XXXX",
               pPayment->c_phone);
```

```
memcpy(&paymentForm[paymentResponseIndexes[PT_C_PHONE].iStartIndex]
,
          szC_Phone, paymentResponseIndexes[PT_C_PHONE].iLen);

  PutFloat2(pPayment->h_amount,
             paymentResponseIndexes[PT_AMT].iLen,

&paymentForm[paymentResponseIndexes[PT_AMT].iStartIndex]);
  PutFloat2(pPayment->c_balance,
             paymentResponseIndexes[PT_BAL].iLen,

&paymentForm[paymentResponseIndexes[PT_BAL].iStartIndex]);

  PutFloat2(pPayment->c_credit_lim,
             paymentResponseIndexes[PT_LIM].iLen,

&paymentForm[paymentResponseIndexes[PT_LIM].iStartIndex]);

  ptr = pPayment->c_credit;
  if ( *ptr == 'B' && *(ptr+1) == 'C' )
  {
    ptr = pPayment->c_data;
    l = strlen( ptr ) / 50;
    for(i=0; i<4; i++, ptr += 50)
    {
      if ( i <= l )
      {
          strncpy(szcdata[i], ptr, 50);
          szcdata[i][50] = '\0';
      }
      else
          szcdata[i][0] = 0;

      PUT_STRING(szcdata[i],
                    paymentResponseIndexes[PT_CUST_DATA+i].iLen,

paymentResponseIndexes[PT_CUST_DATA+i].iStartIndex,
                    StrStruct[ssCnt]);
      ssCnt++;
    }
  }
  else
  {
    for(i=0; i<4; i++)
    {

memcpy(&paymentForm[paymentResponseIndexes[PT_CUST_DATA+i].iStartIn
dex],
              szBlanks,
paymentResponseIndexes[PT_CUST_DATA+i].iLen);
    }
  }

  PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);

  PutHTMLStrings(StrStruct, paymentForm,
giResponseLen[PAYMENT_RESPONSE],
                  &szOutput, &iOutputLen);

#ifdef FFE_DEBUG
  pPayment->iStage |= UNRESERVING;
#endif

  UNRESERVE_TRANSACTION_STRUCT( PAYMENT_TRANS, pPayment );

  SendResponse(pECB, szOutput, iOutputLen);

  UNRESERVE_RESPONSE( PAYMENT_RESPONSE, paymentForm );

  if( szOutput != paymentForm )
    UNRESERVE_PANIC_FORM( szOutput );
}

/* FUNCTION: void TPCCOrderStatusResponse( int retcode,
 *
OrderStatusData *orderStatusData)
 *
 * PURPOSE:         This function fills in the values and returns
the
 *                  response form to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     pointer
to structure containing
 *
         internet service information.
 *                  int                retcode return status from
db call
 *                  OrderStatusData *orderStatusData        pointer
to structure
 *
         of data for this txn.
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */

void
TPCCOrderStatusResponse( int retcode, pOrderStatusData pOrderStatus
)
{
  int     i;
  int     jj;
  int     kk;
  int     mm;
  char  szLongDate[] = "XX-XX-XXXX XX:XX:XX";
  char  szDate[]   = "XX-XX-XXXX";
  char  szBlanks[] = "                          ";
  char  szDollar[] = "$";
  PutStrStruct StrStruct[4];
  int   ssCnt = 0;
  char    *orderStatusForm;
  char    *szOutput;
  int     iOutputLen;
  EXTENSION_CONTROL_BLOCK *pECB;

  pECB = pOrderStatus->pCC;

  if ( ERR_DB_PENDING == retcode )
  {
    return;
  }
  else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
  {
    SendErrorResponse( pECB, ERR_ORDER_STATUS_NOT_PROCESSED,
                           ERR_TYPE_WEBDLL, NULL,
                           pOrderStatus->w_id, pOrderStatus->ld_id,
                           (pConnData)pOrderStatus );
    return;
  }
  else if ( ERR_DB_NOT_COMMITED == retcode )
  {
    SendErrorResponse( pECB, ERR_NOSUCH_CUSTOMER,
                           ERR_TYPE_WEBDLL, NULL,
                           pOrderStatus->w_id, pOrderStatus->ld_id,
                           (pConnData)pOrderStatus );
    return;
  }
  else    if ( ERR_DB_SUCCESS != retcode )
  {
    SendErrorResponse( pECB, ERR_DB_ERROR,
                           ERR_TYPE_WEBDLL, NULL,
                           pOrderStatus->w_id, pOrderStatus->ld_id,
                           (pConnData)pOrderStatus );
  return;
  }

  RESERVE_RESPONSE( ORDER_STATUS_RESPONSE, orderStatusForm );

  PutNumeric(WDID(pOrderStatus->w_id,pOrderStatus->ld_id),
               orderStatusResponseIndexes[OS_WDID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_WDID].iStartIndex]);
  PutNumeric(pOrderStatus->w_id,
               orderStatusResponseIndexes[OS_WID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_WID].iStartIndex]);
  PutNumeric(pOrderStatus->d_id,
               orderStatusResponseIndexes[OS_DID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_DID].iStartIndex]);
  PutNumeric(pOrderStatus->c_id,
               orderStatusResponseIndexes[OS_CID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_CID].iStartIndex]);
  PUT_STRING(pOrderStatus->c_first,
               orderStatusResponseIndexes[OS_FIRST].iLen,
               orderStatusResponseIndexes[OS_FIRST].iStartIndex,
StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pOrderStatus->c_middle,
               orderStatusResponseIndexes[OS_MIDDLE].iLen,
               orderStatusResponseIndexes[OS_MIDDLE].iStartIndex,
               StrStruct[ssCnt]);
  ssCnt++;
  PUT_STRING(pOrderStatus->c_last,
               orderStatusResponseIndexes[OS_LAST].iLen,
               orderStatusResponseIndexes[OS_LAST].iStartIndex,
StrStruct[ssCnt]);
  ssCnt++;
  PutFloat2(pOrderStatus->c_balance,
              orderStatusResponseIndexes[OS_BAL].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_BAL].iStartIndex]);
  PutNumeric(pOrderStatus->o_id,
               orderStatusResponseIndexes[OS_OID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_OID].iStartIndex]);

  PutNumeric(pOrderStatus->o_entry_d.day, 2, &szLongDate[0]);
  PutNumeric(pOrderStatus->o_entry_d.month, 2, &szLongDate[3]);
  PutNumeric(pOrderStatus->o_entry_d.year, 4, &szLongDate[6]);
  PutNumeric(pOrderStatus->o_entry_d.hour, 2, &szLongDate[11]);
  PutNumeric(pOrderStatus->o_entry_d.minute, 2, &szLongDate[14]);
  PutNumeric(pOrderStatus->o_entry_d.second, 2, &szLongDate[17]);

memcpy(&orderStatusForm[orderStatusResponseIndexes[OS_DATE].iStartI
ndex],
          szLongDate, orderStatusResponseIndexes[OS_DATE].iLen);
  PutNumeric(pOrderStatus->o_carrier_id,
               orderStatusResponseIndexes[OS_CAR_ID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_CAR_ID].iStartIndex]
);

  for(i=0; i<pOrderStatus->o_ol_cnt; i++)
  {
    PutNumeric(pOrderStatus->s_ol[i].ol_supply_w_id,
                 orderStatusResponseIndexes[OS_S_WID+(i*5)].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_S_WID+(i*5)].iStartI
ndex]);
    PutNumeric(pOrderStatus->s_ol[i].ol_i_id,
                 orderStatusResponseIndexes[OS_IID+(i*5)].iLen,
```

```
&orderStatusForm[orderStatusResponseIndexes[OS_IID+(i*5)].iStartInd
ex]);
    PutNumeric(pOrderStatus->s_ol[i].ol_quantity,
                orderStatusResponseIndexes[OS_QTY+(i*5)].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_QTY+(i*5)].iStartInd
ex]);

memcpy(&orderStatusForm[orderStatusResponseIndexes[OS_AMT+(i*5)].iS
tartIndex-1],
               szDollar, 1);
    PutFloat2(pOrderStatus->s_ol[i].ol_amount,
               orderStatusResponseIndexes[OS_AMT+(i*5)].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_AMT+(i*5)].iStartInd
ex]);
    PutNumeric(pOrderStatus->s_ol[i].ol_delivery_d.day,
                2, &szDate[0]);
    PutNumeric(pOrderStatus->s_ol[i].ol_delivery_d.month,
                2, &szDate[3]);
    PutNumeric(pOrderStatus->s_ol[i].ol_delivery_d.year,
                4, &szDate[6]);

memcpy(&orderStatusForm[orderStatusResponseIndexes[OS_SM_DATE+(i*5)
].iStartIndex],
               szDate,
orderStatusResponseIndexes[OS_SM_DATE+(i*5)].iLen);
  }
  /* need to blank out the rest of the unused item rows */
  jj = OS_SM_DATE + ((i-1)*5) + 1;
  for(kk=i; kk<15; kk++)
  {
    /* there are 5 items per row - 4 plain and 1 with $*/
    for(mm=0; mm<3; mm++)
    {

memcpy(&orderStatusForm[orderStatusResponseIndexes[jj].iStartIndex]
,
                szBlanks, orderStatusResponseIndexes[jj].iLen);
      jj++;
    }
    /* blank out the '$' for the blank $values */

memcpy(&orderStatusForm[orderStatusResponseIndexes[jj].iStartIndex-
1],
               szBlanks, orderStatusResponseIndexes[jj].iLen+1);
    jj++;

memcpy(&orderStatusForm[orderStatusResponseIndexes[jj].iStartIndex]
,
               szBlanks, orderStatusResponseIndexes[jj].iLen);
    jj++;
  }

  PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);
  PutHTMLStrings(StrStruct, orderStatusForm,
                  giResponseLen[ORDER_STATUS_RESPONSE],
                  &szOutput, &iOutputLen);

#ifdef FFE_DEBUG
  pOrderStatus->iStage |= UNRESERVING;
#endif

  UNRESERVE_TRANSACTION_STRUCT( ORDER_STATUS_TRANS, pOrderStatus );

  SendResponse(pECB, szOutput, iOutputLen);

  UNRESERVE_RESPONSE( ORDER_STATUS_RESPONSE, orderStatusForm );

  if( szOutput != orderStatusForm )
    UNRESERVE_PANIC_FORM( szOutput );
}

/* FUNCTION: void TPCCStockLevelResponse(int retcode,
 *                                                    StockLevelData
*stockLevelData)
 *
 * PURPOSE:         This function puts the response data for the
transaction
 *                  into the form and sends the form back to the
browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     pointer
to structure containing
 *
 *          internet service information.
 *                  int                 retcode return status from
db call
 *                  StockLevelData      *stockLevelData     pointer
to structure containing
 *                                                      data
for this transaction.
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */

void
TPCCStockLevelResponse( int retcode, StockLevelData *pStockLevel )
{
  char *stockLevelForm;
  EXTENSION_CONTROL_BLOCK *pECB;

  pECB = pStockLevel->pCC;

  if ( ERR_DB_PENDING == retcode )
  {
    return;
  }
  else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
  {
    SendErrorResponse( pECB, ERR_STOCKLEVEL_NOT_PROCESSED,
                            ERR_TYPE_WEBDLL, NULL,
                            pStockLevel->w_id, pStockLevel->ld_id,
                            (pConnData)pStockLevel );
    return;
  }
  else if ( ERR_DB_SUCCESS != retcode )
  {
    SendErrorResponse( pECB, ERR_DB_ERROR,
                            ERR_TYPE_WEBDLL, NULL,
                            pStockLevel->w_id, pStockLevel->ld_id,
                            (pConnData)pStockLevel );
    return;
  }

  RESERVE_RESPONSE( STOCK_LEVEL_RESPONSE, stockLevelForm );

  PutNumeric(WDID(pStockLevel->w_id,pStockLevel->ld_id),
              stockLevelResponseIndexes[SL_WDID].iLen,

&stockLevelForm[stockLevelResponseIndexes[SL_WDID].iStartIndex]);
  PutNumeric(pStockLevel->w_id,
              stockLevelResponseIndexes[SL_WID].iLen,

&stockLevelForm[stockLevelResponseIndexes[SL_WID].iStartIndex]);
  PutNumeric(pStockLevel->ld_id,
              stockLevelResponseIndexes[SL_DID].iLen,

&stockLevelForm[stockLevelResponseIndexes[SL_DID].iStartIndex]);
  PutNumeric(pStockLevel->threshold,
              stockLevelResponseIndexes[SL_TH].iLen,

&stockLevelForm[stockLevelResponseIndexes[SL_TH].iStartIndex]);
  PutNumeric(pStockLevel->low_stock,
              stockLevelResponseIndexes[SL_LOW].iLen,

&stockLevelForm[stockLevelResponseIndexes[SL_LOW].iStartIndex]);

#ifdef FFE_DEBUG
  pStockLevel->iStage |= UNRESERVING;
#endif

  UNRESERVE_TRANSACTION_STRUCT( STOCK_LEVEL_TRANS, pStockLevel );

  SendResponse(pECB, stockLevelForm,
                giResponseLen[STOCK_LEVEL_RESPONSE]);

  UNRESERVE_RESPONSE( STOCK_LEVEL_RESPONSE, stockLevelForm );
}


/* FUNCTION: void TPCCResponseComplete( EXTENSION_CONTROL_BLOCK
*pECB )
 *
 * PURPOSE:
 *    This function completes the asynchronous web transaction.
 *
 * ARGUMENTS:
 *    EXTENSION_CONTROL_BLOCK *pECB               Server context
structure.
 *
 * RETURNS:
 *    None
 *
 * COMMENTS:
 */

void
TPCCResponseComplete( EXTENSION_CONTROL_BLOCK *pECB )
{
  DWORD   status;

  status = HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  (pECB->ServerSupportFunction)(pECB->ConnID,
HSE_REQ_DONE_WITH_SESSION,
                                &status, NULL, NULL);
}

/* FUNCTION: int ProcessDeliveryQuery( EXTENSION_CONTROL_BLOCK
*pECB,
 *
 * PURPOSE:         This function parses the query string,
validates the data,
 *                  and sends the request to the db/transport and
returns
 *                  a response to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     ptr to
the structure
 *
 *          containing the internet server
 *
 *          information.
 *
 * RETURNS:         int                 status
 *
 * COMMENTS:        None
 *
 */

int
ProcessDeliveryQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
```

```
                        int w_id, int ld_id )
{
  int                       retcode;
  char                      *ptr;
  char                      *deliveryVals[MAXDELIVERYVALS];
  pDeliveryData             pDelivery;
  pDeliveryData
          CompletedDeliveries[DELIVERY_RESPONSE_COUNT];

  RESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS, pDelivery );

  pDelivery->w_id = w_id;
  pDelivery->ld_id = ld_id;
  pDelivery->pCC = pECB;

  PARSE_QUERY_STRING(lpszQueryString, MAXDELIVERYVALS,
                     deliveryStrs, deliveryVals);

  if ( !GetValuePtr(deliveryVals, QUEUETIME, &ptr) )
    return ERR_DELIVERY_MISSING_QUEUETIME_KEY;

  if ( !GetNumeric(ptr, &pDelivery->queue_time) )
    return ERR_DELIVERY_QUEUETIME_INVALID;

  if ( !GetValuePtr(deliveryVals, OCD, &ptr) )
    return ERR_DELIVERY_MISSING_OCD_KEY;

  if ( !GetNumeric(ptr, &pDelivery->o_carrier_id) )
    return ERR_DELIVERY_CARRIER_INVALID;

  if ( pDelivery->o_carrier_id > 10 || pDelivery->o_carrier_id < 1
)
    return ERR_DELIVERY_CARRIER_ID_RANGE;

#ifdef FFE_DEBUG
  pDelivery->iStage |= CALLING_LH;
#endif
  retcode = TPCCDelivery( pDelivery, CompletedDeliveries );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
  pDelivery->iStage |= CALLING_RESP;
#endif
  TPCCDeliveryResponse( retcode, pDelivery, CompletedDeliveries );

  return retcode;
}

/* FUNCTION: int ProcessNewOrderQuery( EXTENSION_CONTROL_BLOCK
*pECB,
 *
 * PURPOSE:         This function parses the query string,
validates the data,
 *                  and sends the request to the db/transport and
returns
 *                  a response to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     ptr to
structure containing
 *
          internet server info
 *
 * RETURNS:         int                 status
 *
 * COMMENTS:        None
 *
 */

int
ProcessNewOrderQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                      int w_id, int ld_id )
{
  int                       retcode;
  NewOrderData              *pNewOrder;

  RESERVE_TRANSACTION_STRUCT( NEW_ORDER_TRANS, pNewOrder );

  pNewOrder->w_id = w_id;
  pNewOrder->ld_id = ld_id;
  pNewOrder->pCC = pECB;

  if ( ERR_SUCCESS != ( retcode = ParseNewOrderQuery(
lpszQueryString,

pNewOrder )))
    return retcode;

#ifdef FFE_DEBUG
  pNewOrder->iStage |= CALLING_LH;
#endif
  retcode = TPCCNewOrder( pNewOrder );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
  pNewOrder->iStage |= CALLING_RESP;
#endif
  TPCCNewOrderResponse( retcode, pNewOrder );

  return retcode;
}

/* FUNCTION: int ProcessOrderStatusQuery( EXTENSION_CONTROL_BLOCK
*pECB,
 *
 * PURPOSE:         This function parses the query string,
validates the data,
 *                  and sends the request to the db/transport and
returns
 *                  a response to the browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     ptr to
structure that contains
 *                                                             the
internet server info.
 *
 * RETURNS:         int                 status
 *
 * COMMENTS:        None
 *
 */

int
ProcessOrderStatusQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                         int w_id, int ld_id )
{
  int                       retcode;
  OrderStatusData   *pOrderStatus;

  RESERVE_TRANSACTION_STRUCT( ORDER_STATUS_TRANS, pOrderStatus );

  pOrderStatus->w_id = w_id;
  pOrderStatus->ld_id = ld_id;
  pOrderStatus->pCC = pECB;

  if( ERR_SUCCESS != ( retcode = ParseOrderStatusQuery(
lpszQueryString,

          pOrderStatus )))
    return retcode;

#ifdef FFE_DEBUG
  pOrderStatus->iStage |= CALLING_LH;
#endif
  retcode = TPCCOrderStatus( pOrderStatus );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
  pOrderStatus->iStage |= CALLING_RESP;
#endif
  TPCCOrderStatusResponse( retcode, pOrderStatus );

  return retcode;
}

/* FUNCTION: int ProcessPaymentQuery( EXTENSION_CONTROL_BLOCK
*pECB,
 *
 * PURPOSE:         This function gets and validates the input data
from the
 *                  payment form filling in the required input
variables.
 *                  It then calls the SQLPayment transaction,
constructs the
 *                  output form and writes it back to client
browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     ptr to
structure that contains
 *                                                             the
internet server info.
 *
 * RETURNS:         int                 status
 *
 * COMMENTS:        None
 *
 */

int
ProcessPaymentQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                           int w_id, int ld_id )
{
  int                       retcode;
  PaymentData               *pPayment;

  RESERVE_TRANSACTION_STRUCT( PAYMENT_TRANS, pPayment );

  pPayment->w_id = w_id;
  pPayment->ld_id = ld_id;
  pPayment->pCC = pECB;

  if( ERR_SUCCESS != ( retcode = ParsePaymentQuery(
lpszQueryString,

pPayment )))
    return retcode;

#ifdef FFE_DEBUG
  pPayment->iStage |= CALLING_LH;
#endif
  retcode = TPCCPayment( pPayment );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
  pPayment->iStage |= CALLING_RESP;
#endif
  TPCCPaymentResponse( retcode, pPayment );

  return retcode;
}

/* FUNCTION: int ProcessStockLevelQuery( EXTENSION_CONTROL_BLOCK
*pECB,
```

```
 *
 * PURPOSE:         This function gets and validates the input data
from the
 *                  Stock Level form filling in the required input
variables.
 *                  It then calls the SQLStockLevel transaction,
constructs
 *                  the output form and writes it back to client
browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     ptr to
structure that contains
 *                                                          the
internet server info.
 *                  int                 iSyncId             client
browser sync id
 *
 * RETURNS:         int                 status
 *
 * COMMENTS:        None
 *
 */

int
ProcessStockLevelQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                        int w_id, int ld_id )
{
  char          *ptr;
  int               retcode;
  char              *stockLevelVals[MAXSTOCKLEVELVALS];
  StockLevelData    *pStockLevel;

  RESERVE_TRANSACTION_STRUCT( STOCK_LEVEL_TRANS, pStockLevel );

  pStockLevel->w_id = w_id;
  pStockLevel->ld_id = ld_id;
  pStockLevel->pCC = pECB;

  PARSE_QUERY_STRING(lpszQueryString, MAXSTOCKLEVELVALS,
                     stockLevelStrs, stockLevelVals);

  if ( !GetValuePtr(stockLevelVals, TT, &ptr))
    return ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY;

  if ( !GetNumeric(ptr, &pStockLevel->threshold) )
    return ERR_STOCKLEVEL_THRESHOLD_INVALID;

  if ( pStockLevel->threshold >= 100 || pStockLevel->threshold < 0
)
    return ERR_STOCKLEVEL_THRESHOLD_RANGE;

#ifdef FFE_DEBUG
  pStockLevel->iStage |= CALLING_LH;
#endif
  retcode = TPCCStockLevel( pStockLevel );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
  pStockLevel->iStage |= CALLING_RESP;
#endif
  TPCCStockLevelResponse( retcode, pStockLevel );

  return retcode;
}

/* FUNCTION: BOOL GetValuePtr(char *pProcessedQuery[], int iIndex,
 *                                    char **pValue)
 *
 * PURPOSE:         This function passes back a pointer to the char
ptr to the
 *                  value requested.
 *
 * ARGUMENTS:       char  *pProcessedQuery[]   char* array of query
string values
 *              int   iIndex    index into the ProcessedQuery array
 *                char *pValue          character ptr into to the
key's value
 *
 * RETURNS:         BOOL      FALSE     there is no valid ptr for
this value
 *                        TRUE        the ptr returned is valid
 *
 *
 * COMMENTS:        none.
 */

BOOL
GetValuePtr(char *pProcessedQuery[], int iIndex, char **pValue)
{
  *pValue = pProcessedQuery[iIndex];

  if(NULL == *pValue)return FALSE;

  return TRUE;
}

/* FUNCTION: void MakeDeliveryTemplates( char *deliveryForm,
 *                                                    char
*deliveryResponse )
 *
 * PURPOSE:         This function constructs the templates for the
 *                  Delivery input and response HTML forms.
 *
 * ARGUMENTS:       char *deliveryForm  pointer to the HTML input
form.
 *                  char *deliveryResponse        pointer to the
HTML response form.
 *
 * RETURNS:         None
 *
 * COMMENTS:        None
 */

void
MakeDeliveryTemplates( char *deliveryForm, char *deliveryResponse )
{
  int     curLen;

  /* first make the input form template */
  curLen = sprintf(deliveryForm, szFormTemplate, szModName);
  ParseTemplateString(deliveryForm, &curLen, szDeliveryFormTemp2i,
                      deliveryFormIndexesI);
  giFormLen[DELIVERY_FORM] = curLen;

  /* now make the process form template */
  curLen = sprintf(deliveryResponse, szFormTemplate, szModName);
  ParseTemplateString(deliveryResponse, &curLen,
szDeliveryFormTemp2p,
                      deliveryFormIndexesP);
  giResponseLen[DELIVERY_RESPONSE] = curLen;
}

/* FUNCTION: void MakeNewOrderTemplates(char *newOrderForm,
                                                        char
*newOrderResponse )
 *
 * PURPOSE:         This function constructs the templates for both
the input
 *                  and the response HTML forms for NewOrder
function.
 *
 * ARGUMENTS:       char      *newOrderForm       pointer to the
input HTML form.
 *                  char      *newOrderResponse pointer to the
response HTML form.
 *
 * RETURNS:         none
 *
 * COMMENTS:        none.
 */

void
MakeNewOrderTemplates( char *newOrderForm, char *newOrderResponse )
{
  int     curLen;

  /* first make the input template */
  curLen = sprintf(newOrderForm, szFormTemplate, szModName);
  ParseTemplateString(newOrderForm, &curLen, szNewOrderFormTemp2i,
                      newOrderFormIndexes);
  giFormLen[NEW_ORDER_FORM] = curLen;

  /* now make the process template */
  curLen = sprintf(newOrderResponse, szFormTemplate, szModName);
  ParseTemplateString(newOrderResponse, &curLen,
szNewOrderFormTemp2p,
                      newOrderResponseIndexes);
  giResponseLen[NEW_ORDER_RESPONSE] = curLen;
}

/* FUNCTION: void MakeOrderStatusTemplates(char *orderStatusForm,
 *                                         char
*orderStatusResponse)
 *
 * PURPOSE:         This function constructs the template HTML
forms
 *                  for Order Status.
 *
 * ARGUMENTS:       char *orderStatusForm              pointer to
the input HTML form
 *                  char *orderStatusResponse    pointer to the
response HTML form
 *
 * RETURNS:         none
 *
 * COMMENTS:   none
 */

void
MakeOrderStatusTemplates(char *orderStatusForm, char
*orderStatusResponse)
{
  int     curLen;

  /* first make the input form template */
  curLen = sprintf(orderStatusForm, szFormTemplate, szModName);
  ParseTemplateString(orderStatusForm, &curLen,
szOrderStatusFormTemp2i,
                      orderStatusFormIndexes);
  giFormLen[ORDER_STATUS_FORM] = curLen;

  /* now make the process template */
  curLen = sprintf(orderStatusResponse, szFormTemplate, szModName);
  ParseTemplateString(orderStatusResponse, &curLen,
szOrderStatusFormTemp2p,
                      orderStatusResponseIndexes);
  giResponseLen[ORDER_STATUS_RESPONSE] = curLen;
}

/* FUNCTION: void MakePaymentTemplates(char *paymentForm,
 *                                                  char
*paymentResponse)
 *
 * PURPOSE:         This function constructs the templates for the
 *                  Payment input and response HTML forms.
```

```
 *
 * ARGUMENTS:       char      *paymentForm         pointer to the
input HTML form.
 *                  char      *paymentResponse pointer to the
response HTML form.
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */

void
MakePaymentTemplates(char *paymentForm, char *paymentResponse)
{
  int     curLen;

  /* first make the input form template */
  curLen = sprintf(paymentForm, szFormTemplate, szModName);
  ParseTemplateString(paymentForm, &curLen, szPaymentFormTemp2i,
                      paymentFormIndexes);
  giFormLen[PAYMENT_FORM] = curLen;

  /* now make the process form template */
  curLen = sprintf(paymentResponse, szFormTemplate, szModName);
  ParseTemplateString(paymentResponse, &curLen,
szPaymentFormTemp2p,
                      paymentResponseIndexes);
  giResponseLen[PAYMENT_RESPONSE] = curLen;
}

/* FUNCTION: void MakeStockLevelTemplates(char *stockLevelForm,
 *                                        char *stockLevelResponse)
 *
 * PURPOSE:         This function constructs the templates for the
 *                  input and response Stock Level HTML pages.
 *
 * ARGUMENTS:       char      *stockLevelForm           pointer to
the input HTML form
 *                  char      *stockLevelResponse   pointer to the
response HTML form
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */
void
MakeStockLevelTemplates(char *stockLevelForm, char
*stockLevelResponse)
{
  int     curLen;

  /* first make the input template */
  curLen = sprintf(stockLevelForm, szFormTemplate, szModName);
  ParseTemplateString(stockLevelForm, &curLen,
szStockLevelFormTemp2i,
                      stockLevelFormIndexes);
  giFormLen[STOCK_LEVEL_FORM] = curLen;

  /* now make the process template */
  curLen = sprintf(stockLevelResponse, szFormTemplate, szModName);
  ParseTemplateString(stockLevelResponse, &curLen,
szStockLevelFormTemp2p,
                      stockLevelResponseIndexes);
  giResponseLen[STOCK_LEVEL_RESPONSE] = curLen;
}

/* FUNCTION: void MakeResponseHeader(void)
 *
 * PURPOSE:         This function constructs the HTML response
header.
 *
 * ARGUMENTS:       char      *responseString           pointer to
the header string
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */
void
MakeResponseHeader(void)
{
  ParseTemplateString(szResponseHeader, &responseHeaderLen,
                      szResponseHeaderTemplate,
responseHeaderIndexes);
}

/* FUNCTION: void MakePanicPool( DWORD dwResponseSize )
 *
 * PURPOSE:         This function builds the array of panic forms
to be used
 *                  by the threads as they need an oversize form,
or to report
 *                  an error.
 *
 * ARGUMENTS:       none
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */

void
MakePanicPool( DWORD dwResponseSize )
{
  int iMallocSize;
  char *pForm;
  DWORD ii;

  /* set up area for forms (including errors) that are built on the
fly. */
  iMallocSize = (((char *)&gpPanicForms->index - (char
*)gpPanicForms) +
                      (((char *)gpPanicForms->forms - (char
*)gpPanicForms->index)
                       * dwResponseSize) +
                      (((char *)&gpPanicForms-
>forms[PANIC_FORM_SIZE] -
                        (char *)&gpPanicForms->forms[0]) *
dwResponseSize));
  gpPanicForms = malloc( iMallocSize );
  InitializeCriticalSection( &gpPanicForms->critSec );
#ifdef FFE_DEBUG
  gpPanicForms->iMaxIndex = dwResponseSize - 1;
#endif
  gpPanicForms->iNextFree = 0;
  pForm =
    ((char *)&gpPanicForms->index[0] +
     (((char *)&gpPanicForms->forms[0] - (char *)&gpPanicForms-
>index[0]) *
      dwResponseSize));

  for( ii = 0; ii < dwResponseSize; ii++ )
  {
    gpPanicForms->index[ii] = pForm;
    pForm += PANIC_FORM_SIZE;
  }
}

/* FUNCTION: void DeletePanicPool( void )
 *
 * PURPOSE:         This function destroys the array of panic forms
to be used
 *                  by the threads as they need an oversize or
error form.
 *
 * ARGUMENTS:       none
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */
void
DeletePanicPool( void )
{
  DeleteCriticalSection( &gpPanicForms->critSec );
  free( gpPanicForms );
}

/* FUNCTION: void MakeTemplatePool( DWORD dwFormSize, DWORD
dwResponseSize )
 *
 * PURPOSE:         This function builds the array of forms to be
used
 *                  by the threads as they need a form.  The forms
are
 *                  reserved and released by each thread as needed.
 *
 * ARGUMENTS:       none
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */

void
MakeTemplatePool( DWORD dwFormSize, DWORD dwResponseSize )
{
  char szDeliveryForm[sizeof(szFormTemplate)+FILENAMESIZE+
                      sizeof(szDeliveryFormTemp2i)];
  char szNewOrderForm[sizeof(szFormTemplate)+FILENAMESIZE+
                      sizeof(szNewOrderFormTemp2i)];
  char szOrderStatusForm[sizeof(szFormTemplate)+FILENAMESIZE+
                         sizeof(szOrderStatusFormTemp2i)];
  char szPaymentForm[sizeof(szFormTemplate)+FILENAMESIZE+
                     sizeof(szPaymentFormTemp2i)];
  char szStockLevelForm[sizeof(szFormTemplate)+FILENAMESIZE+
                        sizeof(szStockLevelFormTemp2i)];
  char szDeliveryResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                          sizeof(szDeliveryFormTemp2p)];
  char szNewOrderResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                          sizeof(szNewOrderFormTemp2p)];
  char szOrderStatusResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                             sizeof(szOrderStatusFormTemp2p)];
  char szPaymentResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                         sizeof(szPaymentFormTemp2p)];
  char szStockLevelResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                            sizeof(szStockLevelFormTemp2p)];
  int     iFormLen[NUMBER_POOL_FORM_TYPES];
  int     iResponseLen[NUMBER_POOL_RESPONSE_TYPES];
  int iMallocSize;
  int iRowSize;
  DWORD ii;
  int jj;
  char *pForm;
  char *pResponse;

  /* now build the forms that are static */
  MakeDeliveryTemplates( szDeliveryForm, szDeliveryResponse );
  MakeNewOrderTemplates( szNewOrderForm, szNewOrderResponse );
  MakeOrderStatusTemplates( szOrderStatusForm,
szOrderStatusResponse );
  MakePaymentTemplates( szPaymentForm, szPaymentResponse );
  MakeStockLevelTemplates( szStockLevelForm, szStockLevelResponse
);
  MakeResponseHeader( );
```

```
  /* calculate the size of one row of forms */
  iRowSize = 0;
  for( jj = 0; jj < NUMBER_POOL_FORM_TYPES; jj++ )
  {
    iFormLen[jj] = ( giFormLen[jj] + 8 ) & ( ~(int)7 );
    iRowSize += iFormLen[jj];
  }

  iMallocSize = (((char *)&gpForms->index - (char *)gpForms) +
                 (((char *)gpForms->forms - (char *)gpForms->index)
                  * dwFormSize * NUMBER_POOL_FORM_TYPES ) +
                 (((char *)&gpForms->forms[iRowSize * dwFormSize] -
                   (char *)&gpForms->forms[0])));
  gpForms = malloc( iMallocSize );

  for( jj = 0; jj < NUMBER_POOL_FORM_TYPES; jj++ )
  {
    InitializeCriticalSection( &gpForms->critSec[jj] );
    gpForms->iNextFreeForm[jj] = 0;
    gpForms->iFirstFormIndex[jj] = jj * dwFormSize;
#ifdef FFE_DEBUG
    gpForms->iMaxIndex[jj] = dwFormSize - 1;
#endif
  }

  pForm = ((char *)&gpForms->index[0] +
           (((char *)&gpForms->forms[0] - (char *)&gpForms->index[0]) *
            NUMBER_POOL_FORM_TYPES * dwFormSize));
  for( ii = 0; ii < dwFormSize; ii++ )
  {
    for( jj = 0; jj < NUMBER_POOL_FORM_TYPES; jj++ )
    {
      gpForms->index[jj*dwFormSize+ii] = pForm;
      pForm += iFormLen[jj];
    }
  }

  /* load the first row with the templates */
  pForm = gpForms->index[0];

  memcpy( pForm, szDeliveryForm, iFormLen[DELIVERY_FORM] );
  pForm += iFormLen[DELIVERY_FORM];

  memcpy( pForm, szNewOrderForm, iFormLen[NEW_ORDER_FORM] );
  pForm += iFormLen[NEW_ORDER_FORM];

  memcpy( pForm, szOrderStatusForm, iFormLen[ORDER_STATUS_FORM] );
  pForm += iFormLen[ORDER_STATUS_FORM];

  memcpy( pForm, szPaymentForm, iFormLen[PAYMENT_FORM] );
  pForm += iFormLen[PAYMENT_FORM];

  memcpy( pForm, szStockLevelForm, iFormLen[STOCK_LEVEL_FORM] );
  pForm += iFormLen[STOCK_LEVEL_FORM];

  /* copy the first row to all the other rows */
  pForm = gpForms->index[0];
  for( ii = 1; ii < dwFormSize; ii++ )
  {
    memcpy( gpForms->index[ii], pForm, iRowSize );
  }

  /* calculate the size of one row of responses */
  iRowSize = 0;
  for( jj = 0; jj < NUMBER_POOL_RESPONSE_TYPES; jj++ )
  {
    iResponseLen[jj] = ( giResponseLen[jj] + 8 ) & ( ~(int)7 );
    iRowSize += iResponseLen[jj];
  }

  iMallocSize = (((char *)&gpResponses->index - (char *)gpResponses) +
                 (((char *)gpResponses->responses - (char *)gpResponses->index)
                  * dwResponseSize * NUMBER_POOL_RESPONSE_TYPES ) +
                 (((char *)&gpResponses->responses[iRowSize * dwResponseSize] -
                   (char *)&gpResponses->responses[0])));
  gpResponses = malloc( iMallocSize );

  for( jj = 0; jj < NUMBER_POOL_RESPONSE_TYPES; jj++ )
  {
    InitializeCriticalSection( &gpResponses->critSec[jj] );
#ifdef FFE_DEBUG
    gpResponses->iMaxIndex[jj] = dwResponseSize - 1;
#endif
    gpResponses->iNextFreeResponse[jj] = 0;
    gpResponses->iFirstResponseIndex[jj] = jj * dwResponseSize;
  }

  pResponse = ((char *)&gpResponses->index[0] +
              (((char *)&gpResponses->responses[0] -
                (char *)&gpResponses->index[0]) *
               NUMBER_POOL_RESPONSE_TYPES * dwResponseSize));
  for( ii = 0; ii < dwResponseSize; ii++ )
  {
    for( jj = 0; jj < NUMBER_POOL_RESPONSE_TYPES; jj++ )
    {
      gpResponses->index[jj*dwResponseSize+ii] = pResponse;
      pResponse += iResponseLen[jj];
    }
  }

  /* load the first row with the templates */
  pResponse = gpResponses->index[0];

  memcpy( pResponse, szDeliveryResponse,
iResponseLen[DELIVERY_RESPONSE] );
  pResponse += iResponseLen[DELIVERY_RESPONSE];

  memcpy( pResponse, szNewOrderResponse,
iResponseLen[NEW_ORDER_RESPONSE] );
  pResponse += iResponseLen[NEW_ORDER_RESPONSE];

  memcpy(pResponse, szOrderStatusResponse,
iResponseLen[ORDER_STATUS_RESPONSE]);
  pResponse += iResponseLen[ORDER_STATUS_RESPONSE];

  memcpy( pResponse, szPaymentResponse,
iResponseLen[PAYMENT_RESPONSE] );
  pResponse += iResponseLen[PAYMENT_RESPONSE];

  memcpy( pResponse, szStockLevelResponse,
iResponseLen[STOCK_LEVEL_RESPONSE] );
  pResponse += iResponseLen[STOCK_LEVEL_RESPONSE];

  /* copy the first row to all the other rows */
  pResponse = gpResponses->index[0];
  for( ii = 1; ii < dwResponseSize; ii++ )
  {
    memcpy( gpResponses->index[ii], pResponse, iRowSize );
  }
}

/* FUNCTION: void DeleteTemplatePool( void )
 *
 * PURPOSE:         This function destroys the array of forms to be
used
 *                  by the threads as they need a form.
 *
 * ARGUMENTS:       none
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */
void
DeleteTemplatePool( void )
{
  int                           jj;

  for( jj = 0; jj < NUMBER_POOL_RESPONSE_TYPES; jj++ )
  {
    DeleteCriticalSection( &gpResponses->critSec[jj] );
  }
  free( gpResponses );

  for( jj = 0; jj < NUMBER_POOL_FORM_TYPES; jj++ )
  {
    DeleteCriticalSection( &gpForms->critSec[jj] );
  }
  free( gpForms );

  DeleteCriticalSection( &gpPanicForms->critSec );
  free( gpPanicForms );
}

/* FUNCTION: void MakeTransactionPool( DWORD dwTransactionPoolSize
)
 *
 * PURPOSE:         This function builds the array of forms to be
used
 *                  by the threads as they need a form.  The forms
are
 *                  reserved and released by each thread as needed.
 *
 * ARGUMENTS:       none
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */
void
MakeTransactionPool( DWORD dwTransactionPoolSize )
{
  int iMaxSize;
  int iSize;
  char *data;
  DWORD ii;

  /**** set up transaction data pool used during async operation
****/
  iMaxSize = 0;
  iMaxSize = MAX(iMaxSize,sizeof(DeliveryData));
  iMaxSize = MAX(iMaxSize,sizeof(NewOrderData));
  iMaxSize = MAX(iMaxSize,sizeof(OrderStatusData));
  iMaxSize = MAX(iMaxSize,sizeof(PaymentData));
  iMaxSize = MAX(iMaxSize,sizeof(StockLevelData));
  iMaxSize = MAX(iMaxSize,sizeof(LoginData));
  iSize = (((char *)&gpTransactionPool->index - (char
*)gpTransactionPool) +
           (((char *)gpTransactionPool->data - (char
*)gpTransactionPool->index)
            * dwTransactionPoolSize ) +
           (sizeof( char ) * iMaxSize * dwTransactionPoolSize ));
  gpTransactionPool = malloc( iSize );
  InitializeCriticalSection( &gpTransactionPool->critSec );
#ifdef FFE_DEBUG
  gpTransactionPool->iMaxIndex = dwTransactionPoolSize - 1;
  gpTransactionPool->iTransactionSize = iMaxSize;
  gpTransactionPool->iHistoryId = 0;
#endif
```

```
  gpTransactionPool->iNextFree = 0;

  /* careful here, the data is not right after index[0] as the
structure */
  /* defines.  We have wedged 'NumUsers + total' indexes in
between. */
  data = ((char *)&gpTransactionPool->index[0] +
           (((char *)&gpTransactionPool->data[0] -
             (char *)&gpTransactionPool->index[0]) *
            dwTransactionPoolSize ));

  for( ii = 0; ii < dwTransactionPoolSize; ii++ ) {
    gpTransactionPool->index[ii] = data;
    data += iMaxSize;
  }
}

/* FUNCTION: void DeleteTransactionPool( void )
 *
 * PURPOSE:         This function destroys the array of transaction
data
 *                  structures used     by the threads as they
process a transaction.
 *
 * ARGUMENTS:       none
 *
 * RETURNS:         none
 *
 * COMMENTS:        none
 */
void
DeleteTransactionPool( void )
{
  DeleteCriticalSection( &gpTransactionPool->critSec );
  free( gpTransactionPool );
}

/* FUNCTION: void BeginCmd( EXTENSION_CONTROL_BLOCK *pECB )
 *
 * PURPOSE:         This routine is executed in response to the
browser query
 *                  'CMD=Begin&Server=??????'.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                            unique
to this connection.
 *                                                            at
login.
 * RETURNS:         None
 *
 * COMMENTS:        Specification of a server machine is required.
 */

void
BeginCmd( EXTENSION_CONTROL_BLOCK *pECB )
{
  SendWelcomeForm(pECB);
}

/* FUNCTION: void CheckpointCmd(EXTENSION_CONTROL_BLOCK *pECB,
 *
 * PURPOSE:         This function causes a checkpoint to occur in
the database
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                            unique
to this connection.
 *
 * RETURNS:         None
 *
 * COMMENTS:        none
 */

void
CheckpointCmd(EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id )
{
  int                         retcode;
  CheckpointData    checkpoint;

  checkpoint.w_id = w_id;
  checkpoint.ld_id = ld_id;
  checkpoint.pCC = pECB;

  retcode = TPCCCheckpoint( &checkpoint );
  if ( ERR_SUCCESS == retcode ) {
    SendMainMenuForm(pECB, w_id, ld_id,
                     "Checkpoint issued (non-blocking),
completed (blocking).");
  }
  else {
    SendErrorResponse( pECB, retcode, ERR_TYPE_WEBDLL,
                       NULL, w_id, ld_id, NULL );
  }
}

/* FUNCTION: void CheckpointStartupCmd(EXTENSION_CONTROL_BLOCK
*pECB,
 *
 * PURPOSE:         This function causes initialization of the
checkpoint command.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                            unique
to this connection.
 *
 * RETURNS:         None
 *
 * COMMENTS:        none
 */

void
CheckpointStartupCmd(EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id )
{
  int                         retcode;

  retcode = CKPTStartup( );
  if ( ERR_SUCCESS == retcode ) {
    SendMainMenuForm(pECB, w_id, ld_id, "Checkpoint Startup
Succeded.");
  }
  else {
    SendErrorResponse( pECB, retcode, ERR_TYPE_WEBDLL,
                       NULL, w_id, ld_id, NULL );
  }
}

/* FUNCTION: void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB)
 *
 * PURPOSE:         This resets all terminals and resets the log
file.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                            unique
to this connection.
 *
 * RETURNS:                    None
 *
 * COMMENTS:        This function resets the connection information
for the
 *                  dll.  Any "users" with current connections will
be given
 *                  an error message on their next transaction.
 */

void
ClearCmd(EXTENSION_CONTROL_BLOCK *pECB)
{
  if ( bLog )
  {
    TPCCCloseLog( );
    TPCCOpenLog( );
  }

  SendWelcomeForm(pECB);
}

/* FUNCTION: void ExitCmd(EXTENSION_CONTROL_BLOCK *pECB,
 *
 * PURPOSE:         This function deallocates the terminal
associated with
 *                  the browser and presents the login screen.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                            unique
to this connection.
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

void
ExitCmd( EXTENSION_CONTROL_BLOCK *pECB )
{
  TPCCDisconnect( pECB );

  SendWelcomeForm( pECB );
}

/* FUNCTION: void MenuCmd( EXTENSION_CONTROL_BLOCK *pECB,
 *
 * PURPOSE:         This function displays the main menu.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                            unique
to this connection.
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

void
MenuCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id )
{
  SendMainMenuForm(pECB, w_id, ld_id, NULL);
}

/* FUNCTION: void SubmitCmd( EXTENSION_CONTROL_BLOCK *pECB )
 *
 * PURPOSE:         This function assigns a unique terminal id to
the calling
 *                  browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                            unique
to this connection.
 * RETURNS:         None
```

```
 *
 * COMMENTS:        A terminal id can be allocated but still be
invalid if the
 *                  requested warehouse number is outside the range
specified
 *                  in the registry. This then will force the
client id
 *                  to be invalid and an error message sent to the
users browser.
 */

void
SubmitCmd( EXTENSION_CONTROL_BLOCK *pECB, int *w_id, int *ld_id )
{
  int     iStatus;
  LoginData login;
  char    *ptr;

  if ( !GetCharKeyValuePtr( pECB->lpszQueryString, '4', &ptr ) ||
       ( 0 == ( *w_id = atoi( ptr ))) ||
       ( *w_id < 0 ))
  {
    SendErrorResponse( pECB, ERR_W_ID_INVALID, ERR_TYPE_WEBDLL,
                       NULL, *w_id, -1, NULL );
    goto SubmitError;
  }

  if ( !GetCharKeyValuePtr( pECB->lpszQueryString, '5', &ptr ) ||
       ( 0 == ( *ld_id = atoi( ptr ))) ||
       ( *ld_id > 10 ) ||
       ( *ld_id < 0 ))
  {
    SendErrorResponse( pECB, ERR_D_ID_INVALID, ERR_TYPE_WEBDLL,
                       NULL, *w_id, *ld_id, NULL );
    goto SubmitError;
  }

  login.w_id = *w_id;
  login.ld_id = *ld_id;
  login.pCC = pECB;
  strcpy( login.szServer, gszServer );
  strcpy( login.szDatabase, gszDatabase );
  strcpy( login.szUser, gszUser );
  strcpy( login.szPassword, gszPassword );
  sprintf( login.szApplication, "TPCC" );
  iStatus = TPCCConnect( &login );
  if( ERR_DB_SUCCESS != iStatus )
  {
    SendErrorResponse( pECB, iStatus, ERR_TYPE_WEBDLL,
                       NULL, *w_id, *ld_id, NULL );
    goto SubmitError;
  }

  SendMainMenuForm(pECB, *w_id, *ld_id, NULL);
  return;

SubmitError:
  return;

}

/* FUNCTION: void MemoryCheckCmd( EXTENSION_CONTROL_BLOCK *pECB,
 *
 * PURPOSE:         This function displays the main menu.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK       *pECB     IIS
context structure pointer
 *                                                          unique
to this connection.
 * RETURNS:         None
 *
 * COMMENTS:        None
 *
 */

void
MemoryCheckCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id
)
{
  _ASSERTE( _CrtCheckMemory( ));

  SendErrorResponse( pECB, ERR_SUCCESS, ERR_TYPE_WEBDLL, NULL,
                     w_id, ld_id, NULL );
}

/* FUNCTION: BOOL GetKeyValuePtr( char *szIPtr, char *szKey, char
**pszOPtr )
 *
 * PURPOSE:         This function searches the input string for the
key
 *                  specified.  If found, it returns a pointer to
the value.
 *
 * ARGUMENTS:       char    *szIPtr               pointer to string
to check.
 *                  char      *szKey              pointer to key to
find.
 *                  char      **pszOPtr pointer to value.
 *
 * RETURNS:         BOOL      FALSE               if key is not
found.
 *                            TRUE                if key is found.
 *
 * COMMENTS:        A side affect of this routine is that the
output string
 *                  pointer will either point at the start of the
value being
 *                  searched or at the *start* point where ptr
originated.
 */
BOOL
GetKeyValuePtr( char *szIPtr, char *szKey, char **pszOPtr )
{
  char *szPtr1, *szPtr2;

  *pszOPtr = szIPtr;
  while (*szIPtr)
  {
    szPtr1 = szIPtr;
    szPtr2 = szKey;

    while ( *szPtr1 && *szPtr2 && 0 == ( *szPtr1 - *szPtr2 ))
      szPtr1++, szPtr2++;

    if ( '=' == *szPtr1 && '\0' == *szPtr2 )
    {
      *pszOPtr = ++szPtr1;
      return TRUE;
    }

    szIPtr++;
  }

  return FALSE;
}

/* FUNCTION: BOOL GetKeyValueCharPtr( char *szIPtr, char cKey, char
**pszOPtr )
 *
 * PURPOSE:         This function searches the input string for the
single char key
 *                  specified.  If found, it returns a pointer to
the value.
 *
 * ARGUMENTS:       char    *szIPtr               pointer to string
to check.
 *                  char      cKey                pointer to key to
find.
 *                  char      **pszOPtr pointer to value.
 *
 * RETURNS:         BOOL      FALSE               if key is not
found.
 *                            TRUE                if key is found.
 *
 * COMMENTS:        A side affect of this routine is that the
output string
 *                  pointer will either point at the start of the
value being
 *                  searched or at the *start* point where ptr
originated.
 */
BOOL
GetCharKeyValuePtr( char *szIPtr, char cKey, char **pszOPtr )
{
  BOOL    bGotStart;

  *pszOPtr = szIPtr;
  bGotStart = FALSE;
  while( *szIPtr )
  {
    if( cKey == *szIPtr && '=' == *++szIPtr )
    {
      *pszOPtr = ++szIPtr;
      return TRUE;
    }
    while( *szIPtr )
    {
      if( '&' == *szIPtr )
      {
          szIPtr++;
          break;
      }
      szIPtr++;
    }
  }

  return FALSE;
}

/* FUNCTION: BOOL GetNumeric(char *ptr, int *iValue)
 *
 * PURPOSE:         This function converts the string value to
integer, and
 *                  determines if the string is terminated
properly.  If it
 *                  contains non-numeric characters or if any
characters
 *                  other than '&' or '\0' terminate the integer
portion
 *                  of the string, this function fails.
 *
 * ARGUMENTS:       char    *ptr        pointer to string to check.
 *
 * RETURNS:         BOOL      FALSE     if string is not all
numeric and properly
 *                                      terminated.
 *                            TRUE      if string contains only
numeric characters
 *                                      i.e. '0' - '9' and is
properly terminated.
 *
 * COMMENTS:        None
 *
 */
BOOL
```

```
GetNumeric(char *ptr, int *iValue)
{
  int c;             /* current char */
  int total;         /* current total */
  BOOL bGotSomething = FALSE;

  c = (int)(unsigned char)*ptr++;

  total = 0;

  while ((c >= '0') && (c <= '9'))
  {
    total = 10 * total + (c - '0');     /* accumulate digit */
    c = (int)(unsigned char)*ptr++;    /* get next char */
    bGotSomething = TRUE;
  }
  if(('\0' == c) || ('&' == c) && bGotSomething)
  {
    *iValue = total;
    return (TRUE);   /* return result */
  }
  else
  {
    *iValue = 0;
    return(FALSE);
  }
}

/* FUNCTION: BOOL GetWDID(char *ptr, int *lw_id, int *ld_id, char
**optr)
 *
 * PURPOSE:         This function converts the string value to a
pair of integers
 *                  where the ascii numeric field represents an
encoded warehouse
 *                  and district id.  The least significant digit
is one less than
 *                  the actual local district id, and the remaining
high order
 *                  digits are 10 times the actual local warehouse
id.
 *
 * ARGUMENTS:       char    *ptr        pointer to string to check.
 *
 * RETURNS:         BOOL      FALSE     if string is not all
numeric and properly
 *                                      terminated.
 *                            TRUE      if string contains only
numeric characters
 *                                      i.e. '0' - '9' and is
properly terminated.
 *
 * COMMENTS:        A side affect of this routine is that the
output string
 *                  pointer will either point at the end of the
values being
 *                  searched or at the *start* point where ptr
originated.
 *
 */
BOOL
GetWDID(char *ptr, int *lw_id, int *ld_id, char **optr)
{
  int c;              /* current char */
  int pc;       /* previous character */
  int total;         /* current total */
  BOOL bGotSomething = FALSE;

  *lw_id = 0;
  *ld_id = 0;
  total = 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');     /* accumulate digit */
    pc = c;
    c = (int)(unsigned char)*ptr++;    /* get next char */
    bGotSomething = TRUE;
  }
  if(('\0' == c) || ('&' == c) && bGotSomething)
  {
    *lw_id = total;
    *ld_id = (pc - '0') + 1;
    *optr = ptr;
    return TRUE;   /* return result */
  }
  else
    return FALSE;
}

/* FUNCTION: BOOL GetKeyValueString(char *szIPtr, char *szKey,
 *                                          char *szValue, int
iSize)
 *
 * PURPOSE:         This function searches for the key specified
and returns
 *                  the string value associated with it.
 *
 * ARGUMENTS:       char      *szIPtr                       string
to search
 *                  char      *szKey                        key to
search for
 *                  char      *szValue            location to store
value
 *                  int       iSize                         size of
output array.
 *
 * RETURNS:         BOOL      FALSE                         key not
found
 *                            TRUE                          key
found, value stored
 *
 *
 * COMMENTS:        http keys are formatted either KEY=value& or
KEY=value\0.
 *                  This DLL formats TPC-C input fields in such a
manner that
 *                  the keys can be extracted in the above manner.
 */

BOOL
GetKeyValueString(char *szIPtr, char *szKey,
                      char *szValue, int iSize)
{
  char *ptr;

  if( !GetKeyValuePtr( szIPtr, szKey, &ptr ))
    return FALSE;

  /* force zero termination of output string */
  iSize--;

  while( '\0' != *ptr && '&' != *ptr && iSize)
  {
    *szValue++ = *ptr++;
    iSize--;
  }
  *szValue = 0;
  return TRUE;
}

/* FUNCTION: void CheckMemory(void *param)
 *
 * PURPOSE:         This function loops calling _CrtCheckMemory()
 *
 * ARGUMENTS:
 *                  void      *param                        not
used
 *
 * RETURNS:         nothing
 *
 * COMMENTS:
 */

#ifdef FFE_DEBUG

unsigned __stdcall
CheckMemory(void *param)
{
          while (TRUE)
          {
                    _ASSERTE(_CrtCheckMemory());
                    Sleep(1000);
          }

          return 0;
}

#endif


--------------------------
web_ui.h
--------------------------
#ifndef WEB_UI_H
#define WEB_UI_H
/*+**********************************************************************
**********
 *
*
 *  COPYRIGHT (c) 1997 BY
*
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*
 *  ALL RIGHTS RESERVED.
*
 *
*
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
COPIED   *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND
WITH THE   *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY
OTHER   *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
TO ANY   *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS
HEREBY   *
 *  TRANSFERRED.
*
 *
*
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT
NOTICE   *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL
EQUIPMENT   *
 *  CORPORATION.
*
```

```
 *
*
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY
OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
 *
*
 *
*

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

/*+
 * Abstract: This is the header file for web_ui.c.  it contains the
 *        function prototypes for the routines that are called
outside web_ui.c
 *
 * Author: A Bradley
 * Creation Date: May 1997
 *
 * Modified history:
 *
 *
 */

/* function prototypes */
BOOL GetNumeric(char *ptr, int *iValue);
BOOL GetValuePtr(char *pProcessedQuery[], int iIndex, char
**pValue);

/* define indexes for parsing the query string */
/* for the payment, orderstatus and new order txns */
#define DID 0
#define CID DID+1
/* more for the order status txn */
#define CLT_O CID+1
#define MAXORDERSTATUSVALS CLT_O + 1
/* for the stocklevel txn */
#define TT  0
#define MAXSTOCKLEVELVALS TT + 1
/* for the delivery txn */
#define QUEUETIME 0
#define OCD 1
#define MAXDELIVERYVALS OCD + 1
/* more for the payment txn */
#define CWI   CID + 1
#define CDI   CWI + 1
#define CLT_P CDI + 1
#define HAM   CLT_P + 1
#define MAXPAYMENTVALS HAM + 1
/* more for the neworder txn */
#define SP00  CID + 1
#define IID00 SP00 + 1
#define QTY00 IID00 + 1
#define SP01  QTY00 + 1
#define IID01 SP01 + 1
#define QTY01 IID01 + 1
#define SP02  QTY01 + 1
#define IID02 SP02 + 1
#define QTY02 IID02 + 1
#define SP03  QTY02 + 1
#define IID03 SP03 + 1
#define QTY03 IID03 + 1
#define SP04  QTY03 + 1
#define IID04 SP04 + 1
#define QTY04 IID04 + 1
#define SP05  QTY04 + 1
#define IID05 SP05 + 1
#define QTY05 IID05 + 1
#define SP06  QTY05 +1
#define IID06 SP06 + 1
#define QTY06 IID06 + 1
#define SP07  QTY06 + 1
#define IID07 SP07 + 1
#define QTY07 IID07 + 1
#define SP08  QTY07 + 1
#define IID08 SP08 + 1
#define QTY08 IID08 + 1
#define SP09  QTY08 + 1
#define IID09 SP09 + 1
#define QTY09 IID09 + 1
#define SP10  QTY09 + 1
#define IID10 SP10 + 1
#define QTY10 IID10 + 1
#define SP11  QTY10 + 1
#define IID11 SP11 + 1
#define QTY11 IID11 + 1
#define SP12  QTY11 + 1
#define IID12 SP12 + 1
#define QTY12 IID12 + 1
#define SP13  QTY12 + 1
#define IID13 SP13 + 1
#define QTY13 IID13 + 1
#define SP14  QTY13 + 1
#define IID14 SP14 + 1
#define QTY14 IID14 + 1
#define MAXNEWORDERVALS QTY14 + 1

#if 0
#define PARSE_QUERY_STRING(pQueryString,varMax,charTable,valTable)\
          {\
            int ii;\
            char *ptr, *tmpPtr;\
            ptr = pQueryString;\
            for (ii=0; ii < varMax; ii++)\
            {\
              if ( !(tmpPtr=strstr(ptr, stringTable[ii])) )\
                valTable[ii] = NULL;\
              else\
              {\
                ptr = tmpPtr;\
                if ( !(ptr=strchr(ptr, '=')) )\
                    valTable[ii] = NULL;\
                else\
                    valTable[ii] = ++ptr;\
              }\
            }\
          }
#else
#define PARSE_QUERY_STRING(pQueryString,varMax,charTable,valTable)\
{\
  int ii;\
  char *ptr;\
  int iKey;\
  ptr = pQueryString;\
  for( ii=0; ii<varMax; ii++ ) {\
    iKey = charTable[ii];\
    valTable[ii] = NULL;\
    if( iKey == *ptr && '=' == *++ptr ) {\
      valTable[ii] = ++ptr;\
    }\
    while( *ptr ) {\
      if( '&' == *ptr ) {\
          ptr++;\
          break;\
      }\
      ptr++;\
    }\
  }\
}
#endif

typedef struct _FORMINDEXES
{
  int iStartIndex;    // index into the form char array for values
  int iLen;           // length of the current value field
} FORM_INDEXES;

GLOBAL(FORM_INDEXES deliveryFormIndexesI[4], { 0 });
GLOBAL(FORM_INDEXES deliveryFormIndexesP[33], { 0 });
GLOBAL(FORM_INDEXES newOrderFormIndexes[4], { 0 });
GLOBAL(FORM_INDEXES newOrderResponseIndexes[136], { 0 });
GLOBAL(FORM_INDEXES orderStatusFormIndexes[4], { 0 });
GLOBAL(FORM_INDEXES orderStatusResponseIndexes[88], { 0 });
GLOBAL(FORM_INDEXES paymentFormIndexes[4], { 0 });
GLOBAL(FORM_INDEXES paymentResponseIndexes[38], { 0 });
GLOBAL(FORM_INDEXES stockLevelFormIndexes[5], { 0 });
GLOBAL(FORM_INDEXES stockLevelResponseIndexes[7], { 0 });

#ifdef WEB_UI_C
char deliveryStrs[] = {'6', '7'};
char newOrderStrs[] = {
'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'};
char orderStatusStrs[] = {'8', '9', 'Y'};
char paymentStrs[] = {'8', '9', 'Z', 'v', 'Y', 'w'};
char stockLevelStrs[] = {'x'};
#else
extern char deliveryStrs[];
extern char newOrderStrs[];
extern char orderStatusStrs[];
extern char paymentStrs[];
extern char stockLevelStrs[];
#endif /* WEB_UI_C */
GLOBAL(char szModName[FILENAMESIZE], { 0 });
GLOBAL(DWORD gdwPoolThreadLimit, 0);
#endif /* WEB_UI_H */




-------------------------
initnew.sql
-------------------------
-- The initnew package for storing variables used in the
-- New Order anonymous block

CREATE OR REPLACE PACKAGE initnew
AS
 TYPE intarray IS TABLE OF INTEGER index by binary_integer;
 TYPE distarray IS TABLE OF VARCHAR(24) index by binary_integer;
 nulldate       DATE;
 s_dist                      distarray;
 idx1arr            intarray;
 s_remote           intarray;
 PROCEDURE new_init(idxarr intarray);
END initnew;
/
show errors;
```

```
CREATE OR REPLACE PACKAGE BODY initnew AS
  PROCEDURE new_init (idxarr  intarray)
  IS
  BEGIN
        -- initialize null date
   nulldate := TO_DATE('01-01-1811', 'MM-DD-YYYY');
    idx1arr := idxarr;
  END new_init;
END initnew;
/
show errors
exit


-------------------------
initpay.sql
-------------------------
CREATE OR REPLACE PACKAGE initpay
AS
 TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
 row_id                  rowidarray;
 cust_rowid              ROWID;
 dist_name               VARCHAR2(11);
 ware_name               VARCHAR2(11);
 c_num                   BINARY_INTEGER;
 PROCEDURE pay_init;
END initpay;
/
CREATE OR REPLACE PACKAGE BODY initpay AS
  PROCEDURE pay_init IS
  BEGIN
    NULL;
  END pay_init;
END initpay;
/

exit;


-------------------------
new.sql
-------------------------
  DECLARE   /* new order */
      not_serializable         EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
      deadlock                 EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old         EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
   BEGIN
      LOOP BEGIN
         SELECT c_discount, c_last, c_credit
           INTO :c_discount, :c_last, :c_credit
           FROM cust
          WHERE c_id = :c_id
            AND c_d_id = :d_id
            AND c_w_id = :w_id;

        UPDATE wh_dist SET d_next_o_id = d_next_o_id + 1,
d_tax=d_tax+0
          WHERE  d_id = :d_id
            AND  w_id = :w_id
      RETURNING d_tax, d_next_o_id-1, w_tax
           INTO :d_tax, :o_id, :w_tax;

        INSERT INTO nord  (no_o_id, no_d_id, no_w_id)
             VALUES (:o_id , :d_id, :w_id);
        INSERT INTO ordr (o_id, o_w_id, o_d_id, o_c_id,
o_carrier_id,
                          o_ol_cnt, o_all_local,o_entry_d)
             VALUES (:o_id, :w_id, :d_id, :c_id, 11,
                     :o_ol_cnt, :o_all_local, :cr_date);
        EXIT;

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



-------------------------
pay.sql
-------------------------
CREATE OR REPLACE PACKAGE pay
AS
 TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
 row_id                  rowidarray;
 cust_rowid              ROWID;
 dist_name               VARCHAR2(11);
 ware_name               VARCHAR2(11);
 c_num                   BINARY_INTEGER;
 PROCEDURE pay_init;
END pay;
/
CREATE OR REPLACE PACKAGE BODY pay AS
  PROCEDURE pay_init IS
  BEGIN
    NULL;
  END pay_init;
END pay;
/


-------------------------
paynz.sql
-------------------------
DECLARE /* paynz */
--      cust_rowid              ROWID;
--      dist_name               VARCHAR2(11);
--      ware_name               VARCHAR2(11);
      not_serializable        EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
      deadlock                EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old        EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
   BEGIN
      LOOP BEGIN
            UPDATE ware
               SET w_ytd = w_ytd + :h_amount
             WHERE w_id = :w_id
         RETURNING w_name, w_street_1, w_street_2, w_city, w_state,
w_zip
              INTO initpay.ware_name, :w_street_1, :w_street_2,
:w_city,
                   :w_state, :w_zip;

           UPDATE  cust
              SET  c_balance = c_balance - :h_amount,
                   c_ytd_payment = c_ytd_payment + :h_amount,
                   c_payment_cnt = c_payment_cnt+1
            WHERE  c_id = :c_id AND c_d_id = :c_d_id AND
                   c_w_id = :c_w_id
         RETURNING rowid, c_first, c_middle, c_last, c_street_1,
                   c_street_2, c_city, c_state, c_zip, c_phone,
                   c_since, c_credit, c_credit_lim,
                   c_discount, c_balance
              INTO initpay.cust_rowid,:c_first, :c_middle, :c_last,
:c_street_1,
                   :c_street_2, :c_city, :c_state, :c_zip,
:c_phone,
                   :c_since, :c_credit, :c_credit_lim,
                   :c_discount, :c_balance;
           IF SQL%NOTFOUND THEN
             raise NO_DATA_FOUND;
           END IF;
--                 insert into dummy values
(rowidtochar(initpay.cust_rowid));

--         :c_data := ' ';

         IF :c_credit = 'BC' THEN
            UPDATE cust
                SET c_data= substr ((to_char (:c_id) || ' ' ||
                                     to_char (:c_d_id) || ' ' ||
                                     to_char (:c_w_id) || ' ' ||
                                     to_char (:d_id) || ' ' ||
                                     to_char (:w_id) || ' ' ||
                                     to_char (:h_amount, '9999.99')
|| ' | ')
                                     || c_data, 1, 500)
             WHERE rowid = initpay.cust_rowid
         RETURNING substr(c_data,1, 200)
              INTO :c_data;

         END IF;

            UPDATE dist
               SET d_ytd = d_ytd + :h_amount
             WHERE d_id = :d_id
               AND d_w_id = :w_id
         RETURNING d_name, d_street_1, d_street_2, d_city,d_state,
d_zip
              INTO
initpay.dist_name,:d_street_1,:d_street_2,:d_city,:d_state,
                   :d_zip;
          IF SQL%NOTFOUND THEN
            raise NO_DATA_FOUND;
          END IF;


         INSERT INTO hist  (h_c_id, h_c_d_id, h_c_w_id, h_d_id,
h_w_id,
                            h_amount, h_date, h_data)
         VALUES
            (:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
             :cr_date, initpay.ware_name || '    ' ||
initpay.dist_name);
--         COMMIT;
--         :h_date := to_char (:cr_date, 'DD-MM-YYYY.HH24:MI:SS');
         EXIT;

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

      END LOOP;
   END;


-------------------------
payz.sql
-------------------------
DECLARE /* payz */
      not_serializable         EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
```

```
      deadlock                EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old        EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
   BEGIN
      LOOP BEGIN
         UPDATE ware
            SET w_ytd = w_ytd+:h_amount
            WHERE w_id = :w_id
            RETURNING w_name,
                      w_street_1, w_street_2, w_city, w_state,
w_zip
             INTO initpay.ware_name,
                  :w_street_1, :w_street_2, :w_city, :w_state,
:w_zip;


         SELECT /*+ no_index (icust1) index(cust icust2) */ rowid
         BULK COLLECT INTO initpay.row_id
         FROM cust
         WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last =
:c_last
         ORDER BY c_last, c_d_id, c_w_id, c_first;

       initpay.c_num := sql%rowcount;
       initpay.cust_rowid := initpay.row_id((initpay.c_num+1) / 2);

        UPDATE cust
          SET c_balance = c_balance - :h_amount,
              c_ytd_payment = c_ytd_payment+ :h_amount,
              c_payment_cnt = c_payment_cnt+1
        WHERE rowid = initpay.cust_rowid
        RETURNING
                 c_id, c_first, c_middle, c_last, c_street_1,
c_street_2,
                 c_city, c_state, c_zip, c_phone,
                 c_since, c_credit, c_credit_lim,
                 c_discount, c_balance
             INTO :c_id, :c_first, :c_middle, :c_last,
                  :c_street_1, :c_street_2, :c_city, :c_state,
                  :c_zip, :c_phone, :c_since, :c_credit,
                  :c_credit_lim, :c_discount, :c_balance;

         :c_data := ' ';
         IF :c_credit = 'BC' THEN
            UPDATE cust
              SET c_data = substr ((to_char (:c_id) || ' ' ||
                                    to_char (:c_d_id) || ' ' ||
                                    to_char (:c_w_id) || ' ' ||
                                    to_char (:d_id) || ' ' ||
                                    to_char (:w_id) || ' ' ||
                                    to_char (:h_amount/100,
'9999.99') || ' | ')
                                    || c_data, 1, 500)
              WHERE rowid = initpay.cust_rowid
              RETURNING substr(c_data,1, 200)
              INTO :c_data;

         END IF;

         UPDATE dist
            SET d_ytd = d_ytd+:h_amount
            WHERE d_id = :d_id
              AND d_w_id = :w_id
            RETURNING  d_name, d_street_1, d_street_2, d_city,
                  d_state, d_zip
            INTO initpay.dist_name, :d_street_1, :d_street_2,
:d_city,
                 :d_state, :d_zip;

            IF  SQL%NOTFOUND
              THEN
                    raise NO_DATA_FOUND;
            END IF;

         INSERT INTO hist (h_c_id, h_c_d_id, h_c_w_id, h_d_id,
h_w_id,
                           h_amount, h_date, h_data)
             VALUES (:c_id, :c_d_id, :c_w_id, :d_id, :w_id,
:h_amount,
              :cr_date, initpay.ware_name || '    ' ||
initpay.dist_name);

         EXIT;

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

      END LOOP;
   END;


--------------------------
tkcvpnew.sql
--------------------------

-- New Order Anonymous block

  DECLARE
      idx                     BINARY_INTEGER;
      dummy_local             BINARY_INTEGER;
      not_serializable        EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
      deadlock                EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old        EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
    PROCEDURE u1 IS
    BEGIN
          FORALL idx IN 1 .. :o_ol_cnt
           UPDATE stock_item
           SET s_order_cnt = s_order_cnt + 1,
           s_ytd = s_ytd + :ol_quantity(idx),
           s_remote_cnt = s_remote_cnt + :s_remote(idx),
           s_quantity = s_quantity - :ol_quantity(idx) +
                        DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
           WHERE i_id = :ol_i_id(idx)
           AND s_w_id = :ol_supply_w_id(idx)
           RETURNING i_price, i_name, s_quantity, s_dist_01,
             DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                        DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
            BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
:brand_generic;


    END u1;

    PROCEDURE u2 IS
    BEGIN
          FORALL idx IN 1 .. :o_ol_cnt
           UPDATE stock_item
           SET s_order_cnt = s_order_cnt + 1,
           s_ytd = s_ytd + :ol_quantity(idx),
           s_remote_cnt = s_remote_cnt + :s_remote(idx),
           s_quantity = s_quantity - :ol_quantity(idx) +
                        DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
           WHERE i_id = :ol_i_id(idx)
           AND s_w_id = :ol_supply_w_id(idx)
           RETURNING i_price, i_name, s_quantity, s_dist_02,
             DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                        DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
            BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                           :brand_generic;
    END u2;

    PROCEDURE u3 IS
    BEGIN
          FORALL idx IN 1 .. :o_ol_cnt
           UPDATE stock_item
           SET s_order_cnt = s_order_cnt + 1,
           s_ytd = s_ytd + :ol_quantity(idx),
           s_remote_cnt = s_remote_cnt + :s_remote(idx),
           s_quantity = s_quantity - :ol_quantity(idx) +
                        DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
           WHERE i_id = :ol_i_id(idx)
           AND s_w_id = :ol_supply_w_id(idx)
           RETURNING i_price, i_name, s_quantity, s_dist_03,
             DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                        DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
            BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                           :brand_generic;
    END u3;

    PROCEDURE u4 IS
    BEGIN
          FORALL idx IN 1 .. :o_ol_cnt
           UPDATE stock_item
           SET s_order_cnt = s_order_cnt + 1,
           s_ytd = s_ytd + :ol_quantity(idx),
           s_remote_cnt = s_remote_cnt + :s_remote(idx),
           s_quantity = s_quantity - :ol_quantity(idx) +
                        DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
           WHERE i_id = :ol_i_id(idx)
           AND s_w_id = :ol_supply_w_id(idx)
           RETURNING i_price, i_name, s_quantity, s_dist_04,
             DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                        DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
            BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                           :brand_generic;
    END u4;

    PROCEDURE u5 IS
    BEGIN
          FORALL idx IN 1 .. :o_ol_cnt
           UPDATE stock_item
           SET s_order_cnt = s_order_cnt + 1,
           s_ytd = s_ytd + :ol_quantity(idx),
           s_remote_cnt = s_remote_cnt + :s_remote(idx),
           s_quantity = s_quantity - :ol_quantity(idx) +
                        DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
           WHERE i_id = :ol_i_id(idx)
           AND s_w_id = :ol_supply_w_id(idx)
           RETURNING i_price, i_name, s_quantity, s_dist_05,
             DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                        DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
            BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                           :brand_generic;
```

```
   END u5;

   PROCEDURE u6 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_06,
            DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
           BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                             :brand_generic;
   END u6;

   PROCEDURE u7 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_07,
            DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
           BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                             :brand_generic;
   END u7;

   PROCEDURE u8 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_08,
            DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
           BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                             :brand_generic;
   END u8;

   PROCEDURE u9 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_09,
            DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
           BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                             :brand_generic;
   END u9;

   PROCEDURE u10 IS
   BEGIN
         FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = s_quantity - :ol_quantity(idx) +
                       DECODE(sign(s_quantity - :ol_quantity(idx) -
10),-1,91,0)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_10,
            DECODE (instr(i_data,'ORIGINAL'), 0, 'G',
                       DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
'B'))
           BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                             :brand_generic;
   END u10;

   PROCEDURE fix_items IS
      rows_lost                      BINARY_INTEGER;
      max_index                      BINARY_INTEGER;
      temp_index                     BINARY_INTEGER;
   BEGIN
-- gotta shift price, name, s_quantity, brand_generic, s_dist,
ol_amount
         idx := 1;
-- found 0 bad rows
        rows_lost := 0;
-- so many rows in out array to begin with
          max_index := sql%rowcount;

        WHILE (max_index != :o_ol_cnt) LOOP

-- find item where item ids dont match
            WHILE (idx <= sql%rowcount AND
                          sql%bulk_rowcount(idx + rows_lost) = 1)
           LOOP
               idx := idx + 1;
           END LOOP;

-- shift the items please
           temp_index := max_index;
           WHILE (temp_index >= idx + rows_lost) LOOP
                :i_price(temp_index + 1)       :=
:i_price(temp_index);
              :i_name(temp_index + 1)        :=
:i_name(temp_index);
                :s_quantity(temp_index + 1)    :=
:s_quantity(temp_index);
             initnew.s_dist(temp_index + 1) :=
initnew.s_dist(temp_index);
                :brand_generic(temp_index + 1) :=
:brand_generic(temp_index);
              temp_index := temp_index - 1;
           END LOOP;

--  values for the non-existent items if not at end
            IF (idx + rows_lost <= :o_ol_cnt) THEN
               :i_price(idx + rows_lost)     :=   0;
               :i_name(idx + rows_lost)      :=   NULL;
             :s_quantity(idx + rows_lost)  :=   0;
             initnew.s_dist(idx + rows_lost) := NULL;
             :brand_generic(idx + rows_lost) := NULL;

--  one more bad row
             rows_lost := rows_lost + 1;
             max_index := max_index + 1;
              END IF;

         END LOOP;
   END fix_items;

   BEGIN
      LOOP BEGIN
         UPDATE dist SET d_next_o_id = d_next_o_id + 1
          WHERE d_id = :d_id AND  d_w_id = :w_id
          RETURNING d_tax, d_next_o_id-1
          INTO :d_tax, :o_id;

         SELECT c_discount, c_last, c_credit, w_tax
          INTO :c_discount, :c_last, :c_credit , :w_tax
          FROM cust , ware
          WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id
          AND  w_id = :w_id;


         INSERT INTO nord (no_o_id, no_d_id, no_w_id)
          VALUES (:o_id, :d_id, :w_id);
          INSERT INTO ordr  (o_id,o_d_id, o_w_id, o_c_id,
o_entry_d,
                             o_carrier_id, o_ol_cnt, o_all_local)
          VALUES (:o_id, :d_id, :w_id, :c_id,
                  :cr_date, 11, :o_ol_cnt, :o_all_local);

         dummy_local :=  :d_id;

         IF (dummy_local = 1) THEN u1; END IF;

         IF (dummy_local = 2) THEN u2; END IF;

         IF (dummy_local = 3) THEN u3; END IF;

         IF (dummy_local = 4) THEN u4; END IF;

         IF (dummy_local = 5) THEN u5; END IF;

         IF (dummy_local = 6) THEN u6; END IF;

         IF (dummy_local = 7) THEN u7; END IF;

         IF (dummy_local = 8) THEN u8; END IF;

         IF (dummy_local = 9) THEN u9; END IF;

         IF (dummy_local = 10) THEN u10; END IF;

-- cache the no of rows processed
         dummy_local := sql%rowcount;


-- fix the rows if necessary
         IF (dummy_local != :o_ol_cnt )  THEN fix_items; END IF;

-- calculate ol_amount

        FOR idx IN 1 ..:o_ol_cnt LOOP
         :ol_amount(idx):=:ol_quantity(idx)*:i_price(idx);
```

```
        END LOOP;

        FORALL idx IN 1..:o_ol_cnt
         INSERT INTO ordl
            (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_delivery_d,
ol_i_id,
                  ol_supply_w_id,
ol_quantity,ol_amount,ol_dist_info)
         VALUES (:o_id, :d_id, :w_id, initnew.idx1arr(idx),
initnew.nulldate,
                 :ol_i_id(idx), :ol_supply_w_id(idx),
                   :ol_quantity(idx), :ol_amount(idx),
initnew.s_dist(idx));

          IF (dummy_local != :o_ol_cnt) THEN
           :o_ol_cnt := dummy_local;
           ROLLBACK;
        END IF;

        EXIT;

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



-------------------------
tkvcbnew.sql
-------------------------
  DECLARE   /* new order */
      not_serializable         EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
      deadlock                 EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old         EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
   BEGIN
      LOOP BEGIN
         SELECT c_discount, c_last, c_credit
           INTO :c_discount, :c_last, :c_credit
           FROM cust
          WHERE c_id = :c_id
            AND c_d_id = :d_id
            AND c_w_id = :w_id;

        UPDATE wh_dist SET d_next_o_id = d_next_o_id + 1,
d_tax=d_tax+0
          WHERE  d_id = :d_id
            AND  w_id = :w_id
          RETURNING d_tax, d_next_o_id-1, w_tax
           INTO :d_tax, :o_id, :w_tax;

        INSERT INTO nord  (no_o_id, no_d_id, no_w_id)
             VALUES (:o_id , :d_id, :w_id);
        INSERT INTO ordr (o_id, o_w_id, o_d_id, o_c_id,
o_carrier_id,
                          o_ol_cnt, o_all_local,o_entry_d)
             VALUES (:o_id, :w_id, :d_id, :c_id, 11,
                     :o_ol_cnt, :o_all_local, :cr_date);
        EXIT;

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



-------------------------
tkvcinin.sql
-------------------------
-- The initnew package for storing variables used in the
-- New Order anonymous block

CREATE OR REPLACE PACKAGE initnew
AS
 TYPE intarray IS TABLE OF INTEGER index by binary_integer;
 TYPE distarray IS TABLE OF VARCHAR(24) index by binary_integer;
 nulldate       DATE;
 s_dist                   distarray;
 idx1arr            intarray;
 s_remote           intarray;
 PROCEDURE new_init(idxarr intarray);
END initnew;
/
show errors;

CREATE OR REPLACE PACKAGE BODY initnew AS
  PROCEDURE new_init (idxarr  intarray)
  IS
  BEGIN
        -- initialize null date
   nulldate := TO_DATE('01-01-1811', 'MM-DD-YYYY');
    idx1arr := idxarr;
  END new_init;
END initnew;
/
show errors
exit


-------------------------
tkvcpnew.sql
-------------------------

-- New Order Anonymous block

  DECLARE
      idx                      BINARY_INTEGER;
      dummy_local              BINARY_INTEGER;
      not_serializable         EXCEPTION;
      PRAGMA EXCEPTION_INIT(not_serializable,-8177);
      deadlock                 EXCEPTION;
      PRAGMA EXCEPTION_INIT(deadlock,-60);
      snapshot_too_old         EXCEPTION;
      PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
   PROCEDURE u1 IS
   BEGIN

        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_01,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                  ELSE (CASE WHEN s_data NOT LIKE '%original%'
                        THEN 'G'
                        ELSE 'B'
                        END)
            END
          BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u1;

   PROCEDURE u2 IS
   BEGIN
        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_02,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE (CASE WHEN s_data NOT LIKE '%original%'
                       THEN 'G'
                       ELSE 'B'
                       END)
           END
          BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u2;

   PROCEDURE u3 IS
   BEGIN
        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_03,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE (CASE WHEN s_data NOT LIKE '%original%'
                       THEN 'G'
                       ELSE 'B'
                       END)
            END
        BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u3;

   PROCEDURE u4 IS
   BEGIN
        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
```

```
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_04,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE (CASE WHEN s_data NOT LIKE '%original%'
                       THEN 'G'
                       ELSE 'B'
                       END)
           END
          BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u4;

   PROCEDURE u5 IS
   BEGIN
        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_05,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                  ELSE (CASE WHEN s_data NOT LIKE '%original%'
                        THEN 'G'
                        ELSE 'B'
                        END)
           END
          BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u5;

   PROCEDURE u6 IS
   BEGIN
        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_06,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE (CASE WHEN s_data NOT LIKE '%original%'
                       THEN 'G'
                       ELSE 'B'
                       END)
            END
          BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u6;

   PROCEDURE u7 IS
   BEGIN
        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_07,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE (CASE WHEN s_data NOT LIKE '%original%'
                       THEN 'G'
                       ELSE 'B'
                       END)
            END
          BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u7;

   PROCEDURE u8 IS
   BEGIN
        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_08,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE (CASE WHEN s_data NOT LIKE '%original%'
                       THEN 'G'
                       ELSE 'B'
                       END)
            END
          BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u8;

   PROCEDURE u9 IS
   BEGIN
        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_09,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE (CASE WHEN s_data NOT LIKE '%original%'
                       THEN 'G'
                       ELSE 'B'
                       END)
            END
          BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u9;

   PROCEDURE u10 IS
   BEGIN
        FORALL idx IN 1 .. :o_ol_cnt
          UPDATE stock_item
          SET s_order_cnt = s_order_cnt + 1,
          s_ytd = s_ytd + :ol_quantity(idx),
          s_remote_cnt = s_remote_cnt + :s_remote(idx),
          s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) +
10
                             THEN s_quantity +91
                             ELSE s_quantity
                        END) - :ol_quantity(idx)
          WHERE i_id = :ol_i_id(idx)
          AND s_w_id = :ol_supply_w_id(idx)
          RETURNING i_price, i_name, s_quantity, s_dist_10,
                    i_price*:ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%original%'
                 THEN 'G'
                 ELSE (CASE WHEN s_data NOT LIKE '%original%'
                       THEN 'G'
                       ELSE 'B'
                       END)
            END
          BULK COLLECT INTO :i_price, :i_name, :s_quantity,
initnew.s_dist,
                            :ol_amount,:brand_generic;
   END u10;

   PROCEDURE fix_items IS
     rows_lost                      BINARY_INTEGER;
     max_index                      BINARY_INTEGER;
     temp_index                     BINARY_INTEGER;
   BEGIN
-- gotta shift price, name, s_quantity, brand_generic, s_dist,
ol_amount
        idx := 1;
-- found 0 bad rows
        rows_lost := 0;
-- so many rows in out array to begin with
          max_index := sql%rowcount;

        WHILE (max_index != :o_ol_cnt) LOOP

-- find item where item ids dont match
            WHILE (idx <= sql%rowcount AND
                        sql%bulk_rowcount(idx + rows_lost) = 1)
            LOOP
              idx := idx + 1;
            END LOOP;

-- shift the items please
            temp_index := max_index;
            WHILE (temp_index >= idx + rows_lost) LOOP
```

```
             :i_price(temp_index + 1)       :=
:i_price(temp_index);
            :i_name(temp_index + 1)        :=
:i_name(temp_index);
            :s_quantity(temp_index + 1)    :=
:s_quantity(temp_index);
            :ol_amount(temp_index +1)      :=
:ol_amount(temp_index);
            initnew.s_dist(temp_index + 1) :=
initnew.s_dist(temp_index);
             :brand_generic(temp_index + 1) :=
:brand_generic(temp_index);
             temp_index := temp_index - 1;
         END LOOP;

--  values for the non-existent items if not at end
          IF (idx + rows_lost <= :o_ol_cnt) THEN
            :i_price(idx + rows_lost)     :=   0;
            :i_name(idx + rows_lost)      :=   'NO ITEM';
           :ol_amount(idx + rows_lost)   :=   0;
           :s_quantity(idx + rows_lost)  :=   0;
           initnew.s_dist(idx + rows_lost) := NULL;
           :brand_generic(idx + rows_lost) := ' ';

--  one more bad row
           rows_lost := rows_lost + 1;
           max_index := max_index + 1;
            END IF;

       END LOOP;
   END fix_items;

   BEGIN
      LOOP BEGIN
         UPDATE dist SET d_next_o_id = d_next_o_id + 1
          WHERE d_id = :d_id AND  d_w_id = :w_id
          RETURNING d_tax, d_next_o_id-1
          INTO :d_tax, :o_id;

         SELECT c_discount, c_last, c_credit, w_tax
          INTO :c_discount, :c_last, :c_credit , :w_tax
          FROM cust , ware
          WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id
          AND  w_id = :w_id;


         INSERT INTO nord (no_o_id, no_d_id, no_w_id)
          VALUES (:o_id, :d_id, :w_id);
          INSERT INTO ordr  (o_id,o_d_id, o_w_id, o_c_id,
o_entry_d,
                             o_carrier_id, o_ol_cnt, o_all_local)
          VALUES (:o_id, :d_id, :w_id, :c_id,
                  :cr_date, 11, :o_ol_cnt, :o_all_local);

         dummy_local :=  :d_id;

         IF (dummy_local < 6) THEN
           IF (dummy_local < 3) THEN
             IF (dummy_local = 1) THEN
                u1;
             ELSE
                u2;
             END IF;
           ELSE
             IF (dummy_local = 3) THEN
                u3;
             ELSIF (dummy_local = 4) then
                u4;
             ELSE
                u5;
             END IF;
           END IF;
         ELSE
           IF (dummy_local < 8) THEN
             IF (dummy_local = 6) THEN
                u6;
             ELSE
                u7;
             END IF;
           ELSE
             IF (dummy_local = 8) THEN
                u8;
             ELSIF (dummy_local = 9) then
                u9;
             ELSE
                u10;
             END IF;
           END IF;
         END IF;


-- cache the no of rows processed
         dummy_local := sql%rowcount;


-- fix the rows if necessary
         IF (dummy_local != :o_ol_cnt )  THEN fix_items; END IF;

-- calculate ol_amount


        FORALL idx IN 1..:o_ol_cnt
         INSERT INTO ordl
            (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_delivery_d,
ol_i_id,
                ol_supply_w_id,
ol_quantity,ol_amount,ol_dist_info)
         VALUES (:o_id, :d_id, :w_id, initnew.idx1arr(idx),
initnew.nulldate,
                 :ol_i_id(idx), :ol_supply_w_id(idx),
                   :ol_quantity(idx), :ol_amount(idx),
initnew.s_dist(idx));

--If there are no errors, then just return without COMMITing
--The COMMIT is done on the driver side by OCI
-- If there are errors, then rollback and set o_ol_cnt to the
processed value

          IF (dummy_local != :o_ol_cnt) THEN
           :o_ol_cnt := dummy_local;
           ROLLBACK;
         END IF;

         EXIT;

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



-------------------------
views.sql
-------------------------

create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
   from dist d, ware w
   where w.w_id = d.d_w_id
/

create  or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
 select i.i_id, s_w_id, i.i_price, i.i_name, i.i_data, s_data,
s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
  from stock s, item i
 where i.i_id = s.s_i_id
/

-------------------------
PRTE command file
-------------------------
```

PRTE command file

```
#####################################################################
#############
#
#
#
#
#                       PRTE COMMAND FILE FOR v6-1-0
#
#
#
#
#
#####################################################################
#############
noecho
#####################################################################
####
#
#
# 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          0.0005
set start_interval  0.0005
set resume_interval 0.05
set stop_interval   0.0005
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.
#
disable init
connect   ~tpcc/bin/web_user  l1        1002      c1
connect   ~tpcc/bin/web_user  l1        1002      c1
connect   ~tpcc/bin/web_user  l1        1002      c1
connect   ~tpcc/bin/web_user  l1        1002      c1
connect   ~tpcc/bin/web_user  l1        1002      c1
connect   ~tpcc/bin/web_user  l1        1002      c2
connect   ~tpcc/bin/web_user  l1        1002      c2
connect   ~tpcc/bin/web_user  l1        1002      c2
connect   ~tpcc/bin/web_user  l1        1002      c2
connect   ~tpcc/bin/web_user  l1        1002      c2
connect   ~tpcc/bin/web_user  l1        1002      c17
connect   ~tpcc/bin/web_user  l1        1002      c17
connect   ~tpcc/bin/web_user  l1        1002      c17
connect   ~tpcc/bin/web_user  l1        1002      c17
connect   ~tpcc/bin/web_user  l1        1002      c17
connect   ~tpcc/bin/web_user  l2        1002      c3
connect   ~tpcc/bin/web_user  l2        1002      c3
connect   ~tpcc/bin/web_user  l2        1002      c3
connect   ~tpcc/bin/web_user  l2        1002      c3
connect   ~tpcc/bin/web_user  l2        1002      c3
connect   ~tpcc/bin/web_user  l2        1002      c4
connect   ~tpcc/bin/web_user  l2        1002      c4
connect   ~tpcc/bin/web_user  l2        1002      c4
connect   ~tpcc/bin/web_user  l2        1002      c4
connect   ~tpcc/bin/web_user  l2        1002      c4
connect   ~tpcc/bin/web_user  l2        1002      c18
connect   ~tpcc/bin/web_user  l2        1002      c18
connect   ~tpcc/bin/web_user  l2        1002      c18
connect   ~tpcc/bin/web_user  l2        1002      c18
connect   ~tpcc/bin/web_user  l2        1002      c18
connect   ~tpcc/bin/web_user  l3        1002      c5
connect   ~tpcc/bin/web_user  l3        1002      c5
connect   ~tpcc/bin/web_user  l3        1002      c5
connect   ~tpcc/bin/web_user  l3        1002      c5
connect   ~tpcc/bin/web_user  l3        1002      c5
connect   ~tpcc/bin/web_user  l3        1002      c6
connect   ~tpcc/bin/web_user  l3        1002      c6
connect   ~tpcc/bin/web_user  l3        1002      c6
connect   ~tpcc/bin/web_user  l3        1002      c6
connect   ~tpcc/bin/web_user  l3        1002      c6
connect   ~tpcc/bin/web_user  l3        1002      c19
connect   ~tpcc/bin/web_user  l3        1002      c19
connect   ~tpcc/bin/web_user  l3        1002      c19
connect   ~tpcc/bin/web_user  l3        1002      c19
connect   ~tpcc/bin/web_user  l3        1002      c19
connect   ~tpcc/bin/web_user  l4        1002      c7
connect   ~tpcc/bin/web_user  l4        1002      c7
connect   ~tpcc/bin/web_user  l4        1002      c7
connect   ~tpcc/bin/web_user  l4        1002      c7
connect   ~tpcc/bin/web_user  l4        1002      c7
connect   ~tpcc/bin/web_user  l4        1002      c8
connect   ~tpcc/bin/web_user  l4        1002      c8
connect   ~tpcc/bin/web_user  l4        1002      c8
connect   ~tpcc/bin/web_user  l4        1002      c8
connect   ~tpcc/bin/web_user  l4        1002      c20
connect   ~tpcc/bin/web_user  l4        1002      c20
connect   ~tpcc/bin/web_user  l4        1002      c20
connect   ~tpcc/bin/web_user  l4        1002      c20
connect   ~tpcc/bin/web_user  l4        1002      c20
connect   ~tpcc/bin/web_user  l5        1002      c9
connect   ~tpcc/bin/web_user  l5        1002      c9
connect   ~tpcc/bin/web_user  l5        1002      c9
connect   ~tpcc/bin/web_user  l5        1002      c9
connect   ~tpcc/bin/web_user  l5        1002      c9
connect   ~tpcc/bin/web_user  l5        1002      c10
connect   ~tpcc/bin/web_user  l5        1002      c10
connect   ~tpcc/bin/web_user  l5        1002      c10
connect   ~tpcc/bin/web_user  l5        1002      c10
connect   ~tpcc/bin/web_user  l5        1002      c10
connect   ~tpcc/bin/web_user  l5        1002      c21
connect   ~tpcc/bin/web_user  l5        1002      c21
connect   ~tpcc/bin/web_user  l5        1002      c21
connect   ~tpcc/bin/web_user  l5        1002      c21
connect   ~tpcc/bin/web_user  l5        1002      c21
connect   ~tpcc/bin/web_user  l6        1002      c11
connect   ~tpcc/bin/web_user  l6        1002      c11
connect   ~tpcc/bin/web_user  l6        1002      c11
connect   ~tpcc/bin/web_user  l6        1002      c11
connect   ~tpcc/bin/web_user  l6        1002      c11
connect   ~tpcc/bin/web_user  l6        1002      c12
connect   ~tpcc/bin/web_user  l6        1002      c12
connect   ~tpcc/bin/web_user  l6        1002      c12
connect   ~tpcc/bin/web_user  l6        1002      c12
connect   ~tpcc/bin/web_user  l6        1002      c12
connect   ~tpcc/bin/web_user  l6        1002      c22
connect   ~tpcc/bin/web_user  l6        1002      c22
connect   ~tpcc/bin/web_user  l6        1002      c22
connect   ~tpcc/bin/web_user  l6        1002      c22
connect   ~tpcc/bin/web_user  l6        1002      c22
connect   ~tpcc/bin/web_user  l7        1002      c13
connect   ~tpcc/bin/web_user  l7        1002      c13
connect   ~tpcc/bin/web_user  l7        1002      c13
connect   ~tpcc/bin/web_user  l7        1002      c13
connect   ~tpcc/bin/web_user  l7        1002      c13
connect   ~tpcc/bin/web_user  l7        1002      c14
connect   ~tpcc/bin/web_user  l7        1002      c14
connect   ~tpcc/bin/web_user  l7        1002      c14
connect   ~tpcc/bin/web_user  l7        1002      c14
connect   ~tpcc/bin/web_user  l7        1002      c14
connect   ~tpcc/bin/web_user  l7        1002      c23
connect   ~tpcc/bin/web_user  l7        1002      c23
connect   ~tpcc/bin/web_user  l7        1002      c23
connect   ~tpcc/bin/web_user  l7        1002      c23
connect   ~tpcc/bin/web_user  l7        1002      c23
connect   ~tpcc/bin/web_user  l8        1002      c15
connect   ~tpcc/bin/web_user  l8        1002      c15
connect   ~tpcc/bin/web_user  l8        1002      c15
connect   ~tpcc/bin/web_user  l8        1002      c15
connect   ~tpcc/bin/web_user  l8        1002      c15
connect   ~tpcc/bin/web_user  l8        1002      c16
connect   ~tpcc/bin/web_user  l8        1002      c16
connect   ~tpcc/bin/web_user  l8        1002      c16
connect   ~tpcc/bin/web_user  l8        1002      c16
connect   ~tpcc/bin/web_user  l8        1002      c16
connect   ~tpcc/bin/web_user  l8        1002      c24
connect   ~tpcc/bin/web_user  l8        1002      c24
connect   ~tpcc/bin/web_user  l8        1002      c24
connect   ~tpcc/bin/web_user  l8        1002      c24
connect   ~tpcc/bin/web_user  l8        1002      c24
#
connect ~tpcc/bin/reduce      localhost 1         localhost
#connect ~tpcc/bin/aide                localhost 1         localhost
fe_21
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 180
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
c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18,c19,
c20,c21,c22,c23,c24
set network_variable FE_USER_COUNTS
          5010,5010,5010,5010,5010,5010,5010,5010,5010,5010,5010,50
10,5010,5010,5010,5010,5010,5010,5010,5010,5010,5010,5010,5010
set network_variable ADMIN_USER_COUNT 0
set network_variable ADMIN_FE_NAMES c1

#################################
#                               #
#  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 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    /webtux.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 WARMUP_TIME                 1200.0
set network_variable STEADY_STATE_TIME           7800.0
set network_variable MEASUREMENT_INTERVAL        7200.0
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         0.05
set network_variable RESUME_DELAY        0.75
set network_variable STOP_DELAY          0.01
# 100 5000 ko
set network_variable SYNC_OFFSET                 32
set network_variable SYNC_UPDATE                 5000
set network_variable MSG_TIMEOUT                 300.0
set network_variable NO_THINK_TIME 14.8
set network_variable NO_THINK_TIME_UPDATE_INTERVAL 15.0
# 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         ra1 ra2 ra3 ra4 ra5 ra6 ra7
ra8
set network_variable BE_USERNAME        tpcc tpcc tpcc tpcc tpcc
tpcc tpcc tpcc
set network_variable DATABASE_TYPE      Oracle
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 B: Database Design

The TPC-C database was created with the following scripts.

```
--------------------------
addfile.sh
--------------------------
#!sh

$SEQLPLUS tpcc/tpcc <<!
   spool log/step5addfile_$1.log
   set echo on
   alter tablespace $1 add datafile '$2' size $3 reuse;
   set echo off
   spool off
   exit ;
!


--------------------------
addtempfile.sh
--------------------------
#!sh

$SEQLPLUS tpcc/tpcc <<!
   spool step5addfile_$1.log
   set echo on
   alter tablespace $1 add tempfile '$2' size $3 reuse;
   set echo off
   spool off
   exit ;
!


--------------------------
addtempts.sh
--------------------------
#!sh

$SEQLPLUS tpcc/tpcc <<!
   spool step5createts_$1.log
   set echo on
   drop tablespace $1 including contents;
   create temporary tablespace $1 tempfile '$2' size $3 reuse
extent management local uniform size $4;
   set echo off
   spool off
   exit ;
!


--------------------------
addts.sh
--------------------------
#!sh

$SEQLPLUS tpcc/tpcc <<!
   spool log/step5createts_$1.log
   set echo on
   drop tablespace $1 including contents;
   create tablespace $1 datafile '$2' size $3 reuse extent
management local uniform size $4 segment space management auto
nologging ;
   set echo off
   spool off
   exit ;
!


--------------------------
addts_rbk.sh
--------------------------
#!sh

$SEQLPLUS tpcc/tpcc <<!
   spool log/step5createts_$1.log
   set echo on
   drop tablespace $1 including contents;
   create tablespace $1 datafile '$2' size $3 reuse extent
management local uniform size $4 nologging ;
   set echo off
   spool off
   exit ;
!


--------------------------
addts_stokcust.sh
--------------------------
#!sh

$SEQLPLUS tpcc/tpcc <<!
   spool step5createts_$1.log
   set echo on
   drop tablespace $1 including contents;
   create tablespace $1 datafile '$2' size $3 reuse extent
management dictionary   nologging ;
   set echo off
   spool off
   exit ;
!


--------------------------
createcustts.sh
--------------------------
#!sh
addts_stokcust.sh cust_1_1 \\\\.\\cust_1_1 15030024K  >
log/cust_log1 2>&1
wait

NUM_FILES=24
I=2

while [ $I -le $NUM_FILES ]
do
addfile.sh cust_1_1 \\\\.\\cust_1_$I 15030024K > log/cust_log$I
2>&1
wait
I=`expr $I + 1`
done

wait


--------------------------
createdistts.sh
--------------------------
#!sh

addts.sh dist_0 \\\\.\\dist_1 50M 10022K > log/dist_log1 2>&1
wait


NUM_FILES=24
I=2

while [ $I -le $NUM_FILES ]
do
addfile.sh dist_0 \\\\.\\dist_$I 50M > log/dist_log$I 2>&1
I=`expr $I + 1`
done

wait


--------------------------
createhistts.sh
--------------------------
#!sh

NUM_FILES=24
I=1

while [ $I -le $NUM_FILES ]
do
addts.sh hist_$I \\\\.\\hist_$I 7998M 1302M > log/hist_log$I 2>&1
wait
I=`expr $I + 1`
done

wait


--------------------------
createicust1ts.sh
--------------------------
#!sh
addts.sh icust1_1 \\\\.\\icust1_1 499M 498M > log/icust1_log1 2>&1
wait

NUM_FILES=24
I=2

while [ $I -le $NUM_FILES ]
do
addfile.sh icust1_$I \\\\.\\icust1_$I 499M  > log/icust1_log$I 2>&1
I=`expr $I + 1`
done

wait


--------------------------
createicust2ts.sh
--------------------------
#!sh
addts.sh icust2_1 \\\\.\\icust2_1 999M 332M > log/icust2_log1 2>&1
wait
```

```
NUM_FILES=24
I=2

while [ $I -le $NUM_FILES ]
do
addfile.sh icust2_1 \\\\.\\icust2_$I 999M  > log/icust2_log$I 2>&1
wait
I=`expr $I + 1`
done

wait


--------------------------
createidistts.sh
--------------------------
#!sh
addts.sh idist_1 \\\\.\\idist_1 5M 4M > log/idist_log1 2>&1
wait


NUM_FILES=24
I=2

while [ $I -le $NUM_FILES ]
do
addfile.sh idist_1 \\\\.\\idist_$I 5M  > log/idist_log$I 2>&1
I=`expr $I + 1`
done

wait


--------------------------
createiordr1ts.sh
--------------------------
#!sh


NUM_FILES=24
I=1

while [ $I -le $NUM_FILES ]
do
addts.sh iordr1_$I \\\\.\\iordr1_$I 1498M 598M > log/iordr1_log$I
2>&1
I=`expr $I + 1`
done

wait


--------------------------
createiordr2ts.sh
--------------------------
#!sh


NUM_FILES=24
I=1

while [ $I -le $NUM_FILES ]
do
addts.sh iordr2_$I \\\\.\\iordr2_$I 1000M 898M > log/iordr2_log$I
2>&1
I=`expr $I + 1`
done

wait


--------------------------
createistokts.sh
--------------------------
#!sh
addts.sh istok1_1 \\\\.\\istok1_1 999M 998M > log/istok_log1 2>&1
wait


NUM_FILES=24
I=2

while [ $I -le $NUM_FILES ]
do
addfile.sh istok1_1 \\\\.\\istok1_$I 999M  > log/istok1_log$I 2>&1
I=`expr $I + 1`
done

wait


--------------------------
createitemts.sh
--------------------------
#!sh
addts.sh item_0 \\\\.\\item_1 16M  15M
addts.sh iitem_0 \\\\.\\iitem_1 50M  15M
wait


--------------------------
createiwarets.sh
--------------------------
#!sh
addts.sh iware_1 \\\\.\\iware_1 5M 4M > log/iware_log1 2>&1
wait


NUM_FILES=24
I=2

while [ $I -le $NUM_FILES ]
do
addfile.sh iware_1 \\\\.\\iware_$I 5M  > log/iware_log$I 2>&1
I=`expr $I + 1`
done
wait


--------------------------
createnordts.sh
--------------------------
#!sh


NUM_FILES=24
I=1

while [ $I -le $NUM_FILES ]
do
addts.sh nord_$I \\\\.\\nord_$I 400M 178M > log/nord_log$I 2>&1
I=`expr $I + 1`
done

wait


--------------------------
createordlts.sh
--------------------------
#!sh


NUM_FILES=24
I=1

while [ $I -le $NUM_FILES ]
do
addts.sh ordl_$I \\\\.\\ordl_$I 16010M 4002M > log/ordl_log$I 2>&1
wait
I=`expr $I + 1`
done
wait


--------------------------
createordrts.sh
--------------------------
#!sh


NUM_FILES=24
I=1

while [ $I -le $NUM_FILES ]
do
addts.sh ordr_$I \\\\.\\ordr_$I 7998M 944M > log/ordr_log$I 2>&1
wait
I=`expr $I + 1`
done

wait


--------------------------
createrbk_bld.sh
--------------------------
#!sh


NUM_RBK=30
I=1

while [ $I -le $NUM_RBK ]
do
sqlplus '/ as sysdba' << !!
alter rollback segment t$I offline;
drop public rollback segment t$I;
create public rollback segment t$I
       storage (initial 200K minextents 2 next 200K);
alter rollback segment t$I online;
quit;
!!
I=`expr $I + 1`
done


--------------------------
createrbk_run.sh
--------------------------
#!sh

NUM_TS=8
NUM_RBK=100
I=1
J=1

while [ $I -le $NUM_TS ]
do
         while [ $J -le $NUM_RBK ]
         do
         sqlplus tpcc/tpcc  << !!
         create rollback segment t_${I}_${J} tablespace undo_$I;
         alter rollback segment t_${I}_${J} online;
         quit;
         !!
```

```
          J=`expr $J + 1`
          done
J=1
I=`expr $I + 1`
done


--------------------------
createrbkts.sh
--------------------------
#!sh

NUM_TS=8
I=1
J=1


while [ $I -le $NUM_TS ]
do
addts_rbk.sh undo_$I \\\\.\\roll_$J 498M 40K > log/rbk_log$I 2>&1
I=`expr $I + 1`
J=`expr $J + 3`
done

wait

sqlplus tpcc/tpcc << !!

alter tablespace undo_1 add datafile '\\.\ROLL_2' size 496M reuse;
alter tablespace undo_1 add datafile '\\.\ROLL_3' size 496M reuse;
alter tablespace undo_2 add datafile '\\.\ROLL_5' size 496M reuse;
alter tablespace undo_2 add datafile '\\.\ROLL_6' size 496M reuse;
alter tablespace undo_3 add datafile '\\.\ROLL_8' size 496M reuse;
alter tablespace undo_3 add datafile '\\.\ROLL_9' size 496M reuse;
alter tablespace undo_4 add datafile '\\.\ROLL_11' size 496M reuse;
alter tablespace undo_4 add datafile '\\.\ROLL_12' size 496M reuse;
alter tablespace undo_5 add datafile '\\.\ROLL_14' size 496M reuse;
alter tablespace undo_5 add datafile '\\.\ROLL_15' size 496M reuse;
alter tablespace undo_6 add datafile '\\.\ROLL_17' size 496M reuse;
alter tablespace undo_6 add datafile '\\.\ROLL_18' size 496M reuse;
alter tablespace undo_7 add datafile '\\.\ROLL_20' size 496M reuse;
alter tablespace undo_7 add datafile '\\.\ROLL_21' size 496M reuse;
alter tablespace undo_8 add datafile '\\.\ROLL_23' size 496M reuse;
alter tablespace undo_8 add datafile '\\.\ROLL_24' size 496M reuse;

quit;
!!


--------------------------
createstokts.sh
--------------------------
#!sh
addts_stokcust.sh stok_1_1 \\\\.\\stok_1_1 10020024K >
log/stok_log1_1 2>&1
wait
addfile.sh stok_1_1 \\\\.\\stok_2_1 10020024K > log/stok_log2_1
2>&1


NUM_FILES=24
I=2

while [ $I -le $NUM_FILES ]
do
addfile.sh stok_1_1 \\\\.\\stok_1_$I 10020024K > log/stok_log1_$I
2>&1
addfile.sh stok_1_1 \\\\.\\stok_2_$I 10020024K > log/stok_log2_$I
2>&1
wait
I=`expr $I + 1`
done

wait


--------------------------
createtempts.sh
--------------------------
addtempts.sh temp_1 \\\\.\\temp_1 15000M 50M  > log/tmp_log
wait


--------------------------
createwarets.sh
--------------------------
#!sh
addts.sh ware_0 \\\\.\\ware_1 50M  2004K > log/ware_log1
wait


NUM_FILES=24
I=2

while [ $I -le $NUM_FILES ]
do
addfile.sh ware_0 \\\\.\\ware_$I 50M > log/ware_log$I 2>&1
I=`expr $I + 1`
done

wait


--------------------------
loadcust.sh
--------------------------
#!sh
tpccload.exe -M 12024 -c -b 1 -e 1500 > log/step26loadcust.log0
2>&1 &
tpccload.exe -M 12024 -c -b 1501 -e 3000 > log/step26loadcust.log1
2>&1 &
tpccload.exe -M 12024 -c -b 3001 -e 4500 > log/step26loadcust.log2
2>&1 &
tpccload.exe -M 12024 -c -b 4501 -e 6000 > log/step26loadcust.log3
2>&1 &
tpccload.exe -M 12024 -c -b 6001 -e 7500 > log/step26loadcust.log4
2>&1 &
tpccload.exe -M 12024 -c -b 7501 -e 9000 > log/step26loadcust.log5
2>&1 &
tpccload.exe -M 12024 -c -b 9001 -e 10500 > log/step26loadcust.log6
2>&1 &
tpccload.exe -M 12024 -c -b 10501 -e 12024 >
log/step26loadcust.log7 2>&1 &
wait


--------------------------
loadhist.sh
--------------------------
tpccload -M 12024 -h -b 1 -e 501 > log/loadhist.log1 2>&1 &
tpccload -M 12024 -h -b 502 -e 1002 > log/loadhist.log2 2>&1 &
tpccload -M 12024 -h -b 1003 -e 1503 > log/loadhist.log3 2>&1 &
tpccload -M 12024 -h -b 1504 -e 2004 > log/loadhist.log4 2>&1 &
tpccload -M 12024 -h -b 2005 -e 2505 > log/loadhist.log5 2>&1 &
tpccload -M 12024 -h -b 2506 -e 3006 > log/loadhist.log6 2>&1 &
wait
tpccload -M 12024 -h -b 3007 -e 3507 > log/loadhist.log7 2>&1 &
tpccload -M 12024 -h -b 3508 -e 4008 > log/loadhist.log8 2>&1 &
tpccload -M 12024 -h -b 4009 -e 4509 > log/loadhist.log9 2>&1 &
tpccload -M 12024 -h -b 4510 -e 5010 > log/loadhist.log10 2>&1 &
tpccload -M 12024 -h -b 5011 -e 5511 > log/loadhist.log11 2>&1 &
tpccload -M 12024 -h -b 5512 -e 6012 > log/loadhist.log12 2>&1 &
wait
tpccload -M 12024 -h -b 6013 -e 6513 > log/loadhist.log13 2>&1 &
tpccload -M 12024 -h -b 6514 -e 7014 > log/loadhist.log14 2>&1 &
tpccload -M 12024 -h -b 7015 -e 7515 > log/loadhist.log15 2>&1 &
tpccload -M 12024 -h -b 7516 -e 8016 > log/loadhist.log16 2>&1 &
tpccload -M 12024 -h -b 8017 -e 8517 > log/loadhist.log17 2>&1 &
tpccload -M 12024 -h -b 8518 -e 9018 > log/loadhist.log18 2>&1 &
wait
tpccload -M 12024 -h -b 9019 -e 9519 > log/loadhist.log19 2>&1 &
tpccload -M 12024 -h -b 9520 -e 10020 > log/loadhist.log20 2>&1 &
tpccload -M 12024 -h -b 10021 -e 10521 > log/loadhist.log21 2>&1 &
tpccload -M 12024 -h -b 10522 -e 11022 > log/loadhist.log22 2>&1 &
tpccload -M 12024 -h -b 11023 -e 11523 > log/loadhist.log23 2>&1 &
tpccload -M 12024 -h -b 11524 -e 12024 > log/loadhist.log24 2>&1 &
wait


--------------------------
loadnord.sh
--------------------------
tpccload -M 12024 -n -b 1 -e 501 > log/loadnord.log1 2>&1 &
tpccload -M 12024 -n -b 502 -e 1002 > log/loadnord.log2 2>&1 &
tpccload -M 12024 -n -b 1003 -e 1503 > log/loadnord.log3 2>&1 &
tpccload -M 12024 -n -b 1504 -e 2004 > log/loadnord.log4 2>&1 &
tpccload -M 12024 -n -b 2005 -e 2505 > log/loadnord.log5 2>&1 &
tpccload -M 12024 -n -b 2506 -e 3006 > log/loadnord.log6 2>&1 &
wait
tpccload -M 12024 -n -b 3007 -e 3507 > log/loadnord.log7 2>&1 &
tpccload -M 12024 -n -b 3508 -e 4008 > log/loadnord.log8 2>&1 &
tpccload -M 12024 -n -b 4009 -e 4509 > log/loadnord.log9 2>&1 &
tpccload -M 12024 -n -b 4510 -e 5010 > log/loadnord.log10 2>&1 &
tpccload -M 12024 -n -b 5011 -e 5511 > log/loadnord.log11 2>&1 &
tpccload -M 12024 -n -b 5512 -e 6012 > log/loadnord.log12 2>&1 &
wait
tpccload -M 12024 -n -b 6013 -e 6513 > log/loadnord.log13 2>&1 &
tpccload -M 12024 -n -b 6514 -e 7014 > log/loadnord.log14 2>&1 &
tpccload -M 12024 -n -b 7015 -e 7515 > log/loadnord.log15 2>&1 &
tpccload -M 12024 -n -b 7516 -e 8016 > log/loadnord.log16 2>&1 &
tpccload -M 12024 -n -b 8017 -e 8517 > log/loadnord.log17 2>&1 &
tpccload -M 12024 -n -b 8518 -e 9018 > log/loadnord.log18 2>&1 &
wait
tpccload -M 12024 -n -b 9019 -e 9519 > log/loadnord.log19 2>&1 &
tpccload -M 12024 -n -b 9520 -e 10020 > log/loadnord.log20 2>&1 &
tpccload -M 12024 -n -b 10021 -e 10521 > log/loadnord.log21 2>&1 &
tpccload -M 12024 -n -b 10522 -e 11022 > log/loadnord.log22 2>&1 &
tpccload -M 12024 -n -b 11023 -e 11523 > log/loadnord.log23 2>&1 &
tpccload -M 12024 -n -b 11524 -e 12024 > log/loadnord.log24 2>&1 &
wait


--------------------------
loadordrordl.sh
--------------------------
tpccload.exe -M 12024 -o temp -b 1 -e 501 >  log/loadordrordl.log1
2>&1 &
tpccload.exe -M 12024 -o temp -b 502 -e 1002 >
log/loadordrordl.log2 2>&1 &
tpccload.exe -M 12024 -o temp -b 1003 -e 1503 >
log/loadordrordl.log3 2>&1 &
tpccload.exe -M 12024 -o temp -b 1504 -e 2004 >
log/loadordrordl.log4 2>&1 &
tpccload.exe -M 12024 -o temp -b 2005 -e 2505 >
log/loadordrordl.log5 2>&1 &
tpccload.exe -M 12024 -o temp -b 2506 -e 3006 >
log/loadordrordl.log6 2>&1 &
wait
tpccload.exe -M 12024 -o temp -b 3007 -e 3507 >
log/loadordrordl.log7 2>&1 &
tpccload.exe -M 12024 -o temp -b 3508 -e 4008 >
log/loadordrordl.log8 2>&1 &
tpccload.exe -M 12024 -o temp -b 4009 -e 4509 >
log/loadordrordl.log9 2>&1 &
```

```
tpccload.exe -M 12024 -o temp -b 4510 -e 5010 >
log/loadordrordl.log10 2>&1 &
tpccload.exe -M 12024 -o temp -b 5011 -e 5511 >
log/loadordrordl.log11 2>&1 &
tpccload.exe -M 12024 -o temp -b 5512 -e 6012 >
log/loadordrordl.log12 2>&1 &
wait
tpccload.exe -M 12024 -o temp -b 6013 -e 6513 >
log/loadordrordl.log13 2>&1 &
tpccload.exe -M 12024 -o temp -b 6514 -e 7014 >
log/loadordrordl.log14 2>&1 &
tpccload.exe -M 12024 -o temp -b 7015 -e 7515 >
log/loadordrordl.log15 2>&1 &
tpccload.exe -M 12024 -o temp -b 7516 -e 8016 >
log/loadordrordl.log16 2>&1 &
tpccload.exe -M 12024 -o temp -b 8017 -e 8517 >
log/loadordrordl.log17 2>&1 &
tpccload.exe -M 12024 -o temp -b 8518 -e 9018 >
log/loadordrordl.log18 2>&1 &
wait
tpccload.exe -M 12024 -o temp -b 9019 -e 9519 >
log/loadordrordl.log19 2>&1 &
tpccload.exe -M 12024 -o temp -b 9520 -e 10020 >
log/loadordrordl.log20 2>&1 &
tpccload.exe -M 12024 -o temp -b 10021 -e 10521 >
log/loadordrordl.log21 2>&1 &
tpccload.exe -M 12024 -o temp -b 10522 -e 11022 >
log/loadordrordl.log22 2>&1 &
tpccload.exe -M 12024 -o temp -b 11023 -e 11523 >
log/loadordrordl.log23 2>&1 &
tpccload.exe -M 12024 -o temp -b 11524 -e 12024 >
log/loadordrordl.log24 2>&1 &
wait


--------------------------
loadstok.sh
--------------------------
tpccload.exe -M 12024 -S -j 1 -k 12500 > log/step27loadstok.log0
2>&1 &
tpccload.exe -M 12024 -S -j 12501 -k 25000 >
log/step27loadstok.log1 2>&1 &
tpccload.exe -M 12024 -S -j 25001 -k 37500 >
log/step27loadstok.log2 2>&1 &
tpccload.exe -M 12024 -S -j 37501 -k 50000 >
log/step27loadstok.log3 2>&1 &
tpccload.exe -M 12024 -S -j 50001 -k 62500 >
log/step27loadstok.log4 2>&1 &
tpccload.exe -M 12024 -S -j 62501 -k 75000 >
log/step27loadstok.log5 2>&1 &
tpccload.exe -M 12024 -S -j 75001 -k 87500 >
log/step27loadstok.log6 2>&1 &
tpccload.exe -M 12024 -S -j 87501 -k 100000 >
log/step27loadstok.log7 2>&1 &
wait


--------------------------
step10createdist.sh
--------------------------
#!sh
$SEQLPLUS tpcc/tpcc @step10createdist > junk 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi


--------------------------
step11createcust.sh
--------------------------
#!sh
$SEQLPLUS tpcc/tpcc @step11createcust > junkcusttable 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi


--------------------------
step12createhist.sh
--------------------------
#!sh
$SEQLPLUS tpcc/tpcc @step12createhist > junkhist 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi


--------------------------
step13createordr.sh
--------------------------
#!sh
$SEQLPLUS tpcc/tpcc @step13createordr > junkordr 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi


--------------------------
step14createnord.sh
--------------------------
#!sh
$SEQLPLUS tpcc/tpcc @step14createnord > junknord 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi


--------------------------
step15createordl.sh
--------------------------
#!sh
$SEQLPLUS tpcc/tpcc @step15createordl > junkordl 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi


--------------------------
step16createstok.sh
--------------------------
#!sh
$SEQLPLUS tpcc/tpcc @step16createstok > junk 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi


--------------------------
step17createitem.sh
--------------------------
#!sh
$SEQLPLUS tpcc/tpcc @step17createitem > junk 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi


--------------------------
tpccenv.sh
--------------------------
#!sh
#SQLDBA=svrmgrl
#export SQLDBA
SEQLPLUS=sqlplus
export SEQLPLUS
TPCCLOAD=tpccload.exe
export TPCCLOAD
ORACLE_SID=TPC1
export ORACLE_SID



--------------------------
createPERFts.sql
--------------------------
create tablespace PERF datafile '\\.\PERF_1' size 498M reuse;
alter  tablespace PERF add datafile '\\.\PERF_2' size 498M reuse;
alter  tablespace PERF add datafile '\\.\PERF_3' size 498M reuse;
alter  tablespace PERF add datafile '\\.\PERF_4' size 498M reuse;


--------------------------
createcust.sql
--------------------------
spool step11createcust.log;
set echo on;
drop table cust;
drop cluster custcluster including tables;
set timing on;
create cluster custcluster (
c_id    number(5,0)
,c_d_id    number(2,0)
,c_w_id    number(5,0)
  )
  single    table
  hashkeys    360720000
  hash is   (c_w_id * 30000 + c_id * 10 + c_d_id - 30011)
  size    850
  initrans    3
  pctfree    0
  storage (initial 2004004K next 2004000K pctincrease 0 maxextents
200 buffer_pool recycle)
  tablespace    cust_1_1;
```

```
create table cust (
        c_id            number(5,0),
        c_d_id          number(2,0),
        c_w_id          number(5,0),
        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          varchar2(500)
)
cluster custcluster (c_id
,c_d_id
,c_w_id
);
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
createdb.sql
--------------------------
spool createdb.log
set echo on
startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxinstances 8
  datafile '\\.\system_1' size 96M reuse,
           '\\.\system_2' size 96M reuse,
           '\\.\system_3' size 96M reuse,
           '\\.\system_4' size 96M reuse,
           '\\.\system_5' size 96M reuse,
           '\\.\system_6' size 96M reuse,
           '\\.\system_7' size 96M reuse,
           '\\.\system_8' size 96M reuse
  logfile group 1 ('\\.\log_1_1') size 5717M reuse,
           group 2 ('\\.\log_1_2') size 5717M reuse;
alter database add logfile thread 2 group 3 ('\\.\log_2_1') size
5717M reuse,
   group 4 ('\\.\log_2_2') size 5717M reuse;
alter database enable public thread 2;
alter database add logfile thread 3 group 5 ('\\.\log_3_1') size
5717M reuse,
   group 6 ('\\.\log_3_2') size 5717M reuse;
alter database enable public thread 3;
alter database add logfile thread 4 group 7 ('\\.\log_4_1') size
5717M reuse,
   group 8 ('\\.\log_4_2') size 5717M reuse;
alter database enable public thread 4;
alter database add logfile thread 5 group 9 ('\\.\log_5_1') size
5717M reuse,
   group 10 ('\\.\log_5_2') size 5717M reuse;
alter database enable public thread 5;
alter database add logfile thread 6 group 11 ('\\.\log_6_1') size
5717M reuse,
   group 12 ('\\.\log_6_2') size 5717M reuse;
alter database enable public thread 6;
alter database add logfile thread 7 group 13 ('\\.\log_7_1') size
5717M reuse,
   group 14 ('\\.\log_7_2') size 5717M reuse;
alter database enable public thread 7;
alter database add logfile thread 8 group 15 ('\\.\log_8_1') size
5717M reuse,
   group 16 ('\\.\log_8_2') size 5717M reuse;
alter database enable public thread 8;
spool off
@createuser
set echo off
exit sql.sqlcode
!

--------------------------
createddviews.sql
--------------------------
spool step6createddviews.log
@c:/oracle/ora90/rdbms/admin/catalog
@c:/oracle/ora90/rdbms/admin/catproc
@c:/oracle/ora90/rdbms/admin/catparr
spool off
exit sql.sqlcode;


--------------------------
createstok.sql
--------------------------
spool step16createstok.log;
set echo on;
set timing on
drop cluster stokcluster including tables;
set timing on;
create cluster stokcluster (
s_i_id    number(6,0)
,s_w_id    number(5,0)
  )
  single    table
  hashkeys    1202400000
  hash is   (abs(s_i_id - 1) * 1503  + mod((s_w_id - 1), 1503) +
trunc ((s_w_id - 1) / 1503) * 150300000)
  size    380
  initrans    3
  pctfree    0
  storage ( initial 1670002K next 1670000K pctincrease 0 maxextents
300 buffer_pool)
  tablespace    stok_1_1;

create table stok (
        s_i_id          number(6,0),
        s_w_id          number(5,0),
        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
);
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
createuser.sql
--------------------------
spool stepcreateuser.log;
set echo on;
create user tpcc identified by tpcc;
grant dba to tpcc;
set echo off;
spool off;


--------------------------
createware.sql
--------------------------
spool step9createware.log;
set echo on;
drop table ware;
drop cluster warecluster including tables;
set timing on;
create cluster warecluster (
w_id      number(5,0)
)
          single table
          hashkeys  12024
          size              3072
          hash is   w_id - 1
          initrans  3
          pctfree           0
          storage (initial 2006k next 2004k pctincrease 0 freelist
groups 3)
          tablespace ware_0;

create table ware(
        w_id            number(5,0),
        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);

spool off;
set echo off;
exit sql.sqlcode;


--------------------------
dml.sql
--------------------------
REM=============================================================
==+
REM        Copyright (c) 1996  Oracle Corp, Redwood Shores, CA
|
REM                    OPEN SYSTEMS PERFORMANCE GROUP
|
REM                         All Rights Reserved
|
REM=============================================================
==+
REM FILENAME
REM     dml.sql
REM DESCRIPTION
REM     Disable table locks for TPC-C tables.
REM USAGE
REM     sqlplus tpcc/tpcc dml.sql
```

```
REM=================================================================
===
connect tpcc/tpcc;
set echo on;
   alter table ware disable table lock;
   alter table dist disable table lock;
   alter table cust disable table lock;
   alter table hist disable table lock;
   alter table item disable table lock;
   alter table stok disable table lock;
   alter table ordr disable table lock;
   alter table nord disable table lock;
   alter table ordl disable table lock;
set echo off;


-------------------------
step10createdist.sql
-------------------------
spool step10createdist.log;
set echo on;
drop table dist;
drop cluster distcluster including tables;
set timing on;
create cluster distcluster (
          d_w_id    number(5,0),
          d_id      number(2,0)
)
          single table
          hashkeys  120240
          size                3072
          hash is             (d_w_id - 1) * 10 + d_id
          initrans  4
          pctfree             0
          storage ( initial 20042k next 20040k pctincrease 0
freelist groups 3)
          tablespace dist_0;

create table dist (
        d_id            number(2,0),
          d_w_id          number(5,0),
        d_ytd           number,
        d_tax           number,
        d_next_o_id     number,
        d_name          varchar2(10),
        d_street_1      varchar2(20),
        d_street_2      varchar2(20),
        d_city          varchar2(20),
        d_state         char(2),
        d_zip           char(9)
)
cluster distcluster (d_w_id, d_id);

spool off;
set echo off;
exit sql.sqlcode;


-------------------------
step11createcust.sql
-------------------------
spool step11createcust.log;
set echo on;
drop table cust;
drop cluster custcluster including tables;
set timing on;
create cluster custcluster (
c_id    number(5,0)
,c_d_id    number(2,0)
,c_w_id    number(5,0)
  )
  single    table
  hashkeys    360720000
  hash is   (c_w_id * 30000 + c_id * 10 + c_d_id - 30011)
  size     850
  initrans    3
  pctfree    0
  storage (initial 1878752k next 1878750k pctincrease 0 maxextents
200 buffer_pool recycle)
  tablespace    cust_1_1;
create table cust (
        c_id            number(5,0),
        c_d_id          number(2,0),
        c_w_id          number(5,0),
        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          varchar2(500)
)
cluster custcluster (c_id
,c_d_id
,c_w_id
);
spool off;
set echo off;
exit sql.sqlcode;


-------------------------
step12createhist.sql
-------------------------
spool step12createhist.log;
set echo on;
drop table hist;
set timing on;
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)
)
  tablespace hist_1
  partition by range(h_w_id)
  (
    partition hist_1 values less than (502)
      tablespace hist_1,
    partition hist_2 values less than (1003)
      tablespace hist_2,
    partition hist_3 values less than (1504)
      tablespace hist_3,
    partition hist_4 values less than (2005)
      tablespace hist_4,
    partition hist_5 values less than (2506)
      tablespace hist_5,
    partition hist_6 values less than (3007)
      tablespace hist_6,
    partition hist_7 values less than (3508)
      tablespace hist_7,
    partition hist_8 values less than (4009)
      tablespace hist_8,
    partition hist_9 values less than (4510)
      tablespace hist_9,
    partition hist_10 values less than (5011)
      tablespace hist_10,
    partition hist_11 values less than (5512)
      tablespace hist_11,
    partition hist_12 values less than (6013)
      tablespace hist_12,
    partition hist_13 values less than (6514)
      tablespace hist_13,
    partition hist_14 values less than (7015)
      tablespace hist_14,
    partition hist_15 values less than (7516)
      tablespace hist_15,
    partition hist_16 values less than (8017)
      tablespace hist_16,
    partition hist_17 values less than (8518)
      tablespace hist_17,
    partition hist_18 values less than (9019)
      tablespace hist_18,
    partition hist_19 values less than (9520)
      tablespace hist_19,
    partition hist_20 values less than (10021)
      tablespace hist_20,
    partition hist_21 values less than (10522)
      tablespace hist_21,
    partition hist_22 values less than (11023)
      tablespace hist_22,
    partition hist_23 values less than (11524)
      tablespace hist_23,
    partition hist_24 values less than (maxvalue)
      tablespace hist_24
  )
  initrans    4
  pctfree    5
  storage ( freelists 22 freelist groups 96 );
spool off;
set echo off;
exit sql.sqlcode;


-------------------------
step13createordr.sql
-------------------------
spool step13createordr.log;
set echo on;
drop table ordr;
set timing on;
create table ordr (
        o_id            number,
        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
)
  tablespace ordr_1
  partition by range(o_w_id)
  (
    partition ordr_1 values less than (502)
      tablespace ordr_1,
    partition ordr_2 values less than (1003)
      tablespace ordr_2,
    partition ordr_3 values less than (1504)
      tablespace ordr_3,
    partition ordr_4 values less than (2005)
```

```
    tablespace ordr_4,
  partition ordr_5 values less than (2506)
    tablespace ordr_5,
  partition ordr_6 values less than (3007)
    tablespace ordr_6,
  partition ordr_7 values less than (3508)
    tablespace ordr_7,
  partition ordr_8 values less than (4009)
    tablespace ordr_8,
  partition ordr_9 values less than (4510)
    tablespace ordr_9,
  partition ordr_10 values less than (5011)
    tablespace ordr_10,
  partition ordr_11 values less than (5512)
    tablespace ordr_11,
  partition ordr_12 values less than (6013)
    tablespace ordr_12,
  partition ordr_13 values less than (6514)
    tablespace ordr_13,
  partition ordr_14 values less than (7015)
    tablespace ordr_14,
  partition ordr_15 values less than (7516)
    tablespace ordr_15,
  partition ordr_16 values less than (8017)
    tablespace ordr_16,
  partition ordr_17 values less than (8518)
    tablespace ordr_17,
  partition ordr_18 values less than (9019)
    tablespace ordr_18,
  partition ordr_19 values less than (9520)
    tablespace ordr_19,
  partition ordr_20 values less than (10021)
    tablespace ordr_20,
  partition ordr_21 values less than (10522)
    tablespace ordr_21,
  partition ordr_22 values less than (11023)
    tablespace ordr_22,
  partition ordr_23 values less than (11524)
    tablespace ordr_23,
  partition ordr_24 values less than (maxvalue)
    tablespace ordr_24
)
initrans   4
pctfree    5
storage ( freelists 22 freelist groups 96 );
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step14createnord.sql
--------------------------
spool step14createnord.log;
set echo on;
drop table nord;
set timing on;
create table nord (
       no_w_id         number,
       no_d_id         number,
       no_o_id         number,
         constraint inord primary key (no_w_id, no_d_id, no_o_id)
)
  organization index
  tablespace nord_1
  partition by range(no_w_id)
  (
    partition nord_1 values less than (502)
      tablespace nord_1,
    partition nord_2 values less than (1003)
      tablespace nord_2,
    partition nord_3 values less than (1504)
      tablespace nord_3,
    partition nord_4 values less than (2005)
      tablespace nord_4,
    partition nord_5 values less than (2506)
      tablespace nord_5,
    partition nord_6 values less than (3007)
      tablespace nord_6,
    partition nord_7 values less than (3508)
      tablespace nord_7,
    partition nord_8 values less than (4009)
      tablespace nord_8,
    partition nord_9 values less than (4510)
      tablespace nord_9,
    partition nord_10 values less than (5011)
      tablespace nord_10,
    partition nord_11 values less than (5512)
      tablespace nord_11,
    partition nord_12 values less than (6013)
      tablespace nord_12,
    partition nord_13 values less than (6514)
      tablespace nord_13,
    partition nord_14 values less than (7015)
      tablespace nord_14,
    partition nord_15 values less than (7516)
      tablespace nord_15,
    partition nord_16 values less than (8017)
      tablespace nord_16,
    partition nord_17 values less than (8518)
      tablespace nord_17,
    partition nord_18 values less than (9019)
      tablespace nord_18,
    partition nord_19 values less than (9520)
      tablespace nord_19,
    partition nord_20 values less than (10021)
      tablespace nord_20,
    partition nord_21 values less than (10522)
      tablespace nord_21,
    partition nord_22 values less than (11023)
      tablespace nord_22,
    partition nord_23 values less than (11524)
      tablespace nord_23,
    partition nord_24 values less than (maxvalue)
      tablespace nord_24
  )
  initrans   4
  pctfree    5
  storage ( freelists 22 freelist groups 96 );
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step15createordl.sql
--------------------------
spool step15createordl.log;
set echo on;
drop table ordl;
set timing on;
create table ordl (
       ol_w_id         number,
       ol_d_id         number,
       ol_o_id         number,
       ol_number       number,
       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 iordl primary key (ol_w_id, ol_d_id, ol_o_id,
ol_number)
)
  organization index
  tablespace ordl_1
  partition by range(ol_w_id)
  (
    partition ordl_1 values less than (502)
      tablespace ordl_1,
    partition ordl_2 values less than (1003)
      tablespace ordl_2,
    partition ordl_3 values less than (1504)
      tablespace ordl_3,
    partition ordl_4 values less than (2005)
      tablespace ordl_4,
    partition ordl_5 values less than (2506)
      tablespace ordl_5,
    partition ordl_6 values less than (3007)
      tablespace ordl_6,
    partition ordl_7 values less than (3508)
      tablespace ordl_7,
    partition ordl_8 values less than (4009)
      tablespace ordl_8,
    partition ordl_9 values less than (4510)
      tablespace ordl_9,
    partition ordl_10 values less than (5011)
      tablespace ordl_10,
    partition ordl_11 values less than (5512)
      tablespace ordl_11,
    partition ordl_12 values less than (6013)
      tablespace ordl_12,
    partition ordl_13 values less than (6514)
      tablespace ordl_13,
    partition ordl_14 values less than (7015)
      tablespace ordl_14,
    partition ordl_15 values less than (7516)
      tablespace ordl_15,
    partition ordl_16 values less than (8017)
      tablespace ordl_16,
    partition ordl_17 values less than (8518)
      tablespace ordl_17,
    partition ordl_18 values less than (9019)
      tablespace ordl_18,
    partition ordl_19 values less than (9520)
      tablespace ordl_19,
    partition ordl_20 values less than (10021)
      tablespace ordl_20,
    partition ordl_21 values less than (10522)
      tablespace ordl_21,
    partition ordl_22 values less than (11023)
      tablespace ordl_22,
    partition ordl_23 values less than (11524)
      tablespace ordl_23,
    partition ordl_24 values less than (maxvalue)
      tablespace ordl_24
  )
  initrans   4
  pctfree    5
  storage ( freelists 22 freelist groups 96 );
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step16createstok.sql
--------------------------
spool step16createstok.log;
set echo on;
drop table stok;
drop cluster stokcluster including tables;
set timing on;
create cluster stokcluster (
s_i_id    number(6,0)
```

```
,s_w_id    number(5,0)
  )
  single    table
  hashkeys    1202400000
  hash is   (abs(s_i_id - 1) * 501  + mod((s_w_id - 1), 501) +
trunc ((s_w_id - 1) / 501) * 50100000)
  size    350
  initrans    3
  pctfree    0
  storage ( initial 1670002k next 1670000k pctincrease 0 maxextents
300 buffer_pool keep)
  tablespace    stok_1_1;

create table stok (
        s_i_id          number(6,0),
        s_w_id          number(5,0),
        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
);
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step17createitem.sql
--------------------------
spool step17createitem.log;
set echo on;
drop table item;
drop cluster itemcluster including tables;
set timing on;
create cluster itemcluster (
i_id    number(6,0)
  )
  single    table
  hashkeys    100000
  hash is    (i_id + 1)
  size    120
  initrans    3
  pctfree    0
  storage ( buffer_pool keep freelists 22 freelist groups 96 )
  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
);
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step29createiware.sql
--------------------------
spool step29createiware.log;
set echo on;
drop index iware;
set timing on;
create unique index iware on ware (w_id)
  initrans    3
  parallel    8
  pctfree    1
  storage ( freelists 22 freelist groups 16)
  tablespace        iware_1;
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step30createidist.sql
--------------------------
spool step30createidist.log;
set echo on;
drop index idist;
set timing on;
create unique index idist on dist (d_w_id, d_id)
  initrans    3
  parallel    8
  pctfree    5
  storage ( freelists 22 freelist groups 32 )
  tablespace idist_1;
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step31createiitem.sql
--------------------------
spool step31createiitem.log;
set echo on;
drop index iitem;
set timing on;
create unique index iitem on item (i_id)
  initrans    4
  pctfree    5
  storage ( freelists 22 freelist groups 96 )
  tablespace    iitem_0;
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step32createicust1.sql
--------------------------
spool step32createicust1.log;
set echo on;
drop index icust1;
set timing on;
create unique index icust1 on cust (c_w_id, c_d_id, c_id)
  initrans    3
  parallel    4
  pctfree     1
  storage ( freelists 22 freelist groups 96 )
  tablespace    icust1_1;
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step33createicust2.sql
--------------------------
spool step33createicust2.log;
set echo on;
drop index icust2;
set timing on;
create unique index icust2 on cust (c_last, c_w_id, c_d_id,
c_first, c_id)
  initrans    3
  pctfree    1
  tablespace    icust2_1;
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step34createistok.sql
--------------------------
spool step34createistok.log;
set echo on;
drop index istok;
set timing on;
create unique index istok on stok (s_i_id, s_w_id)
  initrans    3
  parallel    8
  pctfree    1
  storage ( freelists 22 freelist groups 96 )
  tablespace    istok1_1;
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step35createiordr1.sql
--------------------------
spool step35createiordr1.log;
set echo on;
drop index iordr1;
set timing on;
create unique index iordr1 on ordr (o_w_id, o_d_id, o_id)
  local
  (
    partition iordr1_0 tablespace iordr1_1,
    partition iordr1_1 tablespace iordr1_2,
    partition iordr1_2 tablespace iordr1_3,
    partition iordr1_3 tablespace iordr1_4,
    partition iordr1_4 tablespace iordr1_5,
    partition iordr1_5 tablespace iordr1_6,
    partition iordr1_6 tablespace iordr1_7,
    partition iordr1_7 tablespace iordr1_8,
    partition iordr1_8 tablespace iordr1_9,
    partition iordr1_9 tablespace iordr1_10,
    partition iordr1_10 tablespace iordr1_11,
    partition iordr1_11 tablespace iordr1_12,
    partition iordr1_12 tablespace iordr1_13,
    partition iordr1_13 tablespace iordr1_14,
    partition iordr1_14 tablespace iordr1_15,
    partition iordr1_15 tablespace iordr1_16,
    partition iordr1_16 tablespace iordr1_17,
    partition iordr1_17 tablespace iordr1_18,
    partition iordr1_18 tablespace iordr1_19,
    partition iordr1_19 tablespace iordr1_20,
    partition iordr1_20 tablespace iordr1_21,
    partition iordr1_21 tablespace iordr1_22,
    partition iordr1_22 tablespace iordr1_23,
    partition iordr1_23 tablespace iordr1_24
  )
  initrans    3
```

```
  parallel    8
  pctfree    1
  storage ( freelists 22 freelist groups 96 );
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
step36createiordr2.sql
--------------------------
spool step36createiordr2.log;
set echo on;
drop index iordr2;
set timing on;
create unique index iordr2 on ordr (o_w_id, o_d_id, o_c_id, o_id)
  local
  (
    partition iordr2_0 tablespace iordr2_1,
    partition iordr2_1 tablespace iordr2_2,
    partition iordr2_2 tablespace iordr2_3,
    partition iordr2_3 tablespace iordr2_4,
    partition iordr2_4 tablespace iordr2_5,
    partition iordr2_5 tablespace iordr2_6,
    partition iordr2_6 tablespace iordr2_7,
    partition iordr2_7 tablespace iordr2_8,
    partition iordr2_8 tablespace iordr2_9,
    partition iordr2_9 tablespace iordr2_10,
    partition iordr2_10 tablespace iordr2_11,
    partition iordr2_11 tablespace iordr2_12,
    partition iordr2_12 tablespace iordr2_13,
    partition iordr2_13 tablespace iordr2_14,
    partition iordr2_14 tablespace iordr2_15,
    partition iordr2_15 tablespace iordr2_16,
    partition iordr2_16 tablespace iordr2_17,
    partition iordr2_17 tablespace iordr2_18,
    partition iordr2_18 tablespace iordr2_19,
    partition iordr2_19 tablespace iordr2_20,
    partition iordr2_20 tablespace iordr2_21,
    partition iordr2_21 tablespace iordr2_22,
    partition iordr2_22 tablespace iordr2_23,
    partition iordr2_23 tablespace iordr2_24
  )
  initrans    4
  parallel    8
  pctfree    25
  storage ( freelists 22 freelist groups 96 );
spool off;
set echo off;
exit sql.sqlcode;


--------------------------
tpcc_ana.sql
--------------------------
rem
rem
================================================================+
rem         Copyright (c) 1995  Oracle Corp, Redwood Shores, CA
|
rem                   OPEN SYSTEMS PERFORMANCE GROUP
|
rem                        All Rights Reserved
|
rem
================================================================+
rem  FILENAME
rem       tpcc_ana.sql
rem  DESCRIPTION
rem       Analyze all tables and indexes of TPC-C database.
rem
================================================================
rem
spool analyze.log;
set echo on;
set timing on;
analyze table ware compute statistics;
analyze table dist compute statistics;
analyze table item estimate statistics;
analyze table hist estimate statistics;
analyze table cust estimate statistics;
analyze table stok estimate statistics;
analyze table ordr estimate statistics;
analyze table nord estimate statistics;
analyze table ordl estimate statistics;
analyze cluster itemcluster estimate statistics;
analyze cluster stokcluster estimate statistics;
analyze cluster custcluster estimate statistics;
analyze cluster distcluster estimate statistics;
analyze cluster warecluster estimate statistics;
analyze index iware compute statistics;
analyze index idist compute statistics;
analyze index icust1 estimate statistics;
analyze index icust2 estimate statistics;
analyze index istok estimate statistics;
analyze index iitem estimate statistics;
analyze index iordr1 estimate statistics;
analyze index iordr2 estimate statistics;
analyze index inord estimate statistics;
analyze index iordl estimate statistics;
spool off;
set echo off;
exit sql.sqlcode;




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

```
tpcc.h
--------------------------
/*
 * $Header: tpcc.h 7030100.1 95/07/19 15:10:55 plai Generic<base> $
Copyr (c) 1993 Oracle
 */
/*================================================================
=+
 |         Copyright (c) 1995  Oracle Corp, Redwood Shores, CA
|
 |                   OPEN SYSTEMS PERFORMANCE GROUP
|
 |                        All Rights Reserved
|

+================================================================
+
 | FILENAME
 |    tpcc.h
 | DESCRIPTION
 |    Include file for TPC-C benchmark programs.

+================================================================
*/

#ifndef TPCC_H
#define TPCC_H

#ifndef FALSE
# define FALSE 0
#endif

#ifndef TRUE
# define TRUE 1
#endif

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

#ifndef boolean
#define boolean int
#endif

#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>
/*
#ifdef __STDC__
#include "ociapr.h"
#else
#include "ocikpr.h"
#endif
*/

typedef struct cda_def csrdef;
typedef struct cda_def ldadef;


/* TPC-C transaction functions */

extern int TPCinit ();
extern int TPCnew ();
extern int TPCpay ();
extern int TPCord ();
extern int TPCdel ();
extern int TPCsto ();
extern void TPCexit ();
extern int TPCdumpinit ();
extern void TPCdumpnew ();
extern void TPCdumppay ();
extern void TPCdumpord ();
extern void TPCdumpdel ();
extern void TPCdumpsto ();
extern void TPCdumpexit ();
extern void userlog();


/* Error codes */

#define RECOVERR -10
#define IRRECERR -20
#define NOERR    111
#define DEL_ERROR -666
#define DEL_DATE_LEN 7
#define NDISTS 10
#define NITEMS 15
#define SQL_BUF_SIZE 8192

#define FULLDATE "dd-mon-yy.hh:mi:ss"
#define SHORTDATE "dd-mm-yyyy"



#define DELRT 80.0

extern int tkvcninit ();
extern int tkvcpinit ();
extern int tkvcoinit ();
extern int tkvcdinit ();
extern int tkvcsinit ();

extern int tkvcn ();
extern int tkvcp ();
extern int tkvco ();
extern int tkvcd ();
extern int tkvcs ();
```

```
extern void tkvcndone ();
extern void tkvcpdone ();
extern void tkvcodone ();
extern void tkvcddone ();
extern void tkvcsdone ();

extern int tkvcss (); /* for alter session to get memory size and
trace */
extern boolean multitranx;
extern int ord_init;


extern void errrpt ();
extern int ocierror(char *fname, int lineno,OCIError *errhp, sword
status);
extern int sqlfile(char *fname, text *linebuf);

extern FILE *lfp;
extern FILE *fopen ();
extern int proc_no;
extern int doid[];

extern int execstatus;
extern int errcode;

extern OCIEnv *tpcenv;
extern OCIServer *tpcsrv;
extern OCIError *errhp;
extern OCISvcCtx *tpcsvc;
extern OCISession *tpcusr;
extern OCIStmt *curntest;
/* The bind and define handles for each transaction are
   included in their respective header files. */



/* for stock-level transaction */

extern int w_id;
extern int d_id;
extern int c_id;
extern int threshold;
extern int low_stock;

/* for delivery transaction */

extern int del_o_id[10];
extern int carrier_id;
extern int retries;

/* for order-status transaction */

extern int bylastname;
extern char c_last[17];
extern char c_first[17];
extern char c_middle[3];
extern double c_balance;
extern int o_id;
extern text o_entry_d[20];
extern int o_carrier_id;
extern int o_ol_cnt;
extern int ol_supply_w_id[15];
extern int ol_i_id[15];
extern int ol_quantity[15];
extern int ol_amount[15];
ub4    ol_del_len[15];
extern text ol_delivery_d[15][11];

/* for payment transaction */

extern int c_w_id;
extern int c_d_id;
extern int h_amount;
extern char w_street_1[21];
extern char w_street_2[21];
extern char w_city[21];
extern char w_state[3];
extern char w_zip[10];
extern char d_street_1[21];
extern char d_street_2[21];
extern char d_city[21];
extern char d_state[3];
extern char d_zip[10];
extern char c_street_1[21];
extern char c_street_2[21];
extern char c_city[21];
extern char c_state[3];
extern char c_zip[10];
extern char c_phone[17];
extern text c_since_d[11];
extern char c_credit[3];
extern int c_credit_lim;
extern float c_discount;
extern char c_data[201];
extern text h_date[20];

/* for new order transaction */

extern int nol_i_id[15];
extern int nol_supply_w_id[15];
extern int nol_quantity[15];
extern int nol_quanti10[15];
extern int nol_quanti91[15];
extern int nol_ytdqty[15];
extern int nol_amount[15];
extern int o_all_local;
extern float w_tax;
extern float d_tax;
extern float total_amount;
extern char i_name[15][25];
extern int i_name_strlen[15];
extern ub2 i_name_strlen_len[15];
extern ub2 i_name_strlen_rcode[15];
extern ub4 i_name_strlen_csize;
extern int s_quantity[15];
extern char brand_gen[15];
extern ub2 brand_gen_len[15];
extern ub2 brand_gen_rcode[15];
extern ub4 brand_gen_csize;
extern int i_price[15];
extern char brand_generic[15][1];
extern int status;
extern int tracelevel;

/* Miscellaneous */
extern OCIDate cr_date;
extern OCIDate c_since;
extern OCIDate o_entry_d_base;
extern OCIDate ol_d_base[15];

#ifndef DISCARD
# define DISCARD (void)
#endif

#ifndef sword
# define sword int
#endif

#define VER7            2

#define NA              -1      /* ANSI SQL NULL */
#define NLT             1       /* length for string null
terminator */
#define DEADLOCK        60      /* ORA-00060: deadlock */
#define NO_DATA_FOUND   1403    /* ORA-01403: no data found */
#define NOT_SERIALIZABLE  8177  /* ORA-08177: transaction not
serializable */
#define SNAPSHOT_TOO_OLD  1555  /* ORA-01555: snapshot too old */

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

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

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

#define min(x,y)  (((x) < (y)) ? (x) : (y))

#define OCIERROR(errp,function)\
        ocierror(__FILE__,__LINE__,(errp),(function));

#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
       ocierror(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)
); \
       ocierror(__FILE__,__LINE__, (errp), \
             OCIBindByName((stmp), &(bndp), (errp), \
                           (text *)(sqlvar), strlen((sqlvar)),\
                           (progv), (progvl),
(ftype),0,0,0,0,0,OCI_DEFAULT));

#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),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)
);\
       ocierror(__FILE__,__LINE__,(errp),\
         OCIBindByName((stmp),&(bndp),(errp),(text
*)(sqlvar),strlen((sqlvar)),\

(progv),(progvl),(ftype),(indp),(alen),(arcode),(ms),(cu),OCI_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
OCIDFNDYN(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,ctxp,cbf_data)
\
       ocierror(__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                                              (dvoid**)0));\
       ocierror(__FILE__,__LINE__,(errp), \
       OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),
(progvl),(ftype),\
                                     (indp),NULL,NULL,
OCI_DYNAMIC_FETCH));\
       ocierror(__FILE__,__LINE__,(errp), \
       OCIDefineDynamic((dfnp),(errp),(ctxp),(cbf_data)));



/* New order */

struct newinstruct {
   int w_id;
   int d_id;
   int c_id;
   int ol_i_id[15];
   int ol_supply_w_id[15];
   int ol_quantity[15];
};

struct newoutstruct {
   int terror;
   int o_id;
   int o_ol_cnt;
   char c_last[17];
   char c_credit[3];
   float c_discount;
   float w_tax;
   float d_tax;
   char o_entry_d[20];
   float total_amount;
   char i_name[15][25];
   int s_quantity[15];
   char brand_generic[15];
   float i_price[15];
   float ol_amount[15];
   char status[26];
   int retry;
};

struct newstruct {
   struct newinstruct newin;
   struct newoutstruct newout;
};


/* Payment */

struct payinstruct {
   int w_id;
   int d_id;
   int c_w_id;
   int c_d_id;
   int c_id;
   int bylastname;
   int h_amount;
   char c_last[17];
};

struct payoutstruct {
   int terror;
   char w_street_1[21];
   char w_street_2[21];
   char w_city[21];
   char w_state[3];
   char w_zip[10];
   char d_street_1[21];
   char d_street_2[21];
   char d_city[21];
   char d_state[3];
   char d_zip[10];
   int  c_id;
   char c_first[17];
   char c_middle[3];
   char c_last[17];
   char c_street_1[21];
   char c_street_2[21];
   char c_city[21];
   char c_state[3];
   char c_zip[10];
   char c_phone[17];
   char c_since[11];
   char c_credit[3];
   double c_credit_lim;
   float  c_discount;
   double c_balance;
   char c_data[201];
   char h_date[20];
   int retry;
};

struct paystruct {
   struct payinstruct payin;
   struct payoutstruct payout;
};


/* Order status */

struct ordinstruct {
   int w_id;
   int d_id;
   int c_id;
   int bylastname;
   char c_last[17];
};

struct ordoutstruct {
   int terror;
   int c_id;
   char c_last[17];
   char c_first[17];
   char c_middle[3];
   double c_balance;
   int o_id;
   char o_entry_d[20];
   int o_carrier_id;
   int o_ol_cnt;
   int ol_supply_w_id[15];
   int ol_i_id[15];
   int ol_quantity[15];
   float ol_amount[15];
   char ol_delivery_d[15][11];
   int retry;
};

struct ordstruct {
   struct ordinstruct ordin;
   struct ordoutstruct ordout;
};


/* Delivery */

struct delinstruct {
   int w_id;
   int o_carrier_id;
   double qtime;
   int in_timing_int;
};

struct deloutstruct {
   int terror;
   int retry;
};

struct delstruct {
   struct delinstruct delin;
   struct deloutstruct delout;
};


/* Stock level */

struct stoinstruct {
   int w_id;
   int d_id;
   int threshold;
};

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

```
};

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

#endif


--------------------------
tpccload.c
--------------------------
#ifdef RCSID
static char *RCSid =
   "$Header: tpccload.c 7030100.1 96/05/13 16:20:36 plai
Generic<base> $ Copyr (c) 1993 Oracle";
#endif /* RCSID */

/*=============================================================
=+
 |        Copyright (c) 1994  Oracle Corp, Redwood Shores, CA
|
 |                   OPEN SYSTEMS PERFORMANCE GROUP
|
 |                        All Rights Reserved
|

+=============================================================
+
 | FILENAME
 |    tpccload.c
 | DESCRIPTION
 |    Load or generate TPC-C database tables.
 |    Usage: tpccload -M <# of wares> [options]
 |               options: -A  load all tables
 |                        -w  load ware table
 |                        -d  load dist table
 |                        -c  load cust table
 |                        -i  load item table
 |                        -s  load stok table (cluster around
s_w_id)
 |                        -S  load stok table (cluster around
s_i_id)
 |                        -h  load hist table
 |                        -n  load new-order table
 |                        -o <oline file> load order and order-line
table
 |                        -b <ware#>  beginning ware number
 |                        -e <ware#>  ending ware number
 |                        -j <item#>  beginning item number (with -
S)
 |                        -k <item#>  ending item number (with -S)
 |                        -g  generate rows to standard output

+=============================================================
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include "tpcc.h"

#ifdef ORA_NT
#undef boolean
#include <process.h>
#include "dpbcore.h"
# define gettime dpbtimef
# define getcpu  dpbcpu
#define lrand48() ((long)rand() <<15 | rand())
#ifdef __STDC__
#  define PROTO(args)    args
#else
#  define PROTO(args)    ()
#endif
#endif

#define DISTARR 10            /* dist insert array size     */
#define CUSTARR 100           /* cust insert array size     */
#define STOCARR 100           /* stok insert array size     */
#define ITEMARR 100           /* item insert array size     */
#define HISTARR 100             /* hist insert array size    */
#define ORDEARR 100             /* order insert array size      */
#define NEWOARR 100             /* new order insert array size  */

#define DISTFAC 10            /* max. dist id     */
#define CUSTFAC 3000                    /* max. cust id     */
#define STOCFAC 100000                  /* max. stok id     */
#define ITEMFAC 100000                  /* max. item id
     */
#define HISTFAC 30000           /* history / warehouse  */
#define ORDEFAC 3000            /* order / district     */
#define NEWOFAC 900             /* new order / district */

#define C       0               /* constant in non-uniform dist.
eqt. */
#define CNUM1   1               /* first constant in non-uniform
dist. eqt. */
#define CNUM2   2               /* second constant in non-uniform
dist. eqt. */
#define CNUM3   3               /* third constant in non-uniform
dist. eqt. */

#define SEED    2               /* seed for random functions */

#define NOT_SERIALIZABLE  8177  /* ORA-08177: transaction not
serializable */
#define SNAPSHOT_TOO_OLD  1555  /* ORA-01555: snapshot too old */
#define RECOVERR -10
#define IRRECERR -20

#define SQLTXTW "INSERT INTO ware (w_id, w_ytd, w_tax, w_name,
w_street_1, w_street_2, w_city, w_state, w_zip) VALUES (:w_id,
30000000, :w_tax, :w_name, :w_street_1, \
   :w_street_2, :w_city, :w_state, :w_zip)"

#define SQLTXTD "INSERT INTO dist (d_id, d_w_id, d_ytd, d_tax,
d_next_o_id, d_name, d_street_1, d_street_2, d_city, d_state,
d_zip) VALUES (:d_id, :d_w_id,3000000, :d_tax, \
     3001, :d_name, :d_street_1, :d_street_2, :d_city, :d_state,
:d_zip)"

#define SQLTXTC "INSERT INTO cust (C_ID, C_D_ID, C_W_ID, C_FIRST,
C_MIDDLE, C_LAST, C_STREET_1, C_STREET_2, C_CITY, C_STATE, C_ZIP,
C_PHONE, C_SINCE, C_CREDIT, C_CREDIT_LIM, C_DISCOUNT, C_BALANCE,
C_YTD_PAYMENT, C_PAYMENT_CNT, C_DELIVERY_CNT, C_DATA) VALUES
(:c_id, :c_d_id, :c_w_id, \
   :c_first, 'OE', :c_last, :c_street_1, :c_street_2, :c_city,
:c_state, \
   :c_zip, :c_phone, SYSDATE, :c_credit, 5000000, :c_discount, -
1000, 1000, 1, \
   0, :c_data)"

#define SQLTXTH "INSERT INTO hist (h_c_id, h_c_d_id, h_c_w_id,
h_d_id, h_w_id, h_date, h_amount, h_data) VALUES (:h_c_id,
:h_c_d_id, :h_c_w_id, \
   :h_d_id, :h_w_id, SYSDATE, 1000, :h_data)"

#define SQLTXTS "INSERT INTO stok (s_i_id, s_w_id,
s_quantity,s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05 ,
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10, s_ytd,
s_order_cnt, s_remote_cnt, s_data) \
VALUES (:s_i_id, :s_w_id, :s_quantity, \
   :s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05,
:s_dist_06, \
   :s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10, 0, 0, 0,
:s_data)" \

#define SQLTXTI "INSERT INTO item
(I_ID,I_IM_ID,I_NAME,I_PRICE,I_DATA) VALUES (:i_id, :i_im_id,
:i_name, :i_price, \
   :i_data)"

#define SQLTXTO1 "INSERT INTO ordr (O_ID,
O_D_ID,O_W_ID,O_C_ID,O_ENTRY_D,O_CARRIER_ID,O_OL_CNT,O_ALL_LOCAL) \
   VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
   SYSDATE, :o_carrier_id, :o_ol_cnt, 1)"

#define SQLTXTO2 "INSERT INTO ordr (O_ID,
O_D_ID,O_W_ID,O_C_ID,O_ENTRY_D,O_CARRIER_ID,O_OL_CNT,O_ALL_LOCAL) \
   VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
   SYSDATE, 11, :o_ol_cnt, 1)"

#define SQLTXTOL1 "INSERT INTO ordl (OL_O_ID, OL_D_ID, OL_W_ID,
OL_NUMBER, OL_DELIVERY_D, OL_I_ID, OL_SUPPLY_W_ID, OL_QUANTITY,
OL_AMOUNT, OL_DIST_INFO) \
   VALUES (:ol_o_id, :ol_d_id, \
   :ol_w_id, :ol_number, SYSDATE, :ol_i_id, :ol_supply_w_id, 5, 0,
\
   :ol_dist_info)"

#define SQLTXTOL2 "INSERT INTO ordl (OL_O_ID, OL_D_ID, OL_W_ID,
OL_NUMBER, OL_DELIVERY_D, OL_I_ID, OL_SUPPLY_W_ID, OL_QUANTITY,
OL_AMOUNT, OL_DIST_INFO) \
   VALUES (:ol_o_id, :ol_d_id, \
   :ol_w_id, :ol_number, to_date('01-Jan-1811'), :ol_i_id,
:ol_supply_w_id, 5, :ol_amount, \
   :ol_dist_info)"

#define SQLTXTNO "INSERT INTO nord (no_o_id, no_d_id, no_w_id)
VALUES (:no_o_id, :no_d_id, :no_w_id)"

#define OCIERROR(errp,function)\
        ocierror(__FILE__,__LINE__,(errp),(function));

#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
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));

static char *lastname[] = {
   "BAR",
```

```
   "OUGHT",
   "ABLE",
   "PRI",
   "PRES",
   "ESE",
   "ANTI",
   "CALLY",
   "ATION",
   "EING"
};

char num9[10];
char num16[17];
char str2[3];
char str24[15][25];
int randperm3000[3000];

void initperm();
void randstr();
void randdatastr();
void randnum();
void randlastname (char*, int);
int NURand();
void sysdate();

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

OCIStmt *curw;
OCIStmt *curd;
OCIStmt *curc;
OCIStmt *curh;
OCIStmt *curs;
OCIStmt *curi;
OCIStmt *curo1;
OCIStmt *curo2;
OCIStmt *curol1;
OCIStmt *curol2;
OCIStmt *curno;

OCIBind *w_id_bp = (OCIBind *) 0;
OCIBind *w_name_bp = (OCIBind *) 0;
OCIBind *w_street1_bp = (OCIBind *) 0;
OCIBind *w_street2_bp = (OCIBind *) 0;
OCIBind *w_city_bp = (OCIBind *) 0;
OCIBind *w_state_bp = (OCIBind *) 0;
OCIBind *w_zip_bp = (OCIBind *) 0;
OCIBind *w_tax_bp = (OCIBind *) 0;

OCIBind *d_id_bp = (OCIBind *) 0;
OCIBind *d_w_id_bp = (OCIBind *) 0;
OCIBind *d_name_bp = (OCIBind *) 0;
OCIBind *d_street1_bp = (OCIBind *) 0;
OCIBind *d_street2_bp = (OCIBind *) 0;
OCIBind *d_city_bp = (OCIBind *) 0;
OCIBind *d_state_bp = (OCIBind *) 0;
OCIBind *d_zip_bp = (OCIBind *) 0;
OCIBind *d_tax_bp = (OCIBind *) 0;

OCIBind *c_id_bp = (OCIBind *) 0;
OCIBind *c_d_id_bp = (OCIBind *) 0;
OCIBind *c_w_id_bp = (OCIBind *) 0;
OCIBind *c_first_bp = (OCIBind *) 0;
OCIBind *c_last_bp = (OCIBind *) 0;
OCIBind *c_street1_bp = (OCIBind *) 0;
OCIBind *c_street2_bp = (OCIBind *) 0;
OCIBind *c_city_bp = (OCIBind *) 0;
OCIBind *c_state_bp = (OCIBind *) 0;
OCIBind *c_zip_bp = (OCIBind *) 0;
OCIBind *c_phone_bp = (OCIBind *) 0;
OCIBind *c_discount_bp = (OCIBind *) 0;
OCIBind *c_credit_bp = (OCIBind *) 0;
OCIBind *c_data_bp = (OCIBind *) 0;

OCIBind *i_id_bp = (OCIBind *) 0;
OCIBind *i_im_id_bp = (OCIBind *) 0;
OCIBind *i_name_bp = (OCIBind *) 0;
OCIBind *i_price_bp = (OCIBind *) 0;
OCIBind *i_data_bp = (OCIBind *) 0;

OCIBind *s_i_id_bp = (OCIBind *) 0;
OCIBind *s_w_id_bp = (OCIBind *) 0;
OCIBind *s_quantity_bp = (OCIBind *) 0;
OCIBind *s_dist_01_bp = (OCIBind *) 0;
OCIBind *s_dist_02_bp = (OCIBind *) 0;
OCIBind *s_dist_03_bp = (OCIBind *) 0;
OCIBind *s_dist_04_bp = (OCIBind *) 0;
OCIBind *s_dist_05_bp = (OCIBind *) 0;
OCIBind *s_dist_06_bp = (OCIBind *) 0;
OCIBind *s_dist_07_bp = (OCIBind *) 0;
OCIBind *s_dist_08_bp = (OCIBind *) 0;
OCIBind *s_dist_09_bp = (OCIBind *) 0;
OCIBind *s_dist_10_bp = (OCIBind *) 0;
OCIBind *s_data_bp = (OCIBind *) 0;

OCIBind *h_c_id_bp = (OCIBind *) 0;
OCIBind *h_c_d_id_bp = (OCIBind *) 0;
OCIBind *h_c_w_id_bp = (OCIBind *) 0;
OCIBind *h_d_id_bp = (OCIBind *) 0;
OCIBind *h_w_id_bp = (OCIBind *) 0;
OCIBind *h_data_bp = (OCIBind *) 0;

OCIBind *ol_o_id_bp = (OCIBind *) 0;
OCIBind *ol_d_id_bp = (OCIBind *) 0;
OCIBind *ol_w_id_bp = (OCIBind *) 0;
OCIBind *ol_i_id_bp = (OCIBind *) 0;
OCIBind *ol_number_bp = (OCIBind *) 0;
OCIBind *ol_supply_w_id_bp = (OCIBind *) 0;
OCIBind *ol_dist_info_bp = (OCIBind *) 0;
OCIBind *ol_amount_bp = (OCIBind *) 0;

OCIBind *o_id_bp = (OCIBind *) 0;
OCIBind *o_d_id_bp = (OCIBind *) 0;
OCIBind *o_w_id_bp = (OCIBind *) 0;
OCIBind *o_c_id_bp = (OCIBind *) 0;
OCIBind *o_carrier_id_bp = (OCIBind *) 0;
OCIBind *o_ol_cnt_bp  = (OCIBind *) 0;

OCIBind *no_o_id_bp  = (OCIBind *) 0;
OCIBind *no_d_id_bp  = (OCIBind *) 0;
OCIBind *no_w_id_bp  = (OCIBind *) 0;

void myusage()
{
   fprintf (stderr, "\n");
   fprintf (stderr, "Usage:\ttpccload -M <multiplier>
[options]\n");
   fprintf (stderr, "options:\n");
   fprintf (stderr, "\t-A :\tload all tables\n");
   fprintf (stderr, "\t-w :\tload ware table\n");
   fprintf (stderr, "\t-d :\tload dist table\n");
   fprintf (stderr, "\t-c :\tload cust table\n");
   fprintf (stderr, "\t-i :\tload item table\n");
   fprintf (stderr, "\t-s :\tload stok table (cluster around
s_w_id)\n");
   fprintf (stderr, "\t-S :\tload stok table (cluster around
s_i_id)\n");
   fprintf (stderr, "\t-h :\tload hist table\n");
   fprintf (stderr, "\t-n :\tload new-order table\n");
   fprintf (stderr, "\t-o <oline file> :\tload order and order-line
table\n");
   fprintf (stderr, "\t-b <ware#> :\tbeginning ware number\n");
   fprintf (stderr, "\t-e <ware#> :\tending ware number\n");
   fprintf (stderr, "\t-j <item#> :\tbeginning item number (with -
S)\n");
   fprintf (stderr, "\t-k <item#> :\tending item number (with -
S)\n");
   fprintf (stderr, "\t-g :\tgenerate rows to standard output\n");
   fprintf (stderr, "\n");
   exit(1);
}

void quit()
{
   OCIERROR(errhp,OCISessionEnd ( tpcsvc,errhp, tpcusr,
OCI_DEFAULT));
   OCIERROR(errhp,OCIServerDetach ( tpcsrv, errhp, OCI_DEFAULT));
   OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
   OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
   OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
   OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
   OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);
}

void main (argc, argv)
int argc;
char *argv[];
{
   char *uid="tpcc";
   char *pwd="tpcc";
   int scale=0;
   int i, j;
   int loop;
   int loopcount;
   int cid;
   int dwid;
   int cdid;
   int cwid;
   int sid;
   int swid;
   int olcnt;
   int nrows;
   int row;

   int w_id;
   char w_name[11];
   char w_street_1[21];
   char w_street_2[21];
   char w_city[21];
   char w_state[2];
   char w_zip[9];
   float w_tax;

   int d_id[10];
   int d_w_id[10];
   char d_name[10][11];
   char d_street_1[10][21];
   char d_street_2[10][21];
   char d_city[10][21];
   char d_state[10][2];
   char d_zip[10][9];
   float d_tax[10];

   int c_id[100];
   int c_d_id[100];
   int c_w_id[100];
   char c_first[100][17];
   char c_last[100][17];
   char c_street_1[100][21];
   char c_street_2[100][21];
   char c_city[100][21];
   char c_state[100][2];
   char c_zip[100][9];
```

```
   char c_phone[100][16];
   char c_credit[100][2];
   float c_discount[100];
   char c_data[100][501];

   int i_id[100];
   int i_im_id[100];
   int i_price[100];
   char i_name[100][25];
   char i_data[100][51];

   int s_i_id[100];
   int s_w_id[100];
   int s_quantity[100];
   char s_dist_01[100][24];
   char s_dist_02[100][24];
   char s_dist_03[100][24];
   char s_dist_04[100][24];
   char s_dist_05[100][24];
   char s_dist_06[100][24];
   char s_dist_07[100][24];
   char s_dist_08[100][24];
   char s_dist_09[100][24];
   char s_dist_10[100][24];
   char s_data[100][51];

   int h_w_id[100];
   int h_d_id[100];
   int h_c_id[100];
   char h_data[100][25];

   int o_id[100];
   int o_d_id[100];
   int o_w_id[100];
   int o_c_id[100];
   int o_carrier_id[100];
   int o_ol_cnt[100];

   int ol_o_id[1500];
   int ol_d_id[1500];
   int ol_w_id[1500];
   int ol_number[1500];
   int ol_i_id[1500];
   int ol_supply_w_id[1500];
   int ol_amount[1500];
   char ol_dist_info[1500][24];

   int no_o_id[100];
   int no_d_id[100];
   int no_w_id[100];

   char sdate[30];

#ifdef ORA_NT
   clock_t begin_time, end_time;
   clock_t begin_cpu, end_cpu;

   char *arg_ptr,  **end_args;
#else
   double begin_time, end_time;
   double begin_cpu, end_cpu;
   double gettime(), getcpu();

   extern int getopt();
   extern char *optarg;
   extern int optind, opterr;
   int opt;
#endif

   char   *argstr="M:AwdcisShno:b:e:j:k:g";
   int do_A=0;
   int do_w=0;
   int do_d=0;
   int do_i=0;
   int do_c=0;
   int do_s=0;
   int do_S=0;
   int do_h=0;
   int do_o=0;
   int do_n=0;
   int gen=0;
   int bware=1;
   int eware=0;
   int bitem=1;
   int eitem=0;

   FILE *olfp=NULL;
   char olfname[100];
   int status;
#ifdef ORA_NT
   char fname[100];
   FILE *logfile;
#endif /* ORA_NT */

/*---------------------------------------------------------------+
 | Parse command line -- look for scale factor.
          |
 +---------------------------------------------------------------*/

   if (argc == 1) {
      myusage ();
   }

#ifdef ORA_NT
   end_args = argv + argc;
   for (++argv; argv < end_args; )
   {
      arg_ptr = *argv++;
          if (*arg_ptr != '-')
          {
             myusage ();
          } else
          {
         switch (arg_ptr[1]) {
          case '?': myusage ();
                    break;
          case 'M': scale = atoi (*argv++);
                    break;
          case 'A': do_A = 1;
                    break;
          case 'w': do_w = 1;
                    break;
          case 'd': do_d = 1;
                    break;
          case 'c': do_c = 1;
                    break;
          case 'i': do_i = 1;
                    break;
          case 's': do_s = 1;
                    break;
          case 'S': do_S = 1;
                    break;
          case 'h': do_h = 1;
                    break;
          case 'n': do_n = 1;
                    break;
          case 'o': do_o = 1;
                    strcpy (olfname, *argv++);
                    break;
          case 'b': bware = atoi (*argv++);
                    break;
          case 'e': eware = atoi (*argv++);
                    break;
          case 'j': bitem = atoi (*argv++);
                    break;
          case 'k': eitem = atoi (*argv++);
                    break;
          case 'g': gen = 1;
                    strcpy (fname, *argv++);
                    break;
          case 'l': logfile=fopen(*argv++,"w");
                    break;
          default: fprintf (stderr, "THIS SHOULD NEVER
HAPPEN!!!\n");
                   fprintf (stderr, "(reached default case in getopt
())\n");
                   myusage ();
         }
          }
   }

#else

   while ((opt = getopt (argc, argv, argstr)) != -1) {
      switch (opt) {
         case '?': myusage ();
                   break;
         case 'M': scale = atoi (optarg);
                   break;
         case 'A': do_A = 1;
                   break;
         case 'w': do_w = 1;
                   break;
         case 'd': do_d = 1;
                   break;
         case 'c': do_c = 1;
                   break;
         case 'i': do_i = 1;
                   break;
         case 's': do_s = 1;
                   break;
         case 'S': do_S = 1;
                   break;
         case 'h': do_h = 1;
                   break;
         case 'n': do_n = 1;
                   break;
         case 'o': do_o = 1;
                   strcpy (olfname, optarg);
                   break;
         case 'b': bware = atoi (optarg);
                   break;
         case 'e': eware = atoi (optarg);
                   break;
         case 'j': bitem = atoi (optarg);
                   break;
         case 'k': eitem = atoi (optarg);
                   break;
         case 'g': gen = 1;
                   break;
         default: fprintf (stderr, "THIS SHOULD NEVER
HAPPEN!!!\n");
                   fprintf (stderr, "(reached default case in getopt
())\n");
                   myusage ();
      }
   }

# endif /* ORA_NT */

/*---------------------------------------------------------------*|
|         Rudimentary error checking
|         |
|*---------------------------------------------------------------*/
```

```
   if (scale  < 1) {
      fprintf (stderr, "Invalid scale factor: '%d'\n", scale);
      myusage ();
   }

   if (!(do_A || do_w || do_d || do_c || do_i || do_s || do_S ||
do_h || do_o ||
         do_n)) {
      fprintf (stderr, "What should I load???\n");
      myusage ();
   }

   if (gen && (do_A || (do_w + do_d + do_c + do_i + do_s + do_S +
do_h + do_o +
                        do_n > 1))) {
      fprintf (stderr, "Can only generate table one at a time\n");
      myusage ();
   }

   if (do_S && (do_A || do_s)) {
      fprintf (stderr, "Cluster stock table around s_w_id or
s_i_id?\n");
      myusage ();
   }

   if (eware <= 0)
      eware = scale;
   if (eitem <= 0)
      eitem = STOCFAC;

   if (do_S) {
      if ((bitem  < 1) || (bitem > STOCFAC)) {
         fprintf (stderr, "Invalid beginning item number: '%d'\n",
bitem);
         myusage ();
      }

      if ((eitem  < bitem) || (eitem > STOCFAC)) {
         fprintf (stderr, "Invalid ending item number: '%d'\n",
eitem);
         myusage ();
      }
   }

   if ((bware  < 1) || (bware > scale)) {
      fprintf (stderr, "Invalid beginning warehouse number:
'%d'\n", bware);
      myusage ();
   }

   if ((eware  < bware) || (eware > scale)) {
      fprintf (stderr, "Invalid ending warehouse number: '%d'\n",
eware);
      myusage ();
   }

   if (gen && do_o) {
      if ((olfp = fopen (olfname, "w")) == NULL) {
         fprintf (stderr, "Can't open '%s' for writing order
lines\n", olfname);
         myusage ();
      }
   }

/*---------------------------------------------------------------+
 | Prepare to insert into database.
          |
 +---------------------------------------------------------------*/

   sysdate (sdate);
   if (!gen) {

      /* log on to Oracle */

      OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0,0);
      OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
      OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv,
OCI_HTYPE_SERVER, 0 , (dvoid **)0);
      OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp,
OCI_HTYPE_ERROR, 0 , (dvoid **)0);
      OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc,
OCI_HTYPE_SVCCTX, 0 , (dvoid **)0);
      OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
      OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid
*)tpcsrv,
                 (ub4)0,OCI_ATTR_SERVER, errhp);
      OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr,
OCI_HTYPE_SESSION, 0 , (dvoid **)0);
      OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
                 (ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
      OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd,
(ub4)strlen(pwd),
              OCI_ATTR_PASSWORD, errhp);
      OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr,
OCI_CRED_RDBMS, OCI_DEFAULT));

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

      fprintf (stderr, "\nConnected to Oracle userid '%s/%s'.\n",
uid, pwd);

      /* open cursors and parse statement */
      if (do_A || do_w)  {
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curw),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIStmtPrepare(curw, errhp, (text
*)SQLTXTW,
                     strlen((char *)SQLTXTW), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
      }

      if (do_A || do_d) {
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curd),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIStmtPrepare(curd, errhp, (text
*)SQLTXTD,
                     strlen((char *)SQLTXTD), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
      }

      if (do_A || do_c) {
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curc),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIStmtPrepare(curc, errhp, (text
*)SQLTXTC,
                     strlen((char *)SQLTXTC), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
      }

      if (do_A || do_h) {
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curh),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIStmtPrepare(curh, errhp, (text
*)SQLTXTH,
                     strlen((char *)SQLTXTH), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
      }

      if (do_A || do_s || do_S) {
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curs),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIStmtPrepare(curs, errhp, (text
*)SQLTXTS,
                     strlen((char *)SQLTXTS), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
      }

      if (do_A || do_i) {
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curi),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIStmtPrepare(curi, errhp, (text
*)SQLTXTI,
                     strlen((char *)SQLTXTI), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
      }

      if (do_A || do_o) {
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curo1),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curo2),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curol1),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curol2),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIStmtPrepare(curo1, errhp, (text
*)SQLTXTO1,
                     strlen((char *)SQLTXTO1), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
         OCIERROR(errhp,OCIStmtPrepare(curo2, errhp, (text
*)SQLTXTO2,
                     strlen((char *)SQLTXTO2), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
         OCIERROR(errhp,OCIStmtPrepare(curol1, errhp, (text
*)SQLTXTOL1,
                     strlen((char *)SQLTXTOL1), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
         OCIERROR(errhp,OCIStmtPrepare(curol2, errhp, (text
*)SQLTXTOL2,
                     strlen((char *)SQLTXTOL2), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
      }

      if (do_A || do_n) {
         OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curno),
OCI_HTYPE_STMT, 0, (dvoid**)0));
         OCIERROR(errhp,OCIStmtPrepare(curno, errhp, (text
*)SQLTXTNO,
                     strlen((char *)SQLTXTNO), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
      }

      /* bind variables */

      /* warehouse */

      if (do_A || do_w) {
         OCIERROR(errhp, OCIBindByName(curw, &w_id_bp, errhp, (text
*)(":w_id"), strlen((":w_id")),
                     (ub1 *)&(w_id), sizeof(w_id), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
                     (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

         OCIERROR(errhp, OCIBindByName(curw, &w_name_bp,
errhp,(text *)":w_name", strlen(":w_name"),
                     (ub1 *)w_name, 11, SQLT_STR, (dvoid *) 0,
(ub2 *)0, (ub2 *)0,
                     (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

         OCIERROR(errhp, OCIBindByName(curw, &w_street1_bp, errhp,
(text *)":w_street_1",
                        strlen(":w_street_1"), (ub1 *)w_street_1,
21, SQLT_STR,
```

```
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curw, &w_street2_bp, errhp,
(text *)":w_street_2",
                        strlen(":w_street_2"), (ub1 *)w_street_2,
21, SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curw, &w_city_bp, errhp,
(text *)":w_city",
                        strlen(":w_city"), (ub1 *)w_city, 21,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curw, &w_state_bp, errhp,
(text *)":w_state",
                        strlen(":w_state"), (ub1 *)w_state, 2,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curw, &w_zip_bp, errhp,
(text *)":w_zip",
                        strlen(":w_zip"), (ub1 *)w_zip, 9,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curw, &w_tax_bp, errhp,
(text *)":w_tax",
                        strlen(":w_tax"), (ub1 *) & w_tax,
sizeof(w_tax), SQLT_FLT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* district */

    if (do_A || do_d) {
        OCIERROR(errhp, OCIBindByName(curd, &d_id_bp, errhp, (text
*)":d_id",
                        strlen(":d_id"), (ub1 *)d_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curd, &d_w_id_bp, errhp,
(text *)":d_w_id",
                        strlen(":d_w_id"), (ub1 *)d_w_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curd, &d_name_bp, errhp,
(text *)":d_name",
                        strlen(":d_name"), (ub1 *)d_name, 11,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curd, &d_street1_bp, errhp,
(text *)":d_street_1",
                        strlen(":d_street_1"), (ub1 *)d_street_1,
21, SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curd, &d_street2_bp, errhp,
(text *)":d_street_2",
                        strlen(":d_street_2"), (ub1 *)d_street_2,
21, SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curd, &d_city_bp, errhp,
(text *)":d_city",
                        strlen(":d_city"), (ub1 *)d_city, 21,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curd, &d_state_bp, errhp,
(text *)":d_state",
                        strlen(":d_state"), (ub1 *)d_state, 2,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curd, &d_zip_bp, errhp,
(text *)":d_zip",
                        strlen(":d_zip"), (ub1 *)d_zip, 9, SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curd, &d_tax_bp, errhp,
(text *)":d_tax",
                        strlen(":d_tax"), (ub1 *)d_tax,
sizeof(float), SQLT_FLT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* customer */

    if (do_A || do_c) {
        OCIERROR(errhp, OCIBindByName(curc, &c_id_bp, errhp, (text
*)":c_id",
                        strlen(":c_id"), (ub1 *)c_id, sizeof(int),
SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_d_id_bp, errhp,
(text *)":c_d_id",
                        strlen(":c_d_id"), (ub1 *)c_d_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_w_id_bp, errhp,
(text *)":c_w_id",
                        strlen(":c_w_id"), (ub1 *)c_w_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_first_bp, errhp,
(text *)":c_first",
                            strlen(":c_first"), (ub1 *)c_first, 17,
SQLT_STR,
                            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_last_bp, errhp,
(text *)":c_last",
                            strlen(":c_last"), (ub1 *)c_last, 17,
SQLT_STR,
                            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_street1_bp, errhp,
(text *)":c_street_1",
                        strlen(":c_street_1"), (ub1 *)c_street_1, 21,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_street2_bp, errhp,
(text *)":c_street_2",
                        strlen(":c_street_2"), (ub1 *)c_street_2, 21,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_city_bp, errhp,
(text *)":c_city",
                        strlen(":c_city"), (ub1 *)c_city, 21,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_state_bp, errhp,
(text *)":c_state",
                        strlen(":c_state"), (ub1 *)c_state, 2,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_zip_bp, errhp,
(text *)":c_zip",
                        strlen(":c_zip"), (ub1 *)c_zip, 9, SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_phone_bp, errhp,
(text *)":c_phone",
                        strlen(":c_phone"), (ub1 *)c_phone, 16,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_credit_bp, errhp,
(text *)":c_credit",
                        strlen(":c_credit"), (ub1 *)c_credit, 2,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_discount_bp, errhp,
(text *)":c_discount",
                        strlen(":c_discount"), (ub1 *)c_discount,
sizeof(float), SQLT_FLT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curc, &c_data_bp, errhp,
(text *)":c_data",
                        strlen(":c_data"), (ub1 *)c_data, 501,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* item */

    if (do_A || do_i) {
        OCIERROR(errhp, OCIBindByName(curi, &i_id_bp, errhp, (text
*)":i_id",
                        strlen(":i_id"), (ub1 *)i_id, sizeof(int),
SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
```

```
        OCIERROR(errhp, OCIBindByName(curi, &i_im_id_bp, errhp,
(text *)":i_im_id",
                        strlen(":i_im_id"), (ub1 *)i_im_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curi, &i_name_bp, errhp,
(text *)":i_name",
                        strlen(":i_name"), (ub1 *)i_name, 25,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curi, &i_price_bp, errhp,
(text *)":i_price",
                        strlen(":i_price"), (ub1 *)i_price,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curi, &i_data_bp, errhp,
(text *)":i_data",
                        strlen(":i_data"), (ub1 *)i_data, 51,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
      }

      /* stock */

      if (do_A || do_s || do_S) {
        OCIERROR(errhp, OCIBindByName(curs, &s_i_id_bp, errhp,
(text *)":s_i_id",
                        strlen(":s_i_id"), (ub1 *)s_i_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curs, &s_w_id_bp, errhp,
(text *)":s_w_id",
                        strlen(":s_w_id"), (ub1 *)s_w_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curs, &s_quantity_bp, errhp,
(text *)":s_quantity",
                        strlen(":s_quantity"), (ub1 *)s_quantity,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curs, &s_dist_01_bp, errhp,
(text *)":s_dist_01",
                        strlen(":s_dist_01"), (ub1 *)s_dist_01, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_02_bp, errhp,
(text *)":s_dist_02",
                        strlen(":s_dist_02"), (ub1 *)s_dist_02, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_03_bp, errhp,
(text *)":s_dist_03",
                        strlen(":s_dist_03"), (ub1 *)s_dist_03, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_04_bp, errhp,
(text *)":s_dist_04",
                        strlen(":s_dist_04"), (ub1 *)s_dist_04, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_05_bp, errhp,
(text *)":s_dist_05",
                        strlen(":s_dist_05"), (ub1 *)s_dist_05, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_06_bp, errhp,
(text *)":s_dist_06",
                        strlen(":s_dist_06"), (ub1 *)s_dist_06, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_07_bp, errhp,
(text *)":s_dist_07",
                        strlen(":s_dist_07"), (ub1 *)s_dist_07, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_08_bp, errhp,
(text *)":s_dist_08",
                        strlen(":s_dist_08"), (ub1 *)s_dist_08, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_09_bp, errhp,
(text *)":s_dist_09",
                        strlen(":s_dist_09"), (ub1 *)s_dist_09, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_10_bp, errhp,
(text *)":s_dist_10",
                        strlen(":s_dist_10"), (ub1 *)s_dist_10, 24,
SQLT_CHR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curs, &s_data_bp, errhp,
(text *)":s_data",
                        strlen(":s_data"), (ub1 *)s_data, 51,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
      }

      /* history */

      if (do_A || do_h) {
        OCIERROR(errhp, OCIBindByName(curh, &h_c_id_bp, errhp,
(text *)":h_c_id",
                        strlen(":h_c_id"), (ub1 *)h_c_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_c_d_id_bp, errhp,
(text *)":h_c_d_id",
                        strlen(":h_c_d_id"), (ub1 *)h_d_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_c_w_id_bp, errhp,
(text *)":h_c_w_id",
                        strlen(":h_c_w_id"), (ub1 *)h_w_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_d_id_bp, errhp,
(text *)":h_d_id",
                        strlen(":h_d_id"), (ub1 *)h_d_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_w_id_bp, errhp,
(text *)":h_w_id",
                        strlen(":h_w_id"), (ub1 *)h_w_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_data_bp, errhp,
(text *)":h_data",
                        strlen(":h_data"), (ub1 *)h_data, 25,
SQLT_STR,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      }

      /* order_line (delivered) */

      if (do_A || do_o) {

        OCIERROR(errhp, OCIBindByName(curol1, &ol_o_id_bp, errhp,
(text *)":ol_o_id",
                        strlen(":ol_o_id"), (ub1 *)ol_o_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol1, &ol_d_id_bp, errhp,
(text *)":ol_d_id",
                        strlen(":ol_d_id"), (ub1 *)ol_d_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol1, &ol_w_id_bp, errhp,
(text *)":ol_w_id",
                        strlen(":ol_w_id"), (ub1 *)ol_w_id,
sizeof(int), SQLT_INT,
                        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol1, &ol_number_bp,
errhp, (text *)":ol_number",
                        strlen(":ol_number"), (ub1 *)ol_number,
sizeof(int), SQLT_INT,
```

```
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol1, &ol_i_id_bp, errhp,
(text *)":ol_i_id",
                    strlen(":ol_i_id"), (ub1 *)ol_i_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol1, &ol_supply_w_id_bp,
errhp, (text *)":ol_supply_w_id",
                    strlen(":ol_supply_w_id"), (ub1
*)ol_supply_w_id, sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol1, &ol_dist_info_bp,
errhp, (text *)":ol_dist_info",
                    strlen(":ol_dist_info"), (ub1 *)ol_dist_info,
24, SQLT_CHR,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

     /* order_line (not delivered) */

        OCIERROR(errhp, OCIBindByName(curol2, &ol_o_id_bp, errhp,
(text *)":ol_o_id",
                    strlen(":ol_o_id"), (ub1 *)ol_o_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol2, &ol_d_id_bp, errhp,
(text *)":ol_d_id",
                    strlen(":ol_d_id"), (ub1 *)ol_d_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol2, &ol_w_id_bp, errhp,
(text *)":ol_w_id",
                    strlen(":ol_w_id"), (ub1 *)ol_w_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol2, &ol_number_bp,
errhp, (text *)":ol_number",
                    strlen(":ol_number"), (ub1 *)ol_number,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol2, &ol_i_id_bp, errhp,
(text *)":ol_i_id",
                    strlen(":ol_i_id"), (ub1 *)ol_i_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol2, &ol_supply_w_id_bp,
errhp, (text *)":ol_supply_w_id",
                    strlen(":ol_supply_w_id"), (ub1
*)ol_supply_w_id, sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol2, &ol_dist_info_bp,
errhp, (text *)":ol_dist_info",
                    strlen(":ol_dist_info"), (ub1 *)ol_dist_info,
24, SQLT_CHR,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curol2, &ol_amount_bp,
errhp, (text *)":ol_amount",
                    strlen(":ol_amount"), (ub1 *)ol_amount,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

     /* orders (delivered) */

        OCIERROR(errhp, OCIBindByName(curo1, &o_id_bp, errhp,
(text *)":o_id",
                    strlen(":o_id"), (ub1 *)o_id, sizeof(int),
SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_d_id_bp, errhp,
(text *)":o_d_id",
                    strlen(":o_d_id"), (ub1 *)o_d_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_w_id_bp, errhp,
(text *)":o_w_id",
                    strlen(":o_w_id"), (ub1 *)o_w_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_c_id_bp, errhp,
(text *)":o_c_id",
                    strlen(":o_c_id"), (ub1 *)o_c_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_carrier_id_bp,
errhp, (text *)":o_carrier_id",
                    strlen(":o_carrier_id"), (ub1 *)o_carrier_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_ol_cnt_bp, errhp,
(text *)":o_ol_cnt",
                    strlen(":o_ol_cnt"), (ub1 *)o_ol_cnt,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

     /* orders (not delivered) */

        OCIERROR(errhp, OCIBindByName(curo2, &o_id_bp, errhp,
(text *)":o_id",
                    strlen(":o_id"), (ub1 *)o_id, sizeof(int),
SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo2, &o_d_id_bp, errhp,
(text *)":o_d_id",
                    strlen(":o_d_id"), (ub1 *)o_d_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo2, &o_w_id_bp, errhp,
(text *)":o_w_id",
                    strlen(":o_w_id"), (ub1 *)o_w_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo2, &o_c_id_bp, errhp,
(text *)":o_c_id",
                    strlen(":o_c_id"), (ub1 *)o_c_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo2, &o_ol_cnt_bp, errhp,
(text *)":o_ol_cnt",
                    strlen(":o_ol_cnt"), (ub1 *)o_ol_cnt,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
     }

     /* new order */

     if (do_A || do_n) {
        OCIERROR(errhp, OCIBindByName(curno, &no_o_id_bp, errhp,
(text *)":no_o_id",
                    strlen(":no_o_id"), (ub1 *)no_o_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curno, &no_d_id_bp, errhp,
(text *)":no_d_id",
                    strlen(":no_d_id"), (ub1 *)no_d_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curno, &no_w_id_bp, errhp,
(text *)":no_w_id",
                    strlen(":no_w_id"), (ub1 *)no_w_id,
sizeof(int), SQLT_INT,
                    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
     }
   }

/*---------------------------------------------------------------+
 | Initialize random number generator
         |
 +---------------------------------------------------------------*/

   srand (SEED);
#ifndef ORA_NT
   srand48 (SEED);
#endif
   initperm ();


/*---------------------------------------------------------------+
 | Load the WAREHOUSE table.
         |
 +---------------------------------------------------------------*/

   if (do_A || do_w) {
      nrows = eware - bware + 1;

      fprintf (stderr, "Loading/generating warehouse: w%d - w%d (%d
rows)\n",
               bware, eware, nrows);

      begin_time = gettime ();
```

```
      begin_cpu = getcpu ();

      for (loop = bware; loop <= eware; loop++) {

         w_tax = (float) ((lrand48 () % 2001) * 0.0001);
         randstr (w_name, 6, 10);
         randstr (w_street_1, 10, 20);
         randstr (w_street_2, 10, 20);
         randstr (w_city, 10, 20);
         randstr (str2, 2, 2);
         randnum (num9, 9);
         num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

         if (gen) {
            printf ("%d 30000000 %6.4f %s %s %s %s %s %s\n", loop,
w_tax,
                    w_name, w_street_1, w_street_2, w_city, str2,
num9);
            fflush (stdout);
         }
         else {
            w_id = loop;
            strncpy (w_state, str2, 2);
            strncpy (w_zip, num9, 9);

              status = OCIStmtExecute(tpcsvc, curw, errhp, (ub4) 1,
(ub4) 0,
                        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                        (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
               fprintf (stderr, "Error at ware %d\n", loop);
               OCIERROR(errhp, status);
               quit ();
             exit (1);
            }
         }
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | Load the DISTRICT table.
         |
 +---------------------------------------------------------------*/

   if (do_A || do_d) {
      nrows = (eware - bware + 1) * DISTFAC;

      fprintf (stderr, "Loading/generating district: w%d - w%d (%d
rows)\n",
               bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      dwid = bware - 1;

      for (row = 0; row < nrows; ) {
         dwid++;

         for (i = 0; i < DISTARR; i++, row++) {
            d_tax[i] = (float) ((lrand48 () % 2001) * 0.0001);
            randstr (d_name[i], 6, 10);
            randstr (d_street_1[i], 10, 20);
            randstr (d_street_2[i], 10, 20);
            randstr (d_city[i], 10, 20);
            randstr (str2, 2, 2);
            randnum (num9, 9);
            num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

            if (gen) {
               printf ("%d %d 3000000 %6.4f 3001 %s %s %s %s %s
%s\n",
                       i + 1, dwid, d_tax[i], d_name[i],
d_street_1[i],
                           d_street_2[i], d_city[i], str2, num9 );
            }
            else {
               d_id[i] = i + 1;
               d_w_id[i] = dwid;
               strncpy (d_state[i], str2, 2);
               strncpy (d_zip[i], num9, 9);
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            status = OCIStmtExecute(tpcsvc, curd, errhp, (ub4)
DISTARR, (ub4) 0,
                        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                        (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                   fprintf (stderr, "Aborted at ware %d, dist
1\n", dwid);
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
         }
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | Load the CUSTOMER table.
         |
 +---------------------------------------------------------------*/

   if (do_A || do_c) {
      nrows = (eware - bware + 1) * CUSTFAC * DISTFAC;

      fprintf (stderr, "Loading/generating customer: w%d - w%d (%d
rows)\n   ",
               bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      cid = 0;
      cdid = 1;
      cwid = bware;
      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < CUSTARR; i++, row++) {
            cid++;
            if (cid > CUSTFAC) {         /* cycle cust id */
               cid = 1;                  /* cheap mod */
               cdid++;                   /* shift dist cycle */
               if (cdid > DISTFAC) {
                  cdid = 1;
                  cwid++;                /* shift ware cycle */
               }
            }
            c_id[i] = cid;
            c_d_id[i] = cdid;
            c_w_id[i] = cwid;
            if (cid <= 1000)
               randlastname (c_last[i], cid - 1);
            else
               randlastname (c_last[i], NURand (255, 0, 999,
CNUM1));
            c_credit[i][1] = 'C';
            if (lrand48 () % 10)
               c_credit[i][0] = 'G';
            else
               c_credit[i][0] = 'B';
            c_discount[i] = (float)((lrand48 () % 5001) * 0.0001);
            randstr (c_first[i], 8, 16);
            randstr (c_street_1[i], 10, 20);
            randstr (c_street_2[i], 10, 20);
            randstr (c_city[i], 10, 20);
            randstr (str2, 2, 2);
            randnum (num9, 9);
            num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
            randnum (num16, 16);
            randstr (c_data[i], 300, 500);

            if (gen) {
               printf ("%d %d %d %s OE %s %s %s %s %s %s %s %cC
5000000 %6.4f -1000 1000 1 0 %s\n",
                        cid, cdid, cwid, c_first[i], c_last[i],
                        c_street_1[i], c_street_2[i], c_city[i],
str2, num9,
                        num16, sdate, c_credit[i][0], c_discount[i],
c_data[i]);
            }
            else {
               strncpy (c_state[i], str2, 2);
               strncpy (c_zip[i], num9, 9);
               strncpy (c_phone[i], num16, 16);
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            status = OCIStmtExecute(tpcsvc, curc, errhp, (ub4)
CUSTARR, (ub4) 0,
                         (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                         (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);

            if (status != OCI_SUCCESS) {
                      fprintf (stderr, "Aborted at w_id %d, d_id
%d, c_id %d\n",
                           c_w_id[0], c_d_id[0], c_id[0]);
                   OCIERROR(errhp, status);
                   quit ();
                   exit (1);
            }
         }

         if ((++loopcount) % 50)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
```

```
   }


/*---------------------------------------------------------------+
 | Load the ITEM table.
                  |
 +---------------------------------------------------------------*/

   if (do_A || do_i) {
      nrows = ITEMFAC;

      fprintf (stderr, "Loading/generating item: (%d rows)\n   ",
nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < ITEMARR; i++, row++) {
            i_im_id[i] = (lrand48 () % 10000) + 1;
            i_price[i] = ((lrand48 () % 9901) + 100);
            randstr (i_name[i], 14, 24);
            randdatastr (i_data[i], 26, 50);

            if (gen) {
               printf ("%d %d %s %d %s\n", row + 1, i_im_id[i],
i_name[i],
                       i_price[i], i_data[i]);
            }
            else {
               i_id[i] = row + 1;
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            status = OCIStmtExecute(tpcsvc, curi, errhp, (ub4)
ITEMARR, (ub4) 0,
                        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                        (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                    fprintf (stderr, "Aborted at i_id %d\n",
i_id[0]);
                  OCIERROR(errhp, status);
                  quit ();
                  exit (1);
            }
         }

         if ((++loopcount) % 50)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }


/*---------------------------------------------------------------+
 | Load the STOCK table.
          |
 +---------------------------------------------------------------*/

   if (do_A || do_s) {

      nrows = (eware - bware + 1) * STOCFAC;

      fprintf (stderr, "Loading/generating stock: w%d - w%d (%d
rows)\n   ",
               bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      sid = 0;
      swid = bware;
      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < STOCARR; i++, row++) {
            if (++sid > STOCFAC) {        /* cheap mod */
               sid = 1;
               swid++;
            }
            s_quantity[i] = (lrand48 () % 91) + 10;
            randstr (str24[0], 24, 24);
            randstr (str24[1], 24, 24);
            randstr (str24[2], 24, 24);
            randstr (str24[3], 24, 24);
            randstr (str24[4], 24, 24);
            randstr (str24[5], 24, 24);
            randstr (str24[6], 24, 24);
            randstr (str24[7], 24, 24);
            randstr (str24[8], 24, 24);
            randstr (str24[9], 24, 24);
            randdatastr (s_data[i], 26, 50);

            if (gen) {
               printf ("%d %d %d %s %s %s %s %s %s %s %s %s %s 0 0
0 %s\n",
                       sid, swid, s_quantity[i], str24[0],
str24[1], str24[2],
                       str24[3], str24[4], str24[5], str24[6],
str24[7],
                       str24[8], str24[9], s_data[i]);
            }
            else {
               s_i_id[i] = sid;
               s_w_id[i] = swid;
               strncpy (s_dist_01[i], str24[0], 24);
               strncpy (s_dist_02[i], str24[1], 24);
               strncpy (s_dist_03[i], str24[2], 24);
               strncpy (s_dist_04[i], str24[3], 24);
               strncpy (s_dist_05[i], str24[4], 24);
               strncpy (s_dist_06[i], str24[5], 24);
               strncpy (s_dist_07[i], str24[6], 24);
               strncpy (s_dist_08[i], str24[7], 24);
               strncpy (s_dist_09[i], str24[8], 24);
               strncpy (s_dist_10[i], str24[9], 24);
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            status = OCIStmtExecute(tpcsvc, curs, errhp, (ub4)
STOCARR, (ub4) 0,
                        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                        (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                    fprintf (stderr, "Aborted at w_id %d, s_i_id
%d\n", s_w_id[0], s_i_id[0]);
                  OCIERROR(errhp, status);
                  quit ();
                  exit (1);
            }
         }

         if ((++loopcount) % 50)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | Load the STOCK table (cluster around s_i_id).            |
 +---------------------------------------------------------------*/

   if (do_S) {

      nrows = (eitem - bitem + 1) * (eware - bware + 1);

      fprintf (stderr, "Loading/generating stock: i%d - i%d, w%d -
w%d (%d rows)\n   ",
               bitem, eitem, bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      sid = bitem;
      swid = bware - 1;
      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < STOCARR; i++, row++) {
            if (++swid > eware) {          /* cheap mod */
               swid = bware;
               sid++;
            }
            s_quantity[i] = (lrand48 () % 91) + 10;
            randstr (str24[0], 24, 24);
            randstr (str24[1], 24, 24);
            randstr (str24[2], 24, 24);
            randstr (str24[3], 24, 24);
            randstr (str24[4], 24, 24);
            randstr (str24[5], 24, 24);
            randstr (str24[6], 24, 24);
            randstr (str24[7], 24, 24);
            randstr (str24[8], 24, 24);
            randstr (str24[9], 24, 24);
            randdatastr (s_data[i], 26, 50);

            if (gen) {
               printf ("%d %d %d %s %s %s %s %s %s %s %s %s %s 0 0
0 %s\n",
                       sid, swid, s_quantity[i], str24[0],
str24[1], str24[2],
                       str24[3], str24[4], str24[5], str24[6],
str24[7],
                       str24[8], str24[9], s_data[i]);
            }
            else {
               s_i_id[i] = sid;
               s_w_id[i] = swid;
               strncpy (s_dist_01[i], str24[0], 24);
               strncpy (s_dist_02[i], str24[1], 24);
               strncpy (s_dist_03[i], str24[2], 24);
               strncpy (s_dist_04[i], str24[3], 24);
```

```
               strncpy (s_dist_05[i], str24[4], 24);
               strncpy (s_dist_06[i], str24[5], 24);
               strncpy (s_dist_07[i], str24[6], 24);
               strncpy (s_dist_08[i], str24[7], 24);
               strncpy (s_dist_09[i], str24[8], 24);
               strncpy (s_dist_10[i], str24[9], 24);
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            status = OCIStmtExecute(tpcsvc, curs, errhp, (ub4)
STOCARR, (ub4) 0,
                        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                        (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                  fprintf (stderr, "Aborted at w_id %d, s_i_id
%d\n", s_w_id[0], s_i_id[0]);
                  OCIERROR(errhp, status);
                  quit ();
                  exit (1);
            }
         }

         if ((++loopcount) % 50)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*-------------------------------------------------------------+
 | Load the HISTORY table.
          |
 +-------------------------------------------------------------*/

   if (do_A || do_h) {
      nrows = (eware - bware + 1) * HISTFAC;

      fprintf (stderr, "Loading/generating history: w%d - w%d (%d
rows)\n   ",
               bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      cid = 0;
      cdid = 1;
      cwid = bware;
      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < HISTARR; i++, row++) {
            cid++;
            if (cid > CUSTFAC) {        /* cycle cust id */
               cid = 1;                 /* cheap mod */
               cdid++;                  /* shift district cycle */
               if (cdid > DISTFAC) {
                  cdid = 1;
                  cwid++;               /* shift warehouse cycle */
               }
            }
            h_c_id[i] = cid;
            h_d_id[i] = cdid;
            h_w_id[i] = cwid;
            randstr (h_data[i], 12, 24);
            if (gen) {
               printf ("%d %d %d %d %d %s 1000 %s\n", cid, cdid,
cwid, cdid,
                       cwid, sdate, h_data[i]);
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            status = OCIStmtExecute(tpcsvc, curh, errhp, (ub4)
HISTARR, (ub4) 0,
                        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                        (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                       fprintf (stderr, "Aborted at w_id %d, d_id
%d, c_id %d\n",
                        h_w_id[0], h_d_id[0], h_c_id[0]);
                  OCIERROR(errhp, status);
                  quit ();
                  exit (1);
            }
         }

         if ((++loopcount) % 50)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*-------------------------------------------------------------+
 | Load the ORDERS and ORDER-LINE table.
          |
 +-------------------------------------------------------------*/

   if (do_A || do_o) {

      int batch_olcnt;

      nrows = (eware - bware + 1) * ORDEFAC * DISTFAC;

      fprintf (stderr, "Loading/generating orders and order-line:
w%d - w%d (%d ord, ~%d ordl)\n   ",
               bware, eware, nrows, nrows * 10);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      cid = 0;
      cdid = 1;
      cwid = bware;
      loopcount = 0;

      for (row = 0; row < nrows; ) {

         batch_olcnt = 0;

         for (i = 0; i < ORDEARR; i++, row++) {
            cid++;
            if (cid > ORDEFAC) {        /* cycle cust id */
               cid = 1;                 /* cheap mod */
               cdid++;                  /* shift district cycle */
               if (cdid > DISTFAC) {
                  cdid = 1;
                  cwid++;               /* shift warehouse cycle */
               }
            }
            o_carrier_id[i] = lrand48 () % 10 + 1;
            o_ol_cnt[i] = olcnt = lrand48 () % 11 + 5;

            if (gen) {
               if (cid < 2101) {
                  printf ("%d %d %d %d %s %d %d 1\n", cid, cdid,
cwid,
                          randperm3000[cid - 1],
sdate,o_carrier_id[i],
                          o_ol_cnt[i]);
               }
               else {
                    /* set carrierid to 11 instead of null */
                  printf ("%d %d %d %d %s 11 %d 1\n", cid, cdid,
cwid,
                          randperm3000[cid - 1], sdate,
o_ol_cnt[i]);
               }
            }
            else {
               o_id[i] = cid;
               o_d_id[i] = cdid;
               o_w_id[i] = cwid;
               o_c_id[i] = randperm3000[cid - 1];
            }

            for (j = 0; j < o_ol_cnt[i]; j++, batch_olcnt++ ) {
               ol_i_id[batch_olcnt] = sid = lrand48 () % 100000 +
1;
               if (cid < 2101)
                  ol_amount[batch_olcnt] = 0;
               else
                  ol_amount[batch_olcnt] = (lrand48 () % 999999 +
1) ;
               randstr (str24[j], 24, 24);

               if (gen) {
                  if (cid < 2101) {
                     fprintf (olfp, "%d %d %d %d %s %d %d 5 %ld
%s\n", cid,
                              cdid, cwid, j + 1, sdate,
ol_i_id[batch_olcnt], cwid,
                              ol_amount[batch_olcnt], str24[j]);
                  }
                  else {
                     /* Insert a default date instead of null date
*/
                     fprintf (olfp, "%d %d %d %d 01-Jan-1811 %d %d
5 %ld %s\n", cid,
                              cdid, cwid, j + 1,
ol_i_id[batch_olcnt], cwid,
                              ol_amount[batch_olcnt], str24[j]);
                  }
               }
               else {
                  ol_o_id[batch_olcnt] = cid;
                  ol_d_id[batch_olcnt] = cdid;
                  ol_w_id[batch_olcnt] = cwid;
                  ol_number[batch_olcnt] = j + 1;
                  ol_supply_w_id[batch_olcnt] = cwid;
                  strncpy (ol_dist_info[batch_olcnt], str24[j],
24);
               }
            }
         }
```

```
         if (gen) {
            fflush (olfp);
            fflush (stdout);
         }
         else {
            if (cid < 2101) {
               status = OCIStmtExecute(tpcsvc, curo1, errhp, (ub4)
ORDEARR, (ub4) 0,
                                       (CONST OCISnapshot*) 0,
(OCISnapshot*) 0,
                                       (ub4) OCI_DEFAULT);
               if (status != OCI_SUCCESS) {
                           fprintf (stderr, "Aborted at w_id %d,
d_id %d, o_id %d\n", cwid, cdid, cid);
                     OCIERROR(errhp, status);
                     quit ();
                     exit (1);
               }

               status = OCIStmtExecute(tpcsvc, curol1, errhp,
(ub4) batch_olcnt, (ub4) 0,
                                       (CONST OCISnapshot*) 0,
(OCISnapshot*) 0,
                                       (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
               if (status != OCI_SUCCESS) {
                     fprintf (stderr, "Aborted at w_id %d, d_id
%d, o_id %d\n", cwid, cdid, cid);
                     OCIERROR(errhp, status);
                     quit ();
                     exit (1);
               }
            }
            else {
               OCIERROR(errhp, OCIStmtExecute(tpcsvc, curo2,
errhp, (ub4) ORDEARR, (ub4) 0,
                                       (CONST OCISnapshot*) 0,
(OCISnapshot*) 0,
                                       (ub4) OCI_DEFAULT));
               if (status != OCI_SUCCESS) {
                     fprintf (stderr, "Aborted at w_id %d, d_id
%d, o_id %d\n", cwid, cdid, cid);
                     OCIERROR(errhp, status);
                     quit ();
                     exit (1);
               }

               OCIERROR(errhp, OCIStmtExecute(tpcsvc, curol2,
errhp, (ub4) batch_olcnt, (ub4) 0,
                                       (CONST OCISnapshot*) 0,
(OCISnapshot*) 0,
                                       (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS));
               if (status != OCI_SUCCESS) {
                     fprintf (stderr, "Aborted at w_id %d, d_id
%d, o_id %d\n", cwid, cdid, cid);
                     OCIERROR(errhp, status);
                     quit ();
                     exit (1);
               }
            }
         }

         if ((++loopcount) % 50)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d orders committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d orders loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | Load the NEW-ORDER table.
         |
 +---------------------------------------------------------------*/

   if (do_A || do_n) {
      nrows = (eware - bware + 1) * NEWOFAC * DISTFAC;

      fprintf (stderr, "Loading/generating new-order: w%d - w%d (%d
rows)\n   ",
               bware, eware, nrows);

      begin_time = gettime ();
      begin_cpu = getcpu ();

      cid = 0;
      cdid = 1;
      cwid = bware;
      loopcount = 0;

      for (row = 0; row < nrows; ) {
         for (i = 0; i < NEWOARR; i++, row++) {
            cid++;
            if (cid > NEWOFAC) {
               cid = 1;
               cdid++;
               if (cdid > DISTFAC) {
                  cdid = 1;
                  cwid++;
               }
            }
            if (gen) {
               printf ("%d %d %d\n", cid + 2100, cdid, cwid);
            }
            else {
               no_o_id[i] = cid + 2100;
               no_d_id[i] = cdid;
               no_w_id[i] = cwid;
            }
         }

         if (gen) {
            fflush (stdout);
         }
         else {
            status = OCIStmtExecute(tpcsvc, curno, errhp, (ub4)
NEWOARR, (ub4) 0,
                                    (CONST OCISnapshot*) 0,
(OCISnapshot*) 0,
                                    (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                  fprintf (stderr, "Aborted at w_id %d, d_id %d,
o_id %d\n", cwid, cdid, cid + 2100);
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
         }

         if ((++loopcount) % 45)
            fprintf (stderr, ".");
         else
            fprintf (stderr, " %d rows committed\n   ", row);
      }

      end_time = gettime ();
      end_cpu = getcpu ();
      fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
               nrows, end_time - begin_time, end_cpu - begin_cpu);
   }

/*---------------------------------------------------------------+
 | clean up and exit.
                  |
 +---------------------------------------------------------------*/

   if (olfp)
      fclose (olfp);
   if (!gen)
      quit ();
   exit (0);

}

void initperm ()
{
   int i;
   int pos;
   int temp;

   /* init randperm3000 */

   for (i = 0; i < 3000; i++)
      randperm3000[i] = i + 1;
   for (i = 3000; i > 0; i--) {
      pos = lrand48 () % i;
      temp = randperm3000[i - 1];
      randperm3000[i - 1] = randperm3000[pos];
      randperm3000[pos] = temp;
   }
}

void randstr (str, x, y)
char *str;
int x;
int y;
{
   int i, j;
   int len;

   len = (lrand48 () % (y - x + 1)) + x;
   for (i = 0; i < len; i++) {
      j = lrand48 () % 62;
      if (j < 26)
         str[i] = (char) (j + 'a');
      else if (j < 52)
         str[i] = (char) (j - 26 + 'A');
      else
         str[i] = (char) (j - 52 + '0');
   }
   str[len] = '\0';

}

void randdatastr (str, x, y)
char *str;
int x;
int y;
{
   int i, j;
   int len;
   int pos;

   len = (lrand48 () % (y - x + 1)) + x;
   for (i = 0; i < len; i++) {
      j = lrand48 () % 62;
      if (j < 26)
```

```
         str[i] = (char) (j + 'a');
      else if (j < 52)
        str[i] = (char) (j - 26 + 'A');
      else
         str[i] = (char) (j - 52 + '0');
   }
   str[len] = '\0';
   if ((lrand48 () % 10) == 0) {
      pos = (lrand48 () % (len - 8));
      str[pos] = 'O';
      str[pos + 1] = 'R';
      str[pos + 2] = 'I';
      str[pos + 3] = 'G';
      str[pos + 4] = 'I';
      str[pos + 5] = 'N';
      str[pos + 6] = 'A';
      str[pos + 7] = 'L';
   }
}

void randnum (str, len)
char *str;
int len;
{
   int i;

   for (i = 0; i < len; i++)
      str[i] = (char) (lrand48 () % 10 + '0');
   str[len] = '\0';

}

void randlastname (str, id)
char *str;
int id;
{
   id = id % 1000;
   strcpy (str, lastname[id / 100]);
   strcat (str, lastname[(id / 10) % 10]);
   strcat (str, lastname[id % 10]);
}

int NURand (A, x, y, cnum)
int A, x, y, cnum;
{
   int a, b;

   a = lrand48 () % (A + 1);
   b = (lrand48 () % (y - x + 1)) + x;
   return ((((a | b) + cnum) % (y - x + 1)) + x);

}

void sysdate (sdate)
char *sdate;
{
   time_t tp;
   struct tm *tmptr;

   time (&tp);
   tmptr = localtime (&tp);
   strftime (sdate, 29, "%d-%b-%Y", tmptr);
}

int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
  text errbuf[512];
  sb4 errcode;
  sb4 lstat;
  ub4 recno=2;

  switch (status) {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_SUCCESS_WITH_INFO\n");
    lstat = OCIErrorGet (errhp, recno++, (text *) NULL, &errcode,
errbuf,
                         (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    fprintf(stderr,"Error - %s\n", errbuf);
    break;
  case OCI_NEED_DATA:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_NEED_DATA\n");
    return (IRRECERR);
  case OCI_NO_DATA:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_NO_DATA\n");
    return (IRRECERR);
  case OCI_ERROR:
    lstat = OCIErrorGet (errhp, (ub4) 1,
                         (text *) NULL, &errcode, errbuf,
                               (ub4) sizeof(errbuf),
OCI_HTYPE_ERROR);
    if (errcode == NOT_SERIALIZABLE) return (errcode);
    if (errcode == SNAPSHOT_TOO_OLD) return (errcode);
    while (lstat != OCI_NO_DATA)
    {
      fprintf(stderr,"Module %s Line %d\n", fname, lineno);
      fprintf(stderr,"Error - %s\n", errbuf);
      lstat = OCIErrorGet (errhp, recno++, (text *) NULL, &errcode,
errbuf,
                         (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    }
    return (errcode);
  case OCI_INVALID_HANDLE:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_INVALID_HANDLE\n");
    exit(-1);
  case OCI_STILL_EXECUTING:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_STILL_EXECUTE\n");
    return (IRRECERR);
  case OCI_CONTINUE:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_CONTINUE\n");
    return (IRRECERR);
  default:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Status - %s\n", status);
    return (IRRECERR);
  }
  return (RECOVERR);
}



--------------------------
makefile
--------------------------
#
#         Copyright (c) 1997  Oracle Corp, Redwood Shores, CA
#                             All Rights Reserved

# FILE:        Makefile
# DESCRIPTION: Builds TPCC Load/Test executables
# CREATED:     02 July 97
# MODIFIED:
#  rmook  02/04/98  added perf counter stuff
#  jdavison         2/4/98              changes for ops
#  rmook  11/20/97  setup enhancements
#  rmook  11/20/97  8.0.4 edits (oci.lib)
#

CPU                           = i386

#rkambo
# !IF "$(CFG)" != "TUXEDO"  && "$(CFG)" != "TOPEND"
# !MESSAGE Invalid configuration "$(CFG)" specified.
# !MESSAGE You can specify a configuration when running NMAKE
# !MESSAGE by defining the macro CFG on the command line. For
example:
# !MESSAGE
# !MESSAGE NMAKE /f "Makefile" CFG="TUXEDO"
# !MESSAGE
# !MESSAGE Possible choices for configuration are:
# !MESSAGE
# !MESSAGE TUXEDO - for tuxedo build
# !MESSAGE TOPEND - for topend build
# !MESSAGE
# !ERROR An invalid configuration is specified.
# !ENDIF
#rkambo

#rkambo
TPCCBENCH           = c:\benchrun
TPCCBIN   = $(TPCCBENCH)\bin
ORACLE_HOME         = c:\oracle\ora90
CLIB_DIR  = c:\program files\devstudio\vc\lib
#rkambo

SOURCE_DIR          = $(TPCCBENCH)\source\server

OCI_SUB             = oci
OCI_INCLUDE         = $(ORACLE_HOME)\$(OCI_SUB)\include
OCI_LIB_HOME        = $(ORACLE_HOME)\$(OCI_SUB)\lib\msvc
#rkambo
# OCI_LIB           = $(OCI_LIB_HOME)\ora803.lib
OCI_LIB             = $(OCI_LIB_HOME)\Oci.lib
#rkambo

DPB_LIB_DIR         = ..\lib
DPB_LIB             = $(DPB_LIB_DIR)\dpbnt.lib

!IF  "$(CFG)" == "TUXEDO"
COMMON_LIB_DIR      = C:\inetpub\wwwroot
COMMON_LIB          = $(COMMON_LIB_DIR)\common_files.lib

!ELSE
COMMON_LIB_DIR      = C:\inetpub\wwwroot
#rkambo
# COMMON_LIB        =
COMMON_LIB          = common_files.lib
#rkambo

!ENDIF

# Visual C++ libraries
SOCKETS_LIB         = "$(CLIB_DIR)\wsock32.lib"
ADVAPI_LIB          = "$(CLIB_DIR)\advapi32.lib"

CL                  = cl
CFLAGS              = -c -W3 -D_X86_=1
COPT                = -Ox -Oy- -Gf
#CDEBUG             = -Od -Z7 -Ge

!IF  "$(CFG)" == "TUXEDO"
TM_CFLAGS = /DTUX /D_CONSOLE /DWIN32 /D_TMSTHREADS
/IC:\TUXEDO\include
!ENDIF
```

```
!IF  "$(CFG)" == "TOPEND"
TM_CFLAGS = /DTOP
!ENDIF

#rkambo
TM_CFLAGS =
#rkambo

ORA_DEFINES         = /DORACLE /DOCI /DORA_NT
ORA_INCLUDES        = /I$(OCI_INCLUDE) /I$(DPB_LIB_DIR)

CL_OPS              = $(CFLAGS) $(COPT) $(CDEBUG) $(ORA_DEFINES)
$(TM_CFLAGS) $(ORA_INCLUDES)

LINK                = link
#LFLAGS             = /debug:full /debugtype:cv /pdb:none

LIBRARIAN = lib

#
#         Private C compilation rule.
#
.c.obj:
          @$(CL) $(CL_OPS) /Fo$@ $*.c

#
#         Destination for executables.
#
BIN_DIR             = ..\..\bin

#
#         Rule for building an "exe" in $(BIN_DIR) and using object
files in current
#         directory.
#
{}.obj{$(BIN_DIR)}.exe:
          @$(LINK) $(LFLAGS) /OUT:$@ $(DPB_LIB) $(OCI_LIB) $**

all:      $(COMMON_LIB)
          \
          $(DPB_LIB)                              \
                    $(BIN_DIR)\tpccload.exe        \
                    $(BIN_DIR)\tpcc.exe            \
                    $(BIN_DIR)\getrand.exe         \
                    $(BIN_DIR)\90per.exe           \
                    $(BIN_DIR)\runtpb.exe          \
                    $(BIN_DIR)\ntstat.exe    \
                    $(BIN_DIR)\single_txn.exe \
                    $(BIN_DIR)\sleep.exe \
                    $(BIN_DIR)\press_return.exe \
               $(BIN_DIR)\runid.exe

c_trnsnt.lib: plnew.obj plpay.obj plord.obj pldel.obj plsto.obj
          @$(LIBRARIAN) /out:$@ $**

$(DPB_LIB):
           cd $(DPB_LIB_DIR)
#rkambo
#         $(MAKE) -f makefile.intel )
         $(MAKE) -f Makefile.nt
#rkambo
         cd $(SOURCE_DIR)

$(COMMON_LIB): plnew.obj plpay.obj plord.obj pldel.obj plsto.obj \
 errrpt.obj report.obj tpccpl.obj
        @$(LIBRARIAN) /out:$@ $**

press_return.obj: press_return.c

runid.obj: runid.c

tpccload.obj: tpccload.c tpcc.h

c_drv_o7.obj:  c_drv_o7.c tpcc.h tpcc_info.h results.h perf.h

perf.obj:   perf.c perf.h

single_txn.obj: single_txn.c tpcc.h tpcc_info.h results.h

sleep.obj:  sleep.c

c_dump.obj:  c_dump.c tpcc.h tpcc_info.h

test_drv.obj:  test_drv.c tpcc.h tpcc_info.h

test_trn.obj:  test_trn.c tpcc.h tpcc_info.h

tpccpl.obj: tpccpl.c tpcc.h tpcc_info.h tpccpl.h

plnew.obj: plnew.c tpcc.h tpccpl.h
plpay.obj: plpay.c tpcc.h tpccpl.h
plord.obj: plord.c tpcc.h tpccpl.h
pldel.obj: pldel.c tpcc.h tpccpl.h
plsto.obj: plsto.c tpcc.h tpccpl.h

report.obj: report.c results.h

errrpt.obj: errrpt.c

#tpccsvr.obj:  tpccsrv.c tpcc.h tpcc_info.h

getrand.obj:  getrand.c

90per.obj:  90per.c

runtpb.obj:  runtpb.c

$(BIN_DIR)\tpccload.exe: tpccload.obj

$(BIN_DIR)\test_drv.exe: test_drv.obj c_drv_o7.obj c_dump.obj
errrpt.obj perf.obj \

$(SOCKETS_LIB)

$(BIN_DIR)\test_trn.exe: test_trn.obj c_trnsnt.lib $(ADVAPI_LIB)
tpccpl.obj c_dump.obj \

errrpt.obj

$(BIN_DIR)\tpcc.exe: c_drv_o7.obj c_trnsnt.lib tpccpl.obj
c_dump.obj report.obj  perf.obj \
                                   errrpt.obj $(DPB_LIB) $(OCI_LIB)
$(SOCKETS_LIB) $(ADVAPI_LIB)
          @$(LINK) $(LFLAGS) /OUT:$@ $**

$(BIN_DIR)\single_txn.exe: single_txn.obj c_trnsnt.lib tpccpl.obj
c_dump.obj \
                    report.obj errrpt.obj $(DPB_LIB) $(OCI_LIB)
$(SOCKETS_LIB) $(ADVAPI_LIB)
          @$(LINK) $(LFLAGS) /OUT:$@ $**

$(BIN_DIR)\getrand.exe:  getrand.obj

$(BIN_DIR)\90per.exe:  90per.obj

$(BIN_DIR)\runtpb.exe:  runtpb.obj

$(BIN_DIR)\sleep.exe:  sleep.obj

$(BIN_DIR)\press_return.exe: press_return.obj

ntstat.obj: ntstat.c counter.h counters.h
          @$(CL) $(CL_OPS) /DORACLE_PROCESS /Fo$@ $*.c

ins_stat.obj: ins_stat.c counter.h

$(BIN_DIR)\ntstat.exe: ntstat.obj ins_stat.obj $(ADVAPI_LIB)

clean:
          @for %f in ( *.obj *.lib $(BIN_DIR)\*.exe ) do if exist
%f del %f
          @for %f in ( *.obj *.lib ) do if exist $(DPB_LIB_DIR)\%f
del $(DPB_LIB_DIR)\%f
```

Appendix C: Tunable Parameters

Windows 2000 Advanced Server and Oracle 9i Enterprise Edition (Server)

```
--------------------------
boot.ini on all 8 servers
--------------------------

[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Microsoft Windows 2000
Advanced Serve
r" /fastdetect /3GB /PAE
C:\CMDCONS\BOOTSECT.DAT="Microsoft Windows 2000 Recovery Console"
/cmdcons


--------------------------
Windows 2000 Advanced
Server Registry
--------------------------

Key Name:          SOFTWARE\oracle\HOME0
Class Name:        Application Global Data
Last Write Time:   2/7/2002 - 12:49 PM

Value 13
  Name:            oracle_priority
  Type:            REG_SZ
  Data:
LMON:HIGH;LMD0:HIGH;LMS0:HIGH;DBW0:HIGH;DBW1:HIGH;LGWR:HIGH;LCK0:HI
GH;CKPT:HIGH




Key Name:          SOFTWARE\oracle\osd9i
Class Name:        <NO CLASS>
Last Write Time:   1/7/2002 - 12:19 PM

Value 7
  Name:            PrivateNodeNames
  Type:            REG_MULTI_SZ
  Data:            rac1
                   rac2
                   rac3
                   rac4
                   rac5
                   rac6
                   rac7
                   rac8
Value 8
  Name:            PublicNodeNames
  Type:            REG_MULTI_SZ
  Data:            ra1
                   ra2
                   ra3
                   ra4
                   ra5
                   ra6
                   ra7
                   ra8


--------------------------
init_1.ora (Server 1)
--------------------------
instance_number     =1
thread    =1
rollback_segments =  (
t_1_1,t_1_2,t_1_3,t_1_4,t_1_5,t_1_6,t_1_7,t_1_8,t_1_9,t_1_10,t_1_11
,t_1_12,t_1_13,t_1_14,t_1_15,t_1_16,t_1_17,t_1_18,t_1_19,t_1_20,
t_1_21,t_1_22,t_1_23,t_1_24,t_1_25,t_1_26,t_1_27,t_1_28,t_1_29,t_1_
30,t_1_31,t_1_32,t_1_33,t_1_34,t_1_35,t_1_36,t_1_37,t_1_38,t_1_39,t
_1_40,
t_1_41,t_1_42,t_1_43,t_1_44,t_1_45,t_1_46,t_1_47,t_1_48,t_1_49,t_1_
50,t_1_51,t_1_52,t_1_53,t_1_54,t_1_55,t_1_56,t_1_57,t_1_58,t_1_59,t
_1_60,t_1_61,t_1_62,t_1_63,t_1_64,t_1_65,t_1_66,t_1_67,t_1_68,t_1_6
9,t_1_70,t_1_71,t_1_72,t_1_73,t_1_74,t_1_75,t_1_76,t_1_77,t_1_78,t_
1_79,t_1_80,
t_1_81,t_1_82,t_1_83,t_1_84,t_1_85,t_1_86,t_1_87,t_1_88,t_1_89,t_1_
90,t_1_91,t_1_92,t_1_93,t_1_94,t_1_95,t_1_96,t_1_97,t_1_98,t_1_100)
ifile=p_run.ora


--------------------------
init_2.ora (Server 2)
--------------------------
instance_number     =2
thread    =2
rollback_segments =  (
t_2_1,t_2_2,t_2_3,t_2_4,t_2_5,t_2_6,t_2_7,t_2_8,t_2_9,t_2_10,t_2_11
,t_2_12,t_2_13,t_2_14,t_2_15,t_2_16,t_2_17,t_2_18,t_2_19,t_2_20,
t_2_21,t_2_22,t_2_23,t_2_24,t_2_25,t_2_26,t_2_27,t_2_28,t_2_29,t_2_
30,t_2_31,t_2_32,t_2_33,t_2_34,t_2_35,t_2_36,t_2_37,t_2_38,t_2_39,t
_2_40,
t_2_41,t_2_42,t_2_43,t_2_44,t_2_45,t_2_46,t_2_47,t_2_48,t_2_49,t_2_
50,t_2_51,t_2_52,t_2_53,t_2_54,t_2_55,t_2_56,t_2_57,t_2_58,t_2_59,t
_2_60,t_2_61,t_2_62,t_2_63,t_2_64,t_2_65,t_2_66,t_2_67,t_2_68,t_2_6
9,t_2_70,t_2_71,t_2_72,t_2_73,t_2_74,t_2_75,t_2_76,t_2_77,t_2_78,t_
2_79,t_2_80,
t_2_81,t_2_82,t_2_83,t_2_84,t_2_85,t_2_86,t_2_87,t_2_88,t_2_89,t_2_
90,t_2_91,t_2_92,t_2_93,t_2_94,t_2_95,t_2_96,t_2_97,t_2_98,t_2_100)
ifile=p_run.ora


--------------------------
init_3.ora (Server 3)
--------------------------
instance_number     =3
thread    =3
rollback_segments =  (
t_3_1,t_3_2,t_3_3,t_3_4,t_3_5,t_3_6,t_3_7,t_3_8,t_3_9,t_3_10,t_3_11
,t_3_12,t_3_13,t_3_14,t_3_15,t_3_16,t_3_17,t_3_18,t_3_19,t_3_20,
t_3_21,t_3_22,t_3_23,t_3_24,t_3_25,t_3_26,t_3_27,t_3_28,t_3_29,t_3_
30,t_3_31,t_3_32,t_3_33,t_3_34,t_3_35,t_3_36,t_3_37,t_3_38,t_3_39,t
_3_40,
t_3_41,t_3_42,t_3_43,t_3_44,t_3_45,t_3_46,t_3_47,t_3_48,t_3_49,t_3_
50,t_3_51,t_3_52,t_3_53,t_3_54,t_3_55,t_3_56,t_3_57,t_3_58,t_3_59,t
_3_60,t_3_61,t_3_62,t_3_63,t_3_64,t_3_65,t_3_66,t_3_67,t_3_68,t_3_6
9,t_3_70,t_3_71,t_3_72,t_3_73,t_3_74,t_3_75,t_3_76,t_3_77,t_3_78,t_
3_79,t_3_80,
t_3_81,t_3_82,t_3_83,t_3_84,t_3_85,t_3_86,t_3_87,t_3_88,t_3_89,t_3_
90,t_3_91,t_3_92,t_3_93,t_3_94,t_3_95,t_3_96,t_3_97,t_3_98,t_3_100)
ifile=p_run.ora


--------------------------
init_4.ora (Server 4)
--------------------------
instance_number     =4
thread    =4
rollback_segments =  (
t_4_1,t_4_2,t_4_3,t_4_4,t_4_5,t_4_6,t_4_7,t_4_8,t_4_9,t_4_10,t_4_11
,t_4_12,t_4_13,t_4_14,t_4_15,t_4_16,t_4_17,t_4_18,t_4_19,t_4_20,
t_4_21,t_4_22,t_4_23,t_4_24,t_4_25,t_4_26,t_4_27,t_4_28,t_4_29,t_4_
30,t_4_31,t_4_32,t_4_33,t_4_34,t_4_35,t_4_36,t_4_37,t_4_38,t_4_39,t
_4_40,
t_4_41,t_4_42,t_4_43,t_4_44,t_4_45,t_4_46,t_4_47,t_4_48,t_4_49,t_4_
50,t_4_51,t_4_52,t_4_53,t_4_54,t_4_55,t_4_56,t_4_57,t_4_58,t_4_59,t
_4_60,t_4_61,t_4_62,t_4_63,t_4_64,t_4_65,t_4_66,t_4_67,t_4_68,t_4_6
9,t_4_70,t_4_71,t_4_72,t_4_73,t_4_74,t_4_75,t_4_76,t_4_77,t_4_78,t_
4_79,t_4_80,
t_4_81,t_4_82,t_4_83,t_4_84,t_4_85,t_4_86,t_4_87,t_4_88,t_4_89,t_4_
90,t_4_91,t_4_92,t_4_93,t_4_94,t_4_95,t_4_96,t_4_97,t_4_98,t_4_100)
ifile=p_run.ora


--------------------------
init_5.ora (Server 5)
--------------------------
instance_number     =5
thread    =5
rollback_segments =  (
t_5_1,t_5_2,t_5_3,t_5_4,t_5_5,t_5_6,t_5_7,t_5_8,t_5_9,t_5_10,t_5_11
,t_5_12,t_5_13,t_5_14,t_5_15,t_5_16,t_5_17,t_5_18,t_5_19,t_5_20,
t_5_21,t_5_22,t_5_23,t_5_24,t_5_25,t_5_26,t_5_27,t_5_28,t_5_29,t_5_
30,t_5_31,t_5_32,t_5_33,t_5_34,t_5_35,t_5_36,t_5_37,t_5_38,t_5_39,t
_5_40,
t_5_41,t_5_42,t_5_43,t_5_44,t_5_45,t_5_46,t_5_47,t_5_48,t_5_49,t_5_
50,t_5_51,t_5_52,t_5_53,t_5_54,t_5_55,t_5_56,t_5_57,t_5_58,t_5_59,t
_5_60,t_5_61,t_5_62,t_5_63,t_5_64,t_5_65,t_5_66,t_5_67,t_5_68,t_5_6
9,t_5_70,t_5_71,t_5_72,t_5_73,t_5_74,t_5_75,t_5_76,t_5_77,t_5_78,t_
5_79,t_5_80,
t_5_81,t_5_82,t_5_83,t_5_84,t_5_85,t_5_86,t_5_87,t_5_88,t_5_89,t_5_
90,t_5_91,t_5_92,t_5_93,t_5_94,t_5_95,t_5_96,t_5_97,t_5_98,t_5_100)
ifile=p_run.ora


--------------------------
init_6.ora (Server 6)
--------------------------
instance_number     =6
thread    =6
rollback_segments =  (
t_6_1,t_6_2,t_6_3,t_6_4,t_6_5,t_6_6,t_6_7,t_6_8,t_6_9,t_6_10,t_6_11
,t_6_12,t_6_13,t_6_14,t_6_15,t_6_16,t_6_17,t_6_18,t_6_19,t_6_20,
t_6_21,t_6_22,t_6_23,t_6_24,t_6_25,t_6_26,t_6_27,t_6_28,t_6_29,t_6_
30,t_6_31,t_6_32,t_6_33,t_6_34,t_6_35,t_6_36,t_6_37,t_6_38,t_6_39,t
_6_40,
t_6_41,t_6_42,t_6_43,t_6_44,t_6_45,t_6_46,t_6_47,t_6_48,t_6_49,t_6_
50,t_6_51,t_6_52,t_6_53,t_6_54,t_6_55,t_6_56,t_6_57,t_6_58,t_6_59,t
_6_60,t_6_61,t_6_62,t_6_63,t_6_64,t_6_65,t_6_66,t_6_67,t_6_68,t_6_6
9,t_6_70,t_6_71,t_6_72,t_6_73,t_6_74,t_6_75,t_6_76,t_6_77,t_6_78,t_
6_79,t_6_80,
t_6_81,t_6_82,t_6_83,t_6_84,t_6_85,t_6_86,t_6_87,t_6_88,t_6_89,t_6_
90,t_6_91,t_6_92,t_6_93,t_6_94,t_6_95,t_6_96,t_6_97,t_6_98,t_6_100)
ifile=p_run.ora


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

```
init_7.ora (Server 7)
--------------------------
instance_number     =7
thread    =7
rollback_segments =  (
t_7_1,t_7_2,t_7_3,t_7_4,t_7_5,t_7_6,t_7_7,t_7_8,t_7_9,t_7_10,t_7_11
,t_7_12,t_7_13,t_7_14,t_7_15,t_7_16,t_7_17,t_7_18,t_7_19,t_7_20,
t_7_21,t_7_22,t_7_23,t_7_24,t_7_25,t_7_26,t_7_27,t_7_28,t_7_29,t_7_
30,t_7_31,t_7_32,t_7_33,t_7_34,t_7_35,t_7_36,t_7_37,t_7_38,t_7_39,t
_7_40,
t_7_41,t_7_42,t_7_43,t_7_44,t_7_45,t_7_46,t_7_47,t_7_48,t_7_49,t_7_
50,t_7_51,t_7_52,t_7_53,t_7_54,t_7_55,t_7_56,t_7_57,t_7_58,t_7_59,t
_7_60,t_7_61,t_7_62,t_7_63,t_7_64,t_7_65,t_7_66,t_7_67,t_7_68,t_7_6
9,t_7_70,t_7_71,t_7_72,t_7_73,t_7_74,t_7_75,t_7_76,t_7_77,t_7_78,t_
7_79,t_7_80,
t_7_81,t_7_82,t_7_83,t_7_84,t_7_85,t_7_86,t_7_87,t_7_88,t_7_89,t_7_
90,t_7_91,t_7_92,t_7_93,t_7_94,t_7_95,t_7_96,t_7_97,t_7_98,t_7_100)
ifile=p_run.ora


--------------------------
init_8.ora (Server 8)
--------------------------
instance_number     =8
thread    =8
rollback_segments =  (
t_8_1,t_8_2,t_8_3,t_8_4,t_8_5,t_8_6,t_8_7,t_8_8,t_8_9,t_8_10,t_8_11
,t_8_12,t_8_13,t_8_14,t_8_15,t_8_16,t_8_17,t_8_18,t_8_19,t_8_20,
t_8_21,t_8_22,t_8_23,t_8_24,t_8_25,t_8_26,t_8_27,t_8_28,t_8_29,t_8_
30,t_8_31,t_8_32,t_8_33,t_8_34,t_8_35,t_8_36,t_8_37,t_8_38,t_8_39,t
_8_40,
t_8_41,t_8_42,t_8_43,t_8_44,t_8_45,t_8_46,t_8_47,t_8_48,t_8_49,t_8_
50,t_8_51,t_8_52,t_8_53,t_8_54,t_8_55,t_8_56,t_8_57,t_8_58,t_8_59,t
_8_60,t_8_61,t_8_62,t_8_63,t_8_64,t_8_65,t_8_66,t_8_67,t_8_68,t_8_6
9,t_8_70,t_8_71,t_8_72,t_8_73,t_8_74,t_8_75,t_8_76,t_8_77,t_8_78,t_
8_79,t_8_80,
t_8_81,t_8_82,t_8_83,t_8_84,t_8_85,t_8_86,t_8_87,t_8_88,t_8_89,t_8_
90,t_8_91,t_8_92,t_8_93,t_8_94,t_8_95,t_8_96,t_8_97,t_8_98,t_8_100)
ifile=p_run.ora


--------------------------
p_build.ora
--------------------------
compatible = 9.0.2.0.0
db_name = tpcc
control_files = "\\.\control_001"
sort_area_size = 10485760
recovery_parallelism = 20
dml_locks = 2000
log_buffer = 1048576
shared_pool_size = 262144000
cursor_space_for_time = TRUE
db_block_size = 4096
_allocate_creation_order      =TRUE
db_files=2000
cluster_database              = FALSE
processes                     = 100
use_indirect_data_buffers         = TRUE
db_block_buffers        = 1472800
log_parallelism = 2
instance_number     =1
thread    =1
java_pool_size = 15728640


--------------------------
p_create.ora
--------------------------
compatible = 9.0.1.0.0
db_name = tpcc
control_files = "\\.\control_001"
processes =200
db_block_size = 4096
_allocate_creation_order      =TRUE
db_files=600
db_cache_size = 800M


--------------------------
p_run.ora
--------------------------
db_name = tpcc
compatible = 9.0.2.0.0
control_files = "\\.\control_001"
db_block_buffers    = 1572864
buffer_pool_keep    = (buffers:1241804,lru_latches:4)
buffer_pool_recycle = (buffers:12800,lru_latches:4)
sort_area_size = 10485760
parallel_max_servers = 0
parallel_min_servers = 0
dml_locks = 2000
log_buffer = 10048576
cursor_space_for_time = TRUE
db_block_size = 4096
db_files=2000
cursor_space_for_time             = TRUE
_db_writer_max_writes                     = 1024
_db_writer_chunk_writes                   = 200
shared_pool_size = 262144000
use_indirect_data_buffers       = TRUE
db_block_size                     = 4096
_log_simultaneous_copies        = 16
dml_locks                         = 50
enqueue_resources                 = 200
hash_join_enabled                 = FALSE
log_archive_start                 = FALSE
log_checkpoint_interval           = 0
log_checkpoint_timeout                    = 0
log_checkpoints_to_alert          = TRUE
max_rollback_segments             = 100
pre_page_sga                      = TRUE
processes                         = 150
sessions                        = 240
open_cursors                      = 180
remote_login_passwordfile         = shared
replication_dependency_tracking   = FALSE
transactions_per_rollback_segment = 1
timed_statistics                  = FALSE
transaction_auditing              = FALSE
db_block_checksum               = FALSE
db_block_checking               = FALSE
cluster_database                = TRUE
_db_block_lru_latches = 64
gc_files_to_locks             = " 1-8=1000EACH:\
9-54=1EACH:\
55-57=50000:\
58-60=50000:\
61-63=50000:\
64-66=50000:\
67-69=50000:\
70-72=50000:\
73-75=50000:\
76-78=50000:\
79-344=1EACH:\
345-350=175000:\
351-356=175000:\
357-362=175000:\
363-368=175000:\
369-374=175000:\
375-380=175000:\
381-386=175000:\
387-392=175000:\
393-416=1EACH:\
421-422=1EACH"
_kcl_name_table_latches = 64
db_writer_processes = 2
db_cache_advice = off
_lm_lms = 1
_lm_dd_interval= 30
statistics_level = basic
log_parallelism = 2
java_pool_size = 15728640




--------------------------
listener.ora (Server 1)
--------------------------
# LISTENER.ORA.RA2 Network Configuration File:
C:\oracle\ora90\network\admin\listener.ora.ra2
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = ra1)(PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\ora90)
      (PROGRAM = extproc)
    )
  )



--------------------------
listener.ora (Server 2)
--------------------------
# LISTENER.ORA.RA2 Network Configuration File:
C:\oracle\ora90\network\admin\listener.ora.ra2
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = ra2)(PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\ora90)
      (PROGRAM = extproc)
    )
  )
```

```
--------------------------
listener.ora (Server 3)
--------------------------
# LISTENER.ORA.RA7 Network Configuration File:
C:\oracle\ora90\network\admin\listener.ora.ra7
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = ra3)(PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\ora90)
      (PROGRAM = extproc)
    )
  )



--------------------------
listener.ora (Server 4)
--------------------------
# LISTENER.ORA.RA2 Network Configuration File:
C:\oracle\ora90\network\admin\listener.ora.ra2
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = ra4)(PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\ora90)
      (PROGRAM = extproc)
    )
  )



--------------------------
listener.ora (Server 5)
--------------------------
# LISTENER.ORA.RA7 Network Configuration File:
C:\oracle\ora90\network\admin\listener.ora.ra7
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = ra5)(PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\ora90)
      (PROGRAM = extproc)
    )
  )



--------------------------
listener.ora (Server 6)
--------------------------
# LISTENER.ORA.RA6 Network Configuration File:
C:\oracle\ora90\network\admin\listener.ora.ra6
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = ra6)(PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\ora90)
      (PROGRAM = extproc)
    )
  )



--------------------------
listener.ora (Server 7)
--------------------------
# LISTENER.ORA.RA7 Network Configuration File:
C:\oracle\ora90\network\admin\listener.ora.ra7
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = ra7)(PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\ora90)
      (PROGRAM = extproc)
    )
  )



--------------------------
listener.ora (Server 8)
--------------------------
# LISTENER.ORA.RA7 Network Configuration File:
C:\oracle\ora90\network\admin\listener.ora.ra7
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = ra8)(PORT = 1521))
      )
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\ora90)
      (PROGRAM = extproc)
    )
  )
```

Windows Server, BEA Tuxedo, Oracle (Clients)

```
--------------------------
Windows 2000 Registry
--------------------------

Key Name:          SOFTWARE\BEA
Systems\TUXEDO\6.5\IPCResources\test
Class Name:        <NO CLASS>
Last Write Time:   4/29/2002 - 1:36 PM
Value 0
  Name:            TUXIPC_MSG_BYTES
  Type:            REG_DWORD
  Data:            0x10000

Value 1
  Name:            TUXIPC_MSG_HDRS
  Type:            REG_DWORD
  Data:            0x2000

Value 2
  Name:            TUXIPC_MSG_QUEUE_BYTES
  Type:            REG_DWORD
  Data:            0xc0000000

Value 3
  Name:            TUXIPC_MSG_QUEUES
  Type:            REG_DWORD
  Data:            0x320
```

```
Value 4
  Name:            TUXIPC_MSG_SEG_BYTES
  Type:            REG_DWORD
  Data:            0x100

Value 5
  Name:            TUXIPC_MSG_SEGS
  Type:            REG_DWORD
  Data:            0x7fff

Value 6
  Name:            TUXIPC_PROC
  Type:            REG_DWORD
  Data:            0x2ee0

Value 7
  Name:            TUXIPC_SEM
  Type:            REG_DWORD
  Data:            0x2ee0

Value 8
  Name:            TUXIPC_SEM_IDS
  Type:            REG_DWORD
  Data:            0x2ee0

Value 9
  Name:            TUXIPC_SEM_UNDO
  Type:            REG_DWORD
  Data:            0x2ee0

Value 10
  Name:            TUXIPC_SHM_PROCS
  Type:            REG_DWORD
  Data:            0x186a0

Value 11
  Name:            TUXIPC_SHM_SEGS
  Type:            REG_DWORD
  Data:            0x350




Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo
Class Name:        <NO CLASS>
Last Write Time:   1/24/2002 - 2:33 PM

Key Name:
SYSTEM\CurrentControlSet\Services\InetInfo\Parameters
Class Name:        <NO CLASS>
Last Write Time:   4/29/2002 - 3:02 PM
Value 0
  Name:            BandwidthLevel
  Type:            REG_DWORD
  Data:            0xffffffff

Value 1
  Name:            DispatchEntries
  Type:            REG_MULTI_SZ
  Data:            LDAPSVC
                   SMTPSVC
Value 2
  Name:            ListenBackLog
  Type:            REG_DWORD
  Data:            0xfa

Value 3
  Name:            MaxConnections
  Type:            REG_DWORD
  Data:            0x1e78

Value 4
  Name:            MaxPoolThreads
  Type:            REG_DWORD
  Data:            0x14

Value 5
  Name:            PoolThreadLimit
  Type:            REG_DWORD
  Data:            0xfe

Value 6
  Name:            ThreadTimeout
  Type:            REG_DWORD
  Data:            0x15180




Key Name:          SOFTWARE\Microsoft\TPCC
Class Name:        <NO CLASS>
Last Write Time:   5/14/2002 - 2:23 PM
Value 0
  Name:            CkptPassword
  Type:            REG_SZ
  Data:            tpcc

Value 1
  Name:            CkptUser
  Type:            REG_SZ
  Data:            tpcc

Value 2
  Name:            Database
  Type:            REG_SZ
  Data:            tpcc

Value 3
  Name:            DeadlockRetry
  Type:            REG_SZ
  Data:            3

Value 4
  Name:            DeliveryServerConnectsToDB
  Type:            REG_SZ
  Data:            FALSE

Value 5
  Name:            DisplayDeliveryCompletions
  Type:            REG_SZ
  Data:            TRUE

Value 6
  Name:            FlushDeliveryLog
  Type:            REG_SZ
  Data:            FALSE

Value 7
  Name:            GenericTransactionServers
  Type:            REG_SZ
  Data:            1

Value 8
  Name:            LOG
  Type:            REG_SZ
  Data:            OFF

Value 9
  Name:            LoginDelay
  Type:            REG_SZ
  Data:            500

Value 10
  Name:            MaxConnections
  Type:            REG_SZ
  Data:            8000

Value 11
  Name:            maxDBConnections
  Type:            REG_SZ
  Data:            80

Value 12
  Name:            MaximumWarehouses
  Type:            REG_SZ
  Data:            12024

Value 13
  Name:            NumberOfDeliveryThreads
  Type:            REG_SZ
  Data:            10

Value 14
  Name:            NumberOfQueuedDeliveryTransactions
  Type:            REG_SZ
  Data:            256

Value 15
  Name:            OracleOPS
  Type:            REG_SZ
  Data:            TRUE

Value 16
  Name:            Password
  Type:            REG_SZ
  Data:            tpcc

Value 17
  Name:            PATH
  Type:            REG_SZ
  Data:            c:\inetpub\wwwroot\

Value 18
  Name:            Server
  Type:            REG_SZ
  Data:            tpc1

Value 19
  Name:            UseDeliveryPipe
  Type:            REG_SZ
  Data:            FALSE

Value 20
  Name:            User
  Type:            REG_SZ
  Data:            tpcc

Value 21
  Name:            WriteDeliveryLog
  Type:            REG_SZ
  Data:            FALSE


--------------------------
ubbconfig_24c
--------------------------
#
# 9i RAC UBBconfig file for 24 clients configuration
#
# Clients systems have indentical configuration except:
# IPCKEY 4000[1-24] on client[1-24]
# MASTER OC[1-24] on Client[1-24]
# LMID OC[1=24] on Client[1-24]
```

```
#
#------------------------------------------------------------------
-----------
*RESOURCES
#------------------------------------------------------------------
-----------
IPCKEY              40001
MASTER              OC1
MAXACCESSERS        7800      # 1024 or more
MAXGTT 2048
MAXSERVERS          128
MAXSERVICES         1000      #MAXSERVERS * #-of-services-each-
server + 10 (for BBL)
MODEL               SHM
LDBAL               Y
*MACHINES
DEFAULT:

          TUXCONFIG="c:\inetpub\wwwroot\tuxconfig"
                              TUXDIR="c:\progra~1\beasys~1\tuxedo"
                              APPDIR="c:\inetpub\wwwroot"
                              UID=0
                              GID=0
                              TYPE="WinNT"
OC1       LMID=OC1

*GROUPS
TPCC1
          LMID=OC1 GRPNO=1 OPENINFO=NONE

*SERVERS
DEFAULT:
          CLOPT="-A"
tuxora9   SRVGRP=TPCC1 SRVID=1
RQADDR=txnque1 REPLYQ=Y
MIN=23 MAX=23

*SERVICES

DEFAULT:
                    LOAD=1
                    PRIO=1
                    BUFTYPE="CARRAY"
                    TRANTIME=900
                    AUTOTRAN=N
dy_transaction
no_transaction
os_transaction
pt_transaction
sl_transaction


--------------------------
tnsnames.ora
--------------------------
oracle =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.210.147)(PORT =
1521)))
    (CONNECT_DATA =
          (SDU=4096)
          (TDU=16384)
        (SERVICE_NAME = tpcc)))
tpc1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.210.101)(PORT =
1521)))
    (CONNECT_DATA =
          (SDU=4096)
          (TDU=16384)
        (SERVICE_NAME = tpcc)))
tpc2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.210.102)(PORT =
1521)))
    (CONNECT_DATA =
          (SDU=4096)
          (TDU=16384)
        (SERVICE_NAME = tpcc)))
tpc3 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.210.103)(PORT =
1521)))
    (CONNECT_DATA =
          (SDU=4096)
          (TDU=16384)
        (SERVICE_NAME = tpcc)))
tpc4 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.210.104)(PORT =
1521)))
    (CONNECT_DATA =
          (SDU=4096)
          (TDU=16384)
        (SERVICE_NAME = tpcc)))
tpc5 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.210.105)(PORT =
1521)))
    (CONNECT_DATA =
          (SDU=4096)
          (TDU=16384)
          (SERVICE_NAME = tpcc)))
tpc6 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.210.106)(PORT =
1521)))
    (CONNECT_DATA =
          (SDU=4096)
          (TDU=16384)
          (SERVICE_NAME = tpcc)))
tpc7 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.210.107)(PORT =
1521)))
    (CONNECT_DATA =
          (SDU=4096)
          (TDU=16384)
          (SERVICE_NAME = tpcc)))
  )
tpc8 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 130.168.210.108)(PORT =
1521)))
    (CONNECT_DATA =
          (SDU=4096)
          (TDU=16384)
          (SERVICE_NAME = tpcc)))
```

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

May 21, 2002

Compaq Computer
Corporation
Paul Cao
MS150402
20555 SH 249
Houston, TX 77070

Paul:

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

All pricing shown is in US Dollars ($).

Part Number	Description	Unit Price	Quantity	Price
C11-00821	**Windows 2000 Server** *Server license only - No CALs* *Discount schedule: Open Program - No Level*	$738	24	$17,712
C10-00475	**Windows 2000 Advanced Server** *Server license only - No CALs* *Discount schedule: Open Program - No Level*	$2,399	8	$19,192
048-00317	**Visual C++ Professional 6.0 Win32**	$ 549	1	$ 549
	3-year maintenance for above software	$1,950	1	$5,850

All products are currently orderable through Microsoft's normal distribution channels.

This quote is valid for the next 90 days.

If we can be of any further assistance, please contact Jamie Reding at (425) 703-0510 or jamiere@microsoft.com.

Reference ID: PCpaca0221058535
Please include this Reference ID in any correspondence regarding this price quote.

THE ECOMMERCE TRANSACTION PLATFORM

August 13, 2002

Raghunath K. Othayoth
ISS - Solutions and Strategy
Hewlett Packard Company
281-518-2748 tel
281-514-8375 fax

Per your request I am enclosing the pricing information regarding TUXEDO 6.5 that you requested. This pricing applies to Tuxedo 6.4, 6.5, 7.1, and 8.0. Please note that Tuxedo 8.0 is our most recent version of Tuxedo. Core functionality services pricing is appropriate for your activities. As per the table below HP/Compaq systems are classified as either a Tier 1, 2, 3, 4 or 5 systems depending on the performance and CPU capacity of the system. The Compaq DL 360 or Intel 2P machines are Tier 1 machines – price is $3,000 per server (License) + $630 per server (7x24) for support. This quote is valid for 60 days from the date of this letter.

Tuxedo Core Functionality Services (CFS) Program Product Pricing and Description

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.4, 6.5, 7.1, and 8.0. Prices range from $3,000 for Tier 1 to $250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

Very Truly Yours,

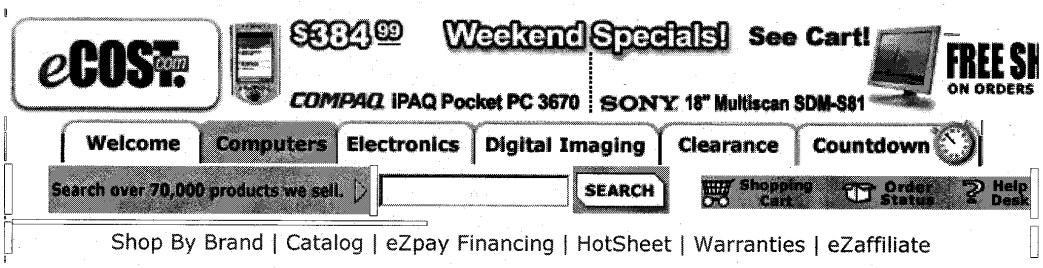
Robert J. Gieringer

Rob Gieringer,
Worldwide Pricing Manager

BEA Tux/CFS Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 9) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers	Unlimited	$3,000.00	$540.00	$630.00
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations with up to 2 CPUs	Unlimited	$12,000.00	$2,160.00	$2,520.00
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	Unlimited	$30,000.00	$5,400.00	$6,300.00
Tier 4 - Large (more than 8, less than 32 CPUs)	Unlimited	$100,000.00	$18,000.00	$21,000.00
Tier 5 - Massively Parallel Systems, > 32 processors	Unlimited	$250,000.00	$45,000.00	$52,500.00

	Tier 1	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Operating System						
HP/UX 9.X;10.X	Uni-processor Workstation B Class - 132/180/2000 C Class (3000/3600/ 3700) 2P Client Machines Compaq DL360	9000/E25 9000/E35 9000/E45 9000/E55 9000/G30 9000/G40 9000/A180 9000/A180C 9000 /A400	9000/G50 9000/G60 Multi-Processor Workstations J Class (J282/J2240/J5600/ J6000/J6700) 9000/R380,390 9000/D200,210 220/30/50/60/80 D310/20/30 D350/60/70/80 9000 /A500 9000 – L1000 9000 – R Class	9000/H20, 30 9000/H40, 50 9000/I30, 40 9000/K1XX 9000 – L2000/L3000 9000/I50,60 9000/H60 9000/G70 9000/H70 9000/I70 9000/K2XX 9000/K3XX 9000/K4XX 9000/K5XX N4xxx Series	9000/T500,T5 20,T600 1-16 CPUs S-Class	9000/V series all models X-Class 9000 Series - Superdome



Shop By Brand | Catalog | eZpay Financing | HotSheet | Warranties | eZaffiliate

Refine Your Search Click here for Advance Search

Find All Products **manufactured by** All Manufacturers **in order of** Category

Limit results to products containing the phrase "3c16980a-us".

SEARCH

Page 1 of 1 : 1 1 - 1 of 1 result(s).

Networking Hardware

Product Description	Platform	Price	Availability	Compare
3Com **SuperStack 3 3300 24 Port 10/100 Switch** Stackable modular 10/100 switch that provides 24 ports of flexible switching for workgroups and backbones.	**PC and Mac**	**$999.95** #900830	Same Day **Buy Now!**	☐

Page 1 of 1 : 1 1 - 1 of 1 result(s).

Home - Help Desk - Shopping Cart - Order Status - How to Order - Why eCOST.com - Customer Se
eZpay Financing Plan - HotSheet - Order By Phone - Privacy Policy - Terms of Service

Appendix G:

Oracle9i Database Enterprise Edition Release 2, v9.2.0.1 for Windows 2000
Processor 3 year term for 32 processors, Unlimited Users: $640,000

Real Application Clusters, Processor 3 year term for 32 processors,
Unlimited Users: $320,000

Partitioning, Processor 3 year term for 32 processors, Unlimited Users:
$160,000

Oracle Database Server Support Package for 3 years: $48,000

Mandatory E-Business Discount: <$292,000>

Total Oracle price: $876,000

Oracle pricing contact: MaryBeth Pierantoni
mary.beth.pierantoni@oracle.com
650-506-2118